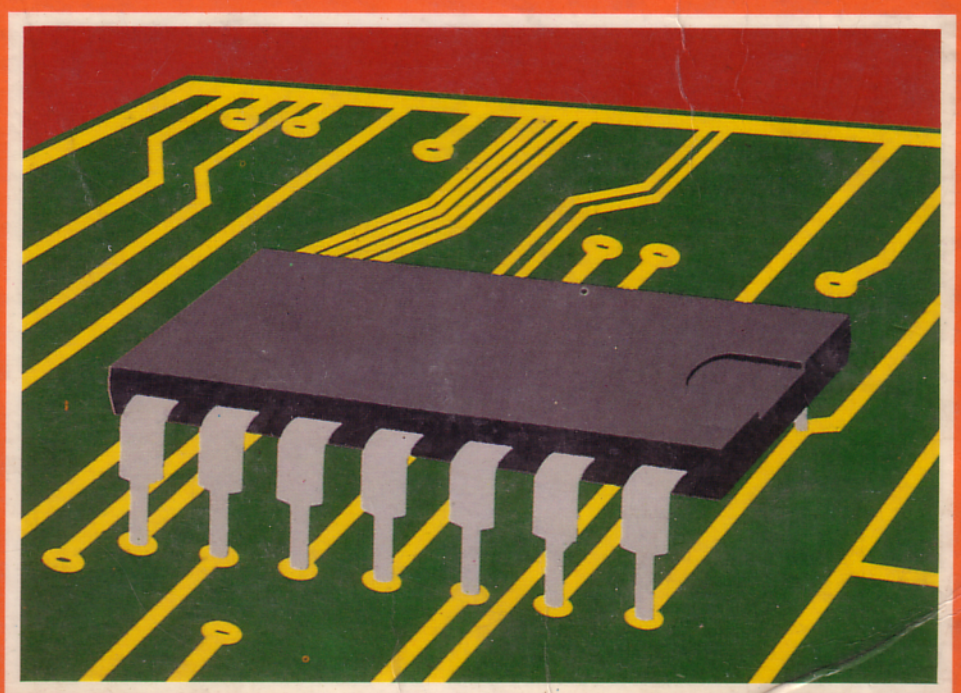
Schaum

FUNDAMENTOS DE LOS MICROPROCESADORES

SEGUNDA EDICION

Roger L. Tokheim



McGraw Hill

FUNDAMENTOS DE LOS MICROPROCESADORES

Segunda edición

FUNDAMENTOS DE LOS MICROPROCESADORES

Segunda edición

ROGER L. TOKHEIM
Henry Sibley High School
Mendota Heights, Minnesota

Traducción
JUAN MANUEL SANCHEZ
Departamento de Informática y Automática
Facultad de Ciencias Físicas
Universidad Complutense de Madrid

Revisión técnica
ANTONIO VAQUERO
Catedrático de Informática
Facultad de Ciencias Físicas
Universidad Complutense de Madrid

MEXICO

87560

McGRAW-HILL

MÉXICO • BUENOS AIRES • CARACAS • GUATEMALA
LISBOA • MADRID • NUEVA YORK • PANAMÁ • SAN JUAN
SANTAFÉ DE BOGOTÁ • SANTIAGO • SÃO PAULO
AUCKLAND • HAMBURGO • LONDRES • MILÁN • MONTREAL
NUEVA DELHI • PARÍS • SAN FRANCISCO • SINGAPUR
ST. LOUIS • SIDNEY • TOKIO • TORONTO

PROLOGO

El estudiante de electrónica o especialista del futuro necesita conocer los microprocesadores y sistemas basados en microprocesador. Este conocimiento incluye tanto el hardware como el software de las microcomputadoras. Los microprocesadores son la base de un nuevo tipo de dispositivos inteligentes; se encuentran en diversos productos: desde los juguetes de los niños a los automóviles, desde las microcomputadoras a los electrodomésticos, desde los robots y maquinaria industrial al equipo de entretenimiento casero. Gracias a estos dispositivos programables, denominados microprocesadores, se espera que el desarrollo de máquinas inteligentes se acelere en el futuro.

La segunda edición de *Fundamentos de los Microprocesadores* de la serie Schaum proporciona al lector los aspectos básicos, que normalmente se cubren en un primer curso de microprocesadores. La filosofía de la serie Schaum, de concentrarse en los problemas característicos que surgen cuando se estudia cualquier materia, está soportada en este libro gracias a la inclusión de unos 1.100 problemas, suplementarios y resueltos, sobre microprocesadores y microcomputadoras. Las soluciones completas vienen con la mayoría de los problemas de este libro.

La segunda edición del libro contiene la mayor parte de los temas que hicieron que la primera edición tuviese un gran éxito. En la segunda edición se han añadido algunas cosas que reflejan nuevas tendencias prácticas, cambios tecnológicos y sugerencias de estudiantes e instructores. Se han añadido dos capítulos que cubren los microprocesadores de 8 bits 6502/65C02 en sus aspectos hardware y de programación. Por demanda popular, se ha añadido un capítulo donde se examinan algunos microprocesadores seleccionados de 16 y 32 bits. Entre éstos se encuentran los Intel de 16 y 32 bits: 8086, 8088, 80186, 80188, 80286 y 80386; además de los microprocesadores Motorola 68000, 68008, 68010, 68012, 68020 y 68030. Finalmente, se examinan los microprocesadores de 16 bits 65802 y 65816 de Western Design Center.

Los temas tratados en este libro se seleccionaron cuidadosamente para que coincidiesen con los cursos impartidos en los niveles de escuela superior, profesional, colegio de comunidad y colegio de principiantes. Se analizaron muchos de los libros de texto más utilizados a nivel profesional y manuales de laboratorio sobre microprocesadores y microcomputadoras. Los temas y problemas incluidos en este libro de la serie Schaum son similares a los encontrados en los libros analizados. Los segmentos de los programas listados en este libro han sido cuidadosamente comprobados para que funcionen en microcomputadoras de bajo costo. Algunos de estos entrenadores están disponibles en Intel Corporation, Heath/Zenith y Sistemas Lab-Volt.

La segunda edición de *Fundamentos de los Microprocesadores* de Schaum comienza con una corta introducción a las computadoras, a continuación se presenta información general sobre números, códigos y aritmética de computadoras junto con una breve revisión de los dispositivos digitales básicos. Los sistemas comerciales basados en microprocesador son muy complejos, esta es la razón por la que se utiliza un microprocesador genérico simplificado para introducir los fundamentos de las microcomputadoras, el microprocesador, la programación del microprocesador y las interfaces con el microprocesador. El conjunto de registros y el repertorio de instrucciones del microprocesador genérico son subconjuntos de los de la MPU 8080/8085. A continuación, se cubren con cierto detalle tres de los microprocesadores de 8 bits más populares: las unidades microprocesadoras descritas son la Intel 8080/8085, la Motorola 6800 y la 6502/65C02:

dedicándose dos capítulos a cada microprocesador. El capítulo final examina brevemente otros microprocesadores más complejos de 16 y 32 bits de Intel, Motorola y Western Design Center.

Extendemos nuestro agradecimiento a John Aliano de McGraw-Hill por su asistencia y paciencia: al estudiante de ingeniería Darrel Klotzbach por sus buenos efectos de investigación, corrección meticulosa de las pruebas y comprobación cuidadosa de la mayoría de los programas de este libro. Agradezco de manera especial a mi familia, Caroline, Dan y Marshall su ayuda, paciencia y estímulo.

ROGER L. TOKHEIM

CONTENIDO

PROLOGO v

Capítulo 1. INTRODUCCION A LAS COMPUTADORAS 1

1.1. Introducción 1
1.2. Organización de la computadora 1
1.3. Organización de la microcomputadora 3
1.4. Operación de la microcomputadora 5

Capítulo 2. NUMEROS, CODIGOS DE COMPUTADORA Y ARITMETICA 13

2.1. Números binarios 13
2.2. Números hexadecimales 16
2.3. Números BCD 20
2.4. Aritmética binaria 21
2.5. Notación en complemento a 2 24
2.6. Aritmética en complemento a 2 28
2.7. Agrupaciones de bits 31
2.8. Códigos alfanuméricos 35

Capítulo 3. DISPOSITIVOS DIGITALES BASICOS 39

3.1. Puertas lógicas 39
3.2. Combinación de puertas lógicas 43
3.3. Flip-flops y cerrojos 46
3.4. Codificadores, decodificadores y visualizadores de siete segmentos 52
3.5. Buffers y dispositivos de tres estados 55
3.6. Memorias semiconductoras 57
3.7. Utilización de las RAM y ROM 60

Capítulo 4. FUNDAMENTOS DE LAS MICROCOMPUTADORAS 69

4.1. Introducción 69
4.2. Arquitectura simplificada de una microcomputadora 69
4.3. Organización simplificada de la memoria 74

4.4. Repertorio de instrucciones 79
4.5. Organización simplificada de la CPU 84
4.6. Operación de la microcomputadora 89

Capítulo 5. **EL MICROPROCESADOR** **100**

5.1. Características comunes de los microprocesadores 100
5.2. Descripciones de la hoja de datos 103
5.3. Funciones y diagramas de las patillas 110
5.4. Arquitectura del microprocesador 114
5.5. Utilización del registro de direcciones/datos 120
5.6. Utilización del puntero de pila 127

Capítulo 6. **PROGRAMACION DEL MICROPROCESADOR** **136**

6.1. Lenguajes máquina y ensamblador 136
6.2. Repertorio simplificado de instrucciones 140
6.3. Repertorio de instrucciones: operaciones aritméticas 142
6.4. Repertorio de instrucciones: operaciones lógicas 152
6.5. Repertorio de instrucciones: operaciones de transferencia de datos 158
6.6. Repertorio de instrucciones: operaciones de bifurcación 165
6.7. Repertorio de instrucciones: operaciones de llamada y vuelta de subrutina ... 169
6.8. Repertorio de instrucciones: operaciones misceláneas 174
6.9. Escritura de un programa 176
6.10. Modos de direccionamiento 183
6.11. Bifurcación en programas 188
6.12. Programas cíclicos 192
6.13. Utilización de subrutinas 196

Capítulo 7. **INTERFACES CON EL MICROPROCESADOR** **213**

7.1. Introducción 213
7.2. Interfaz con la ROM 215
7.3. Interfaz con la RAM 219
7.4. Interfaces básicas de entrada/salida 224
7.5. Interfaz con puertos prácticos de entrada/salida 230
7.6. Sincronización de transferencias de entrada/salida de datos mediante interrupciones 233
7.7. Decodificación de direcciones 237

Capítulo 8. **EL MICROPROCESADOR INTEL 8080/8085** **247**

8.1. Introducción 247
8.2. Funciones y diagrama de patillas del 8085 248
8.3. Arquitectura del 8085 253
8.4. Modos de direccionamiento 261

8.5. Repertorio de instrucciones del 8080/8085 266
8.6. Instrucciones de transferencia de datos del 8080/8085 272
8.7. Instrucciones aritméticas del 8080/8085 279
8.8. Instrucciones lógicas del 8080/8085 289
8.9. Instrucciones de bifurcación del 8080/8085 295
8.10. Instrucciones de pila, E/S y control de máquina del 8080/8085 302

Capítulo 9. **PROGRAMACION DEL MICROPROCESADOR 8080/8085** **318**

9.1. Introducción 318
9.2. Programas lineales 324
9.3. Programas cíclicos 329
9.4. Programas matemáticos 335

Capítulo 10. **EL MICROPROCESADOR MOTOROLA 6800** **345**

10.1. Introducción 345
10.2. Funciones y diagrama de patillas del 6800 347
10.3. Arquitectura del 6800 351
10.4. Modos de direccionamiento 367
10.5. Repertorio de instrucciones del 6800 376
10.6. Instrucciones de memoria y acumulador del 6800 383
10.7. Instrucciones del puntero de pila y registro índice del 6800 388
10.8. Instrucciones de bifurcación y salto del 6800 390
10.9. Instrucciones del registro de códigos de condición del 6800 397

Capítulo 11. **PROGRAMACION DEL MICROPROCESADOR 6800** **405**

11.1. Introducción 405
11.2. Interpretación de un programa sencillo del 6800 412
11.3. Utilización del registro índice 418
11.4. Un programa del 6800 que utiliza una tabla de búsqueda *(look-up)* 422

Capítulo 12. **EL MICROPROCESADOR 6502** **432**

12.1. Introducción 432
12.2. Funciones y diagrama de patillas del 6502 434
12.3. Arquitectura del 6502 438
12.4. Modos sencillos de direccionamiento 451
12.5. Más modos de direccionamiento del 6502 456
12.6. Repertorio de instrucciones del 6502 464
12.7. Instrucciones de transferencia de datos del 6502 472
12.8. Instrucciones de flujo de control del 6502 475
12.9. Instrucciones aritméticas del 6502 478
12.10. Instrucciones lógicas y de manipulación de bits del 6502 483
12.11. Instrucciones de interrupción y de control del sistema 6502 488

Capítulo 13. **PROGRAMACION DEL MICROPROCESADOR 6502** **494**

13.1. Introducción 494
13.2. Interpretación de un programa de suma BCD 498
13.3. Multiplicación binaria 504
13.4. Decodificación hexadecimal a siete segmentos 511
13.5. Utilización de subrutinas 518

Capítulo 14. **ALGUNOS MICROPROCESADORES DE 16 Y 32 BITS** **530**

14.1. Introducción 530
14.2. Intel 8086 y 8088 531
14.3. Intel 80186, 80188 y 80286 535
14.4. Intel 80386 538
14.5. Motorola 68000, 68008, 68010 y 68012 541
14.6. Motorola 68020 y 68030 546
14.7. Western Design Center 65802 y 65816 549

INDICE **555**

Capítulo 1

INTRODUCCION A LAS COMPUTADORAS

1.1. INTRODUCCION

Las computadoras se han venido utilizando de forma general desde los años 1950. En un principio las computadoras digitales eran máquinas grandes y caras, utilizadas por los gobiernos y grandes empresas. El tamaño y forma de la computadora digital ha cambiado gracias a un nuevo dispositivo denominado microprocesador. El *microprocesador* es un CI (circuito integrado) que tiene la mayoría de las capacidades de procesamiento de las grandes computadoras. El microprocesador es un pequeño, pero extremadamente complejo, dispositivo LSI (gran escala de integración) programable. Las computadoras utilizan un programa almacenado. Las pequeñas computadoras, llamadas *microcomputadoras*, también utilizan el concepto de *programa almacenado*. Una microcomputadora contiene un microprocesador y como mínimo algún tipo de memoria semiconductora.

Las grandes y caras computadoras habitualmente son unidades de *propósito general*. Normalmente se programan muchas veces y se utilizan para realizar muchas tareas. Las *computadoras dedicadas* se emplean cada vez más, debido al uso de los pequeños y baratos microprocesadores. Una computadora dedicada se programa para realizar sólo unas pocas tareas, como ocurre en los juguetes, termostatos, automóviles y herramientas.

1.2. ORGANIZACION DE LA COMPUTADORA

Las secciones tradicionales de una computadora digital se muestran en la Figura 1.1. El sistema básico de la computadora consta de cinco unidades: la unidad de *entrada*, las unidades *aritmética y de control* (contenidas en la CPU, o unidad central de tratamiento), la unidad de *memoria* y la unidad de *salida*. Esta organización de las partes funcionales se denomina *arquitectura* de la computadora.

Las unidades físicas representadas por bloques de la Figura 1.1 constituyen el *hardware*. Para que sea útil, la *memoria de programa* debe decir a la CPU qué tiene que hacer. La preparación de la lista de *instrucciones* se denomina *programación*. La lista de instrucciones constituye un *programa* que se almacena de forma temporal o permanente en la memoria de programa. Estos programas manipulan informaciones denominadas *datos*. *Software* es un término general utilizado para denominar a todos los programas; si el software se almacena permanentemente en la memoria programa, a veces, se denomina *firmware*.

En resumen, la computadora funciona de la siguiente forma: los programas y datos son introducidos en la CPU y transferidos a sus respectivas posiciones de memoria. La CPU lee la

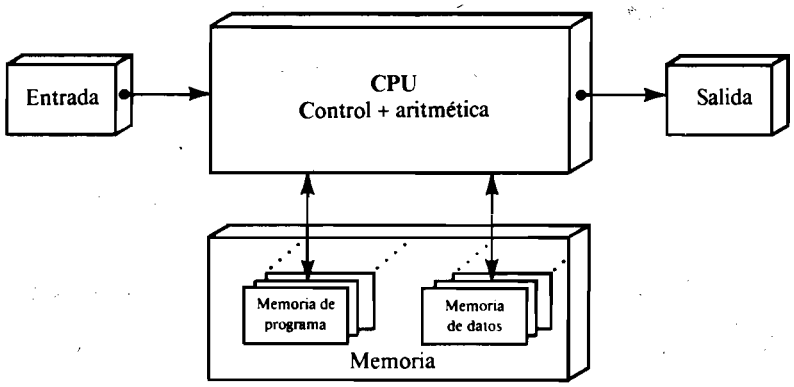


Figura 1.1. Organización general de una computadora.

primera instrucción de la memoria de programa y la ejecuta. Las instrucciones pueden ser tan simples como SUMAR (ADD) dos números, TRANSFERIR (MOVE) un dato, INTRODUCIR o SACAR (INPUT u OUTPUT) un dato, o BIFURCAR (JUMP) a una parte diferente del programa. Cuando finalizan las manipulaciones de los datos, los resultados son transferidos a la salida de la computadora. De nuevo, la mayoría de las acciones de la CPU son debidas a las instrucciones almacenadas en el programa de memoria.

PROBLEMAS RESUELTOS

1.1. Listar las cinco secciones funcionales de una computadora digital.

Solución:

Las cinco secciones funcionales de una computadora se muestran en la Figura 1.1 y son las unidades de entrada, control, aritmética, memoria y salida. CPU es un término general que incluye el control y la aritmética, así como algunos otros elementos.

1.2. Las unidades electrónicas reales como, por ejemplo, las representadas por las cajas de la Figura 1.1 se denominan _____, mientras que los programas que instruyen a la computadora se denominan software.

Solución:

Las unidades electrónicas físicas reales de una computadora se denominan hardware.

1.3. Una persona llamada programador escribe una lista de _____, que se denomina programa.

Solución:

Un programador escribe una lista de instrucciones que se denomina programa.

1.4. Listar dos tipos de información que se introduzca y almacene en la memoria de una computadora.

Solución:

El programa y los datos deben ser introducidos en la computadora y almacenados en memoria.

1.5. La mayoría de las acciones de la CPU en una computadora son debidas a las instrucciones almacenadas en la memoria de _____ .

Solución:

La mayoría de las acciones de la CPU son debidas a las instrucciones almacenadas en la memoria de programa. Los datos que se van a procesar se almacenan en la memoria de datos. En algunas computadoras, no hay diferencia física entre la memoria de programa y la de datos.

1.3. ORGANIZACION DE LA MICROCOMPUTADORA

Un sistema microcomputador es una computadora digital. Se utiliza el prefijo *micro* debido a su pequeño tamaño y bajo costo. El *microprocesador* generalmente forma la sección de la CPU de un sistema microcomputador, cuya organización está esbozada en la Figura 1.2. La microcomputadora contiene las cinco secciones básicas de una computadora: (1) la unidad de *entrada,* (2) las unidades de *control* y (3) *aritmética* contenidas en el microprocesador, (4) la unidad de *memoria* y (5) la unidad de *salida.*

El microprocesador controla todas las unidades del sistema utilizando las líneas de control que aparecen a la izquierda de la Figura 1.2. El *bus de direcciones* que aparece junto a las líneas de control (16 conductores en paralelo) selecciona una cierta posición de memoria, puerto de entrada o puerto de salida. El *bus de datos* (ocho conductores paralelos), a la derecha de la Figura 1.2, es un *camino de doble dirección* que se utiliza para introducir y sacar datos de la unidad microprocesadora. Es importante observar que la unidad microprocesadora (MPU) puede enviar, o recibir, datos de la memoria utilizando el bus de datos.

Si un programa se va a almacenar de forma permanente, habitualmente se coloca en un dispositivo de memoria denominado *memoria de sólo lectura* (ROM). La ROM normalmente es una pastilla de memoria, programada permanentemente, en forma de CI. La memoria temporal donde se almacenan los datos, generalmente es un dispositivo integrado denominado *memoria de lectura/escritura* (RWM). En la práctica, la memoria de lectura/escritura se denomina *memoria de acceso aleatorio,* o RAM. Los programas de usuario de la microcomputadora, que son de naturaleza temporal, también están almacenados en la sección RAM de la memoria junto a los datos. Las secciones RAM y ROM de memoria aparecen separadas en la Figura 1.2 debido a que, generalmente, son circuitos integrados separados.

El sistema de la Figura 1.2 representa la organización general de una microcomputadora. La mayoría de ellas, como mínimo, deben tener estas características. Por claridad, es costumbre omitir en los diagramas de bloque fuentes de alimentación, relojes y algunas líneas de realimentación de la unidad microprocesadora.

PROBLEMAS RESUELTOS

1.6. La CPU de una microcomputadora es un CI denominado _____ .

Solución:

La CPU de una microcomputadora, generalmente, es un CI denominado microprocesador. En algunos diseños se necesitan varios CI para realizar el trabajo de la CPU.

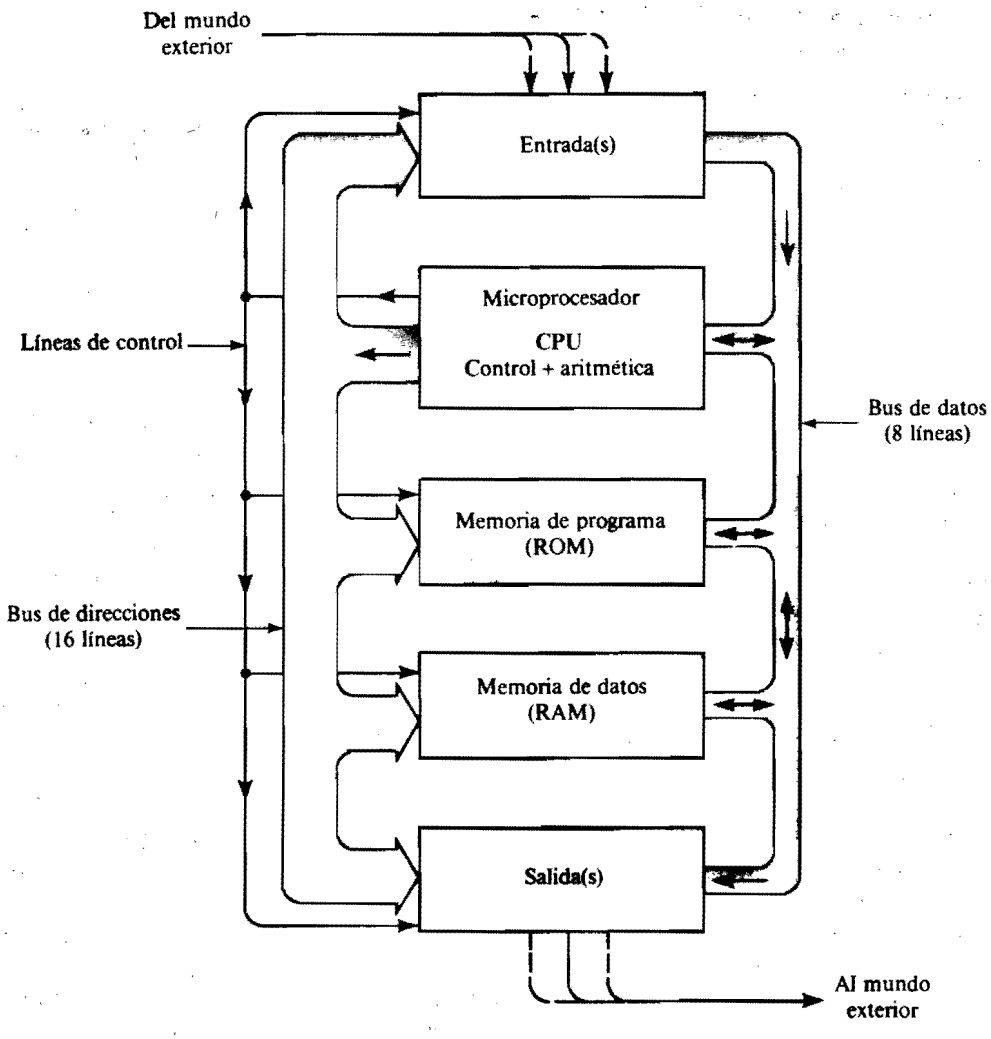


Figura 1.2. Organización típica de una microcomputadora.

1.7. Acudir a la Figura 1.2. ¿Qué bloque de esta microcomputadora puede considerarse la unidad de *control*?

Solución:

El microprocesador (también identificado como CPU en la Figura 1.2) controla las demás unidades de la microcomputadora.

1.8. Listar tres tipos de interconexiones en el sistema microcomputador de la Figura 1.2.

Solución:

Las interconexiones entre los CI de la microcomputadora de la Figura 1.2 son: el bus de direcciones, el bus de datos y las líneas de control. En realidad hay más líneas entre los CI que las mostradas en la Figura 1.2.

1.9. El bus de direcciones de la Figura 1.2 es un camino de una dirección para la información codificada, mientras que el bus de _____ es un camino de dos direcciones.

Solución:

El bus de datos es un camino de dos direcciones para la información del sistema microcomputador de la Figura 1.2.

1.10. Los programas permanentes, normalmente se almacenan en dispositivos integrados denominados memorias de _____ - _____ .

Solución:

Los programas permanentes comúnmente se almacenan en dispositivos integrados denominados memorias de sólo lectura (ROM).

1.11. Las siglas ROM ¿qué tipo de memoria de computadora referencian?

Solución:

ROM significa memoria de sólo lectura.

1.12. Los datos y los programas temporales de la microcomputadora se almacenan en memorias denominadas _____ (RAM, ROM).

Solución:

Los datos y programas temporales se almacenan en memorias denominadas RAM.

1.13. El almacenamiento de datos en la RAM de una microcomputadora es _____ (permanente, temporal).

Solución:

El almacenamiento de la RAM es temporal.

1.14. El lugar por donde la información entra o sale de una computadora se denomina _____ (puerto, zócalo).

Solución:

La información entra o sale de la computadora a través de sitios denominados puertos.

1.4. OPERACION DE LA MICROCOMPUTADORA

En la Figura 1.3 se muestra un ejemplo de como opera una microcomputadora. En el ejemplo se ilustra el siguiente procedimiento:

1. Pulsar la tecla A del teclado.
2. Almacenar la letra A en memoria.
3. Imprimir la letra A en la pantalla del monitor de tubo de rayos catódicos (CRT).

El procedimiento de entrada-almacenamiento-salida esbozado en la Figura 1.3 es una operación común de un sistema microcomputador. El hardware electrónico utilizado en un sistema como el de la Figura 1.3 es bastante complicado. Sin embargo, la transferencia de datos en el sistema ayudará a explicar el uso de las diferentes unidades de la microcomputadora.

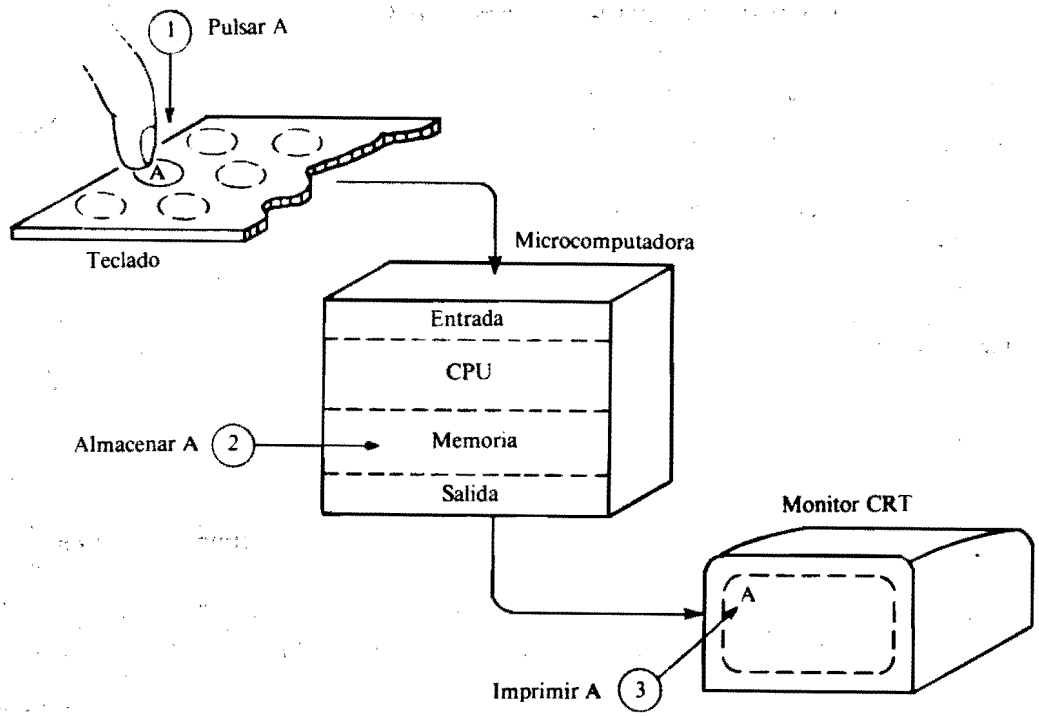


Figura 1.3. Operación típica de la microcomputadora.

Un diagrama más detallado en la Figura 1.4 ayudará a comprender el procedimiento común de entrada-almacenamiento-salida de la microcomputadora. Primero, observar cuidadosamente los *contenidos* de la memoria de programa en la Figura 1.4. Observar que las instrucciones ya se han cargado en las seis primeras posiciones de memoria. De la Figura 1.4, se desprende que las instrucciones que están en la memoria de programa son:

1. INTRODUCIR (INPUT) el dato del puerto de entrada 1.
2. ALMACENAR (STORE) el dato del puerto 1 en la posición 200 de la memoria de datos.
3. SACAR (OUTPUT) el dato al puerto de salida 10.

Observar que solamente hay tres instrucciones en el programa anterior, aunque puede parecer que hay seis en la memoria de programa de la Figura 1.4. La razón para pensar esto, es que las instrucciones se descomponen habitualmente en varias partes. La primera parte de la instrucción 1 consiste en INTRODUCIR (INPUT) un dato. La segunda parte indica de dónde proviene el dato (del puerto 1). La primera parte de *acción* de la instrucción se denomina *operación* y la segunda parte *operando*. Operación y operando están localizados en posiciones separadas de memoria en la memoria de programa de la Figura 1.4. Para la primera instrucción de la Figura 1.4, la posición 100 de memoria de programa contiene la operación de INTRODUCIR (INPUT) mientras que la posición 101 contiene el operando (puerto 1) desde donde se introducirá la información.

En el microprocesador de la Figura 1.4, se han identificado dos nuevas secciones denominadas *registros*. Estos registros especiales son el *registro acumulador* y el de *instrucción*.

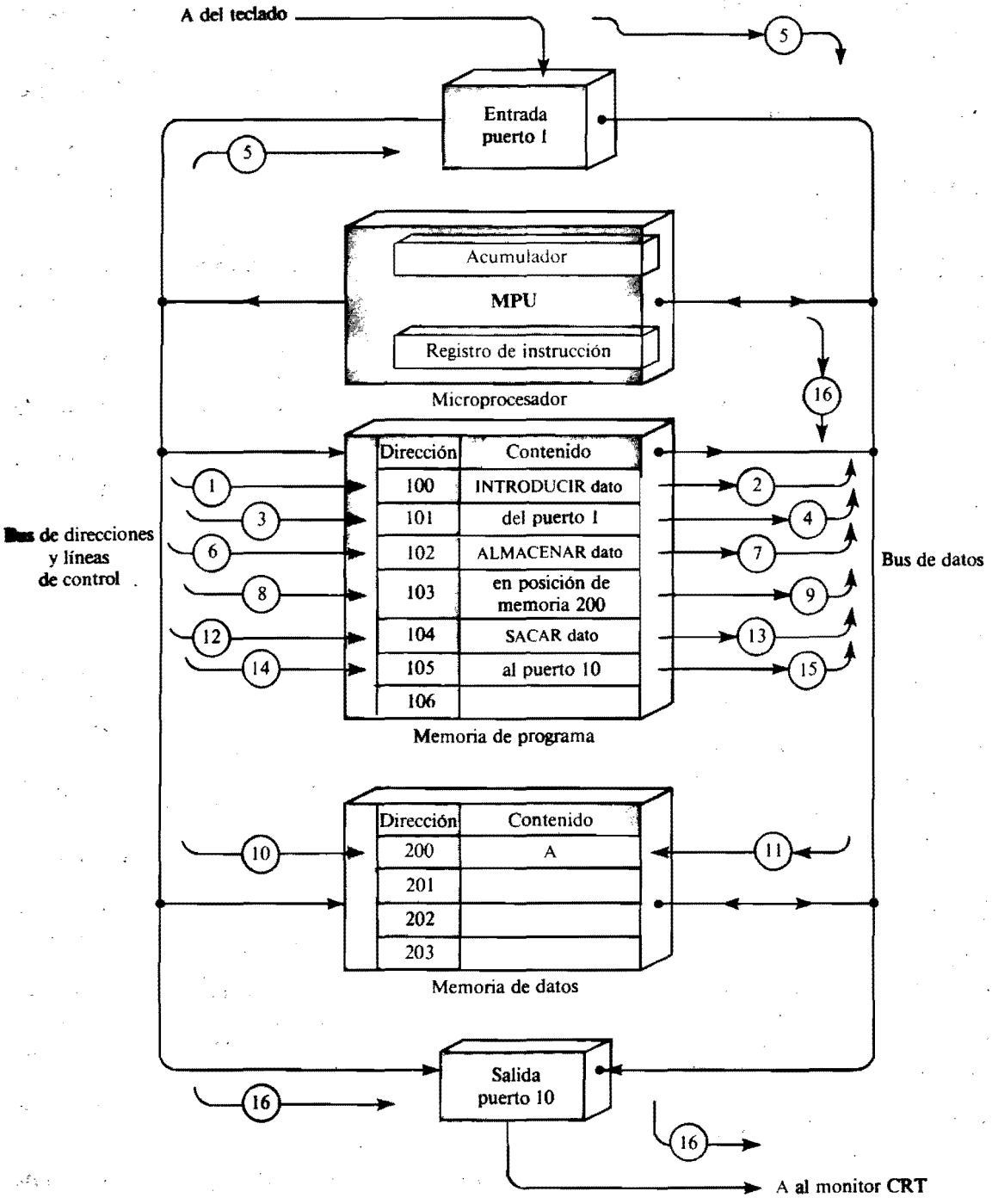


Figura 1.4. Operación paso a paso de una microcomputadora cuando ejecuta las instrucciones de la memoria de programa.

La secuencia de eventos que ocurren en la microcomputadora en el ejemplo de introducir-almacenar-sacar de la Figura 1.3 se esboza en la Figura 1.4. El flujo de instrucciones y datos puede seguirse en el diagrama por los números inscritos en los círculos. Recordar que el *microprocesador (MPU) es el centro de todas las transferencias de datos y operaciones.* En la Figura 1.4 se pueden seguir los pasos que a continuación se detallan.

Paso 1. La MPU envía la dirección 100 al bus de direcciones. Una línea de control *habilita* (conecta) la entrada de *lectura* en el CI de la memoria de programa (leer significa copiar información de una posición de memoria). Este paso se simboliza en la Figura 1.4 por un 1 inscrito en un círculo.

Paso 2. La memoria de programa anuncia la primera instrucción (INTRODUCIR dato) al bus de datos, y la MPU acepta este mensaje codificado. La instrucción se coloca en una posición de memoria especial de la MPU, denominada registro de instrucción. La MPU *decodifica* (interpreta) la instrucción y determina que necesita el operando de la instrucción INTRODUCIR dato.

Paso 3. La MPU envía la dirección 101 al bus de direcciones. La línea de control habilita la entrada de lectura de la memoria de programa.

Paso 4. La memoria de programa coloca el operando (del puerto 1) en el bus de datos. El operando estaba localizado en la dirección 101 de la memoria de programa. Este mensaje codificado (la dirección del puerto 1) se acepta en el bus de datos y se coloca en el registro de instrucción. La MPU decodifica ahora la instrucción completa (INTRODUCIR dato del puerto 1).

Paso 5. La MPU hace que se abra el puerto 1 utilizando el bus de direcciones y las líneas de control en la unidad de entrada. La forma codificada de A es transferida y almacenada en el *acumulador* de la MPU.

Es importante observar que la MPU sigue siempre una *secuencia de busca-decodifica-ejecuta.* Primero busca la instrucción en la memoria de programa; segundo, la decodifica y tercero, la ejecuta. Tratar de observar esta secuencia de busca decodifica-ejecuta en las dos instrucciones siguientes. Continuar con el programa listado en la memoria de programa de la Figura 1.4.

Paso 6. La MPU direcciona la posición 102 en el bus de direcciones; después habilita la entrada de lectura en la memoria de programa utilizando las líneas de control.

Paso 7. El código para la instrucción ALMACENAR dato es leído en el bus de datos y es aceptado por la MPU en el registro de instrucción.

Paso 8. La MPU decodifica la instrucción ALMACENAR dato y determina que necesita el operando. La MPU direcciona la siguiente posición de memoria (103) y habilita la entrada de lectura de la memoria de programa.

Paso 9. El código para «en la posición de memoria 200» es colocado en el bus de datos por la memoria de programa. La MPU acepta este operando y lo almacena en el registro de instrucción. La instrucción completa «ALMACENAR datos en la posición de memoria 200» ha sido buscada en memoria y decodificada.

Paso 10. Ahora comienza el proceso de ejecución. La MPU envía la dirección 200 al bus de direcciones y habilita la entrada de *escritura* de la memoria de datos (escribir significa copiar datos en una posición de memoria).

Paso 11. La MPU pone la información, almacenada en el acumulador, en el bus de datos (la forma codificada de A). La A se escribe en la posición 200 de la memoria de datos. La segunda instrucción ha sido ejecutada. El proceso de ALMACENAR no destruye el contenido del acumulador, éste contiene todavía la forma codificada de A.

Paso 12. La MPU debe buscar la instrucción siguiente. Direcciona la posición 104 y habilita la entrada de lectura de la memoria de programa.

Paso 13. El código de la instrucción «SACAR dato» se coloca en el bus de datos. La MPU acepta la instrucción en el registro de instrucción, la decodifica y determina que necesita un operando.

Paso 14. La MPU coloca la dirección 105 en el bus de direcciones y habilita la entrada de lectura de la memoria de programa.

Paso 15. La memoria de programa pone el código del operando «al puerto 10» en el bus de datos. La MPU acepta este código en el registro de instrucción.

Paso 16. La MPU decodifica la instrucción completa «SACAR el dato al puerto 10» y activa el puerto 10, utilizando el bus de direcciones y las líneas de control en la unidad de salida; coloca, además, el código de A (todavía almacenado en el acumulador) en el bus de datos. La A es transmitida desde el puerto 10 al monitor CRT.

La mayoría de los microprocesadores transfieren información de forma similar a la detallada en la Figura 1.4. Las variaciones más grandes están probablemente en las secciones de entrada y salida. A veces se necesitan algunos pasos más, para lograr que las secciones de entrada y salida operen adecuadamente.

Es importante observar que la MPU es el centro y control de todas las operaciones y sigue la secuencia de busca-decodifica-ejecuta. Sin embargo, las operaciones reales del microprocesador son dictadas por las instrucciones listadas en la memoria del programa.

PROBLEMAS RESUELTOS

1.15. A la lista de las instrucciones ejecutadas por una microcomputadora se denomina _____.

Solución:

A la lista de las instrucciones ejecutadas por una microcomputadora se denomina programa.

1.16. La mayoría de las instrucciones de una microcomputadora constan de dos partes denominadas: operación y _____.

Solución:

La mayoría de las instrucciones constan de dos partes denominadas operación y operando.

1.17. Las instrucciones de la microcomputadora en la memoria de programa generalmente están colocadas en orden _____ (consecutivo, aleatorio).

Solución:

Las instrucciones en la memoria de programa *generalmente* están colocadas en orden consecutivo. Más tarde, se introducirán instrucciones especiales de *bifurcación*, para hacer que la MPU bifurque a otras instrucciones no consecutivas. Nunca están colocadas en orden aleatorio.

1.18. Acudir a la Figura 1.4. Después de ejecutar la instrucción «SACAR dato al puerto 10» la MPU apuntará a la posición de dirección _____ en la memoria de programa para la siguiente instrucción.

Solución:

Después de ejecutar la instrucción «SACAR dato al puerto 10» en la Figura 1.4, la MPU apuntará a la posición de dirección 106 en la memoria del programa para la siguiente instrucción. La

posición de memoria del programa 106 está en blanco en la Figura 1.4, así que la acción de la MPU será impredecible. Probablemente habrá que colocar aquí una instrucción de ESPERAR (WAIT) o ALTO (HALT) para que pueda predecirse la operación de la computadora.

1.19. Acudir a la Figura 1.4. Una instrucción de la microcomputadora como, por ejemplo, «ALMACENAR dato en la posición de memoria 201» producirá como resultado la transferencia de un dato de la MPU a la posición 201 de la memoria de _____.

Solución:

Una instrucción como, por ejemplo, «ALMACENAR dato en la posición de memoria 201» producirá como resultado la transferencia de un dato de la MPU a la posición 201 de la memoria de datos.

1.20. Acudir a la Figura 1.4. Una instrucción de la microcomputadora como, por ejemplo, «ALMACENAR dato en la posición de memoria 202» dará como resultado la transferencia del dato del _____ (acumulador, registro de instrucción) de la MPU a la posición de memoria _____ de la RAM.

Solución:

Una instrucción como, por ejemplo, «ALMACENAR dato en la posición de memoria 202» dará como resultado la transferencia del dato del acumulador de la MPU a la posición de memoria 202 de la RAM. Este dato estará entonces en la posición de memoria 202 de la RAM y en el acumulador de la MPU. El contenido del acumulador no se destruye cuando se almacena el dato.

1.21. El proceso de _____ (lectura, escritura) tiene lugar cuando se copia un dato de una posición de memoria.

Solución:

El proceso de lectura tiene lugar cuando se copia un dato de una posición de memoria.

1.22. Colocar datos en una posición de memoria es un proceso denominado _____ (lectura, escritura) en memoria.

Solución:

Colocar datos en una posición de memoria es un proceso denominado escritura en memoria.

1.23. Para cada instrucción en la memoria de programa, la MPU va a través de una secuencia _____ - _____ - _____.

Solución:

Para cada instrucción en la memoria de programa, la MPU va a través de una secuencia de busca-decodifica-ejecuta. Algunos libros llaman a esta secuencia el *ciclo de localiza-lee-interpreta-ejecuta.*

1.24. Acudir a la Figura 1.4. Después del paso 16, cuando se completa la instrucción «SACAR dato al puerto 10», ¿qué dato contiene el acumulador de la MPU?

Solución:

El acumulador contendrá el código de A. La lectura de datos, de un registro o posición de memoria, solamente copia la información que está allí y no destruye el dato.

1.25. Acudir a la Figura 1.4. Después del paso 16, ¿existirán todavía las instrucciones en las posiciones de memoria 100 a 105?

Solución:

Sí. Las instrucciones de lectura no destruyen los contenidos de esas posiciones de memoria.

PROBLEMAS SUPLEMENTARIOS

1.26. En electrónica, las siglas CI significan _____ _____.

Res. circuito integrado.

1.27. En electrónica, las siglas LSI significan _____ _____ _____.

Res. gran escala de integración.

1.28. En informática, las siglas CPU significan _____ _____ _____.

Res. unidad central de tratamiento.

1.29. Un CI LSI que tiene la mayoría de las capacidades de la CPU de una computadora se denomina _____.

Res. microprocesador.

1.30. En microcomputadoras, las letras MPU significan _____ _____.

Res. unidad microprocesadora.

1.31. Las microcomputadoras que realizan solamente una o dos tareas se dicen que son _____ (de propósito general, dedicadas).

Res. dedicadas.

1.32. Una organización clásica de computadora ¿qué cinco unidades importantes tiene?

Res. entrada, control, aritmética, memoria, salida.

1.33. Una lista de instrucciones de computadora se considera _____ (hardware, software).

Res. software.

1.34. Las informaciones que se manipulan en una computadora se denominan _____ (datos, informaciones numéricas).

Res. datos.

1.35. Una persona que escribe listas de instrucciones para una computadora se denomina un _____.

Res. programador.

1.36. El almacenamiento permanente de un programa en una microcomputadora, probablemente, puede hacerse en una _____ (RAM, ROM).

Res. ROM.

1.37. La _____ (RAM, ROM) es un CI muy utilizado en las microcomputadoras como memoria de lectura/escritura.

Res. RAM.

1.38. Acudir a la Figura 1.2. Las instrucciones almacenadas en la memoria de programa serán _____ (permanentes, temporales).

Res. permanentes (almacenadas en ROM).

1.39. Acudir a la Figura 1.4. Después de que se completa el paso 16, la MPU intentará el procedimiento de _____ (decodificación, búsqueda, ejecución).

Res. búsqueda.

1.40. Acudir a la Figura 1.4. La MPU accede a una posición específica de memoria al utilizar el bus de _____ .

Res. direcciones.

1.41. Acudir a la Figura 1.4. La información codificada es transferida desde el acumulador de la MPU a una posición de memoria, en la memoria de datos, utilizando el bus de _____ .

Res. datos.

1.42. Si la MPU busca y decodifica una instrucción como «ALMACENAR datos en la posición de memoria 205», el dato provendrá del _____ .

Res. acumulador (en la MPU).

1.43. Una microcomputadora contiene como mínimo una unidad de entrada, una unidad de salida, una MPU y una _____ de programas y datos.

Res. memoria.

1.44. Una memoria temporal de lectura/escritura se designa por las tres siguientes letras RWN o _____ .

Res. RAM.

Capítulo 2

NUMEROS, CODIGOS DE COMPUTADORA Y ARITMETICA

2.1. NUMEROS BINARIOS

Las computadoras digitales utilizan *números binarios*. El sistema de numeración binario, o de *base 2*, utiliza solamente los dígitos 0 y 1; los dígitos binarios se llaman *bits*. En los circuitos electrónicos de las computadoras el bit 0 habitualmente se representa por una tensión BAJA, mientras que el bit 1 corresponde a una tensión ALTA.

Las personas están acostumbradas a comprender el *sistema de numeración decimal*, o de *base 10*, que tiene 10 dígitos (0-9). Este sistema también tiene la característica de *valor por posición;* por ejemplo, la Figura 2.1*a* muestra que el número decimal 1327 es igual a 1000 más tres 100 más dos 10 más siete 1 (1000 + 300 + 20 + 7 = 1327).

El sistema de numeración binario también tiene la característica de valor por posición. El valor decimal de las cuatro primeras posiciones binarias se muestra en la Figura 2.1*b*. El número binario 1001 (se pronuncia uno, cero, cero, uno) se convierte a su equivalente decimal de 9. El bit del 1 del número binario de la Figura 2.1*b* se denomina *bit menos significativo* (LSB), mientras que el bit del 8 se denomina *bit más significativo* (MSB).

Los equivalentes binarios de los números decimales entre 0 y 15 se muestran en la Figura 2.1*c*. Las personas que trabajan con computadoras memorizan como mínimo estos números binarios.

Convertir el número binario 10110110 (se pronuncia uno, cero, uno, uno, cero, uno, uno, cero) a su equivalente decimal. El procedimiento se muestra en la Figura 2.2*a*. Por cada bit 1 del número binario se escribe debajo el valor de la posición decimal y después se suman los decimales (128 + 32 + 16 + 4 + 2 = 182), dando 182. Los pequeños *subíndices* de la Figura 2.2*b* se utilizan para anotar la base (a veces denominada raíz) del número. El número 10110110_2 es por tanto un número binario, o en base 2 y el número 182_{10} es un número decimal o en base 10.

Convertir el número decimal 155 a binario. La Figura 2.3 muestra un procedimiento para hacer esta conversión. El número decimal 155 se divide primero por 2, dando un cociente de 77 y un resto de 1; el resto se convierte en el bit menos significativo (LSB) del número binario y se transfiere a esta posición en la Figura 2.3. El cociente (77) se transfiere como muestra la flecha y se convierte en el siguiente dividendo. Los cocientes se dividen repetidamente por 2 hasta que el cociente se hace 0 con un resto de 1. La Figura 2.3 muestra este procedimiento. La línea inferior muestra el resultado de la conversión: $155_{10} = 10011011_2$.

Potencias de 10	10^3	10^2	10^1	10^0
Valor de posición	1000	100	10	1
Decimal	1	3	2	7
Decimal	1000 +	300 +	20 +	7 = 1327

(a) Valores de la posición en un número decimal

Potencias de 2	2^3	2^2	2^1	2^0
Valor de posición	8	4	2	1
	MSB			LSB
Binario	1	0	0	1
Decimal	8 +	0 +	0 +	1 = 9

(b) Valores de la posición en un número binario

Decimal	Binario				Decimal	Binario			
10 1	8	4	2	1	10 1	8	4	2	1
0				0	8	1	0	0	0
1				1	9	1	0	0	1
2			1	0	1 0	1	0	1	0
3			1	1	1 1	1	0	1	1
4		1	0	0	1 2	1	1	0	0
5		1	0	1	1 3	1	1	0	1
6		1	1	0	1 4	1	1	1	0
7		1	1	1	1 5	1	1	1	1

(c) Equivalentes decimales y binarios

Figura 2.1.

Potencias de 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Valor de posición	128	64	32	16	8	4	2	1
Binario	1	0	1	1	0	1	1	0
Decimal	128	+	32 +	16	+	4	+ 2	= 182

(a) Conversión de binario a decimal

$$10110110_2 = 182_{10}$$

(b) Los índices designan la base del número

Figura 2.2.

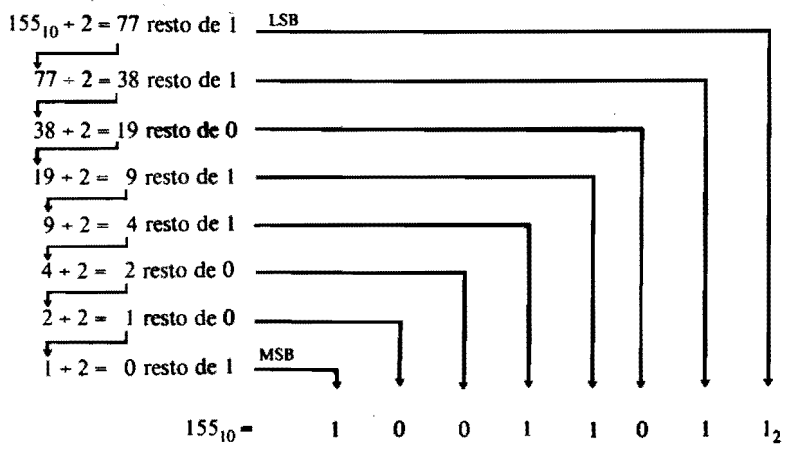


Figura 2.3.

PROBLEMAS RESUELTOS

2.1. La mayoría de las personas comprenden el sistema de numeración decimal, mientras que las computadoras digitales utilizan el sistema de numeración de base 2, o _____ .

Solución:

Las computadoras digitales utilizan el sistema de numeración de base 2, o binario, y contiene solamente los dígitos 0 y 1.

2.2. El término *bit* significa _____ _____ cuando se trata de números binarios.

Solución:

Bit significa dígito binario.

2.3. El número 100_{10} es un número en base _____ .

Solución:

El número 100_{10} es un número en base 10, como indica el subíndice 10 al final del número.

2.4. Escribir en base 2 el número uno, uno, cero, cero utilizando un subíndice.

Solución:

1100_2.

2.5. Las letras MSB significan _____ _____ _____ cuando se trata con números binarios.

Solución:

Las letras MSB significan bit más significativo. Si el número es 1000_2, el 1 es el MSB.

2.6. De memoria, convertir los siguientes números binarios en sus equivalentes decimales:
(*a*) 0001 (*b*) 0101 (*c*) 1000 (*d*) 1011 (*e*) 1111 (*f*) 0111

Solución:

Acudir a la Figura 2.1*c*. Estos números binarios han de ser memorizados. Los equivalentes decimales para los números binarios son los siguientes:

(*a*) $0001_2 = 1_{10}$ (*c*) $1000_2 = 8_{10}$ (*e*) $1111_2 = 15_{10}$
(*b*) $0101_2 = 5_{10}$ (*d*) $1011_2 = 11_{10}$ (*f*) $0111_2 = 7_{10}$

2.7. Convertir los siguientes números binarios en sus equivalentes decimales:
(*a*) 10000000 (*c*) 00110011 (*e*) 00011111
(*b*) 00010000 (*d*) 01100100 (*f*) 11111111

Solución:

Siguiendo el procedimiento mostrado en la Figura 2.1*b*, los equivalentes decimales de los números binarios son los que se indican:

(*a*) $10000000_2 = 128_{10}$ (*c*) $00110011_2 = 51_{10}$ (*e*) $00011111_2 = 31_{10}$
(*b*) $00010000_2 = 16_{10}$ (*d*) $01100100_2 = 100_{10}$ (*f*) $11111111_2 = 255_{10}$

2.8. Convertir los siguientes números decimales en sus equivalentes binarios:
(*a*) 39 (*b*) 48

Solución:

Seguir el procedimiento mostrado en la Figura 2.3. Los equivalentes binarios de los números decimales son los siguientes:

(*a*)
$39_{10} \div 2 = 19$ resto de 1 (LSB)
$19 \div 2 = 9$ resto de 1
$9 \div 2 = 4$ resto de 1
$4 \div 2 = 2$ resto de 0
$2 \div 2 = 1$ resto de 0
$1 \div 2 = 0$ resto de 1 (MSB)
$39_{10} = 100111_2$

(*b*)
$48_{10} \div 2 = 24$ resto de 0 (LSB)
$24 \div 2 = 12$ resto de 0
$12 \div 2 = 6$ resto de 0
$6 \div 2 = 3$ resto de 0
$3 \div 2 = 1$ resto de 1
$1 \div 2 = 0$ resto de 1 (MSB)
$48_{10} = 110000_2$

2.2. NUMEROS HEXADECIMALES

Una posición de la memoria de una microcomputadora puede contener el número binario 10011110. Esta larga cadena de ceros y unos es difícil de recordar y teclear. El número 10011110_2 puede convertirse en un número decimal. Una vez convertido éste es el número 158_{10}. Este proceso de conversión es demasiado largo. La mayoría de los sistemas de microcomputadoras utilizan la *notación hexadecimal* para simplificar la tarea de recordar y teclear números binarios como por ejemplo 10011110.

El sistema de numeración hexadecimal, o de base 16, utiliza los 16 símbolos del 0 al 9, A, B, C, D, E y F. Los equivalentes binarios, hexadecimales y decimales se muestran en la Figura 2.4.

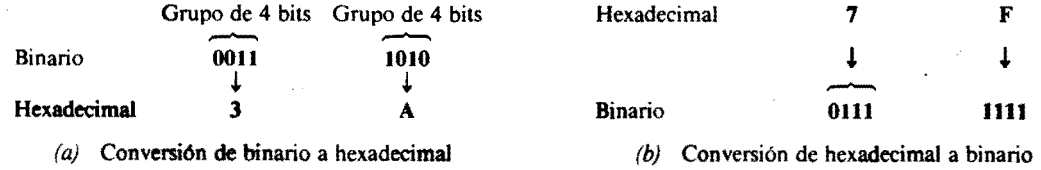
Decimal	Hexadecimal	Binario			
		8	4	2	1
0	0	0	0	0	0
1	1	0	0	0	1
2	2	0	0	1	0
3	3	0	0	1	1
4	4	0	1	0	0
5	5	0	1	0	1
6	6	0	1	1	0
7	7	0	1	1	1
8	8	1	0	0	0
9	9	1	0	0	1
10	A	1	0	1	0
11	B	1	0	1	1
12	C	1	1	0	0
13	D	1	1	0	1
14	E	1	1	1	0
15	F	1	1	1	1

Figura 2.4. Contar en decimal, hexadecimal y binario.

Observar en la Figura 2.4 que cada símbolo hexadecimal representa una única combinación de 4 bits. El número binario 10011110 puede entonces ser representado como 9E en hexadecimal. Esto es, la parte 1001 del número binario es igual a 9, de acuerdo con la Figura 2.4, y la parte 1110 del número binario es igual a E en hexadecimal. Por tanto 10011110_2 es igual a $9E_{16}$. Recordar que el subíndice indica la base del número.

Convertir el número binario 111010 en hexadecimal (hex). Comenzar por el LSB y dividir el número binario en grupos de 4 bits cada uno, como indica la Figura 2.5*a*. Entonces sustituir cada grupo de 4 bits por su dígito hex equivalente. El 1010_2 es igual a A en hex (ver Fig. 2.4). El 0011_2 es igual a 3 en hex. Por tanto 111010_2 es igual a $3A_{16}$.

Convertir el número hexadecimal 7F a su equivalente binario. La Figura 2.5*b* muestra que cada dígito hex es sustituido por su equivalente binario de 4 bits. En este ejemplo, el binario 0111 es sustituido por el hex 7 y 1111_2 sustituye a F_{16}. Por tanto $7F_{16}$ es igual a 1111111_2.



(a) Conversión de binario a hexadecimal *(b)* Conversión de hexadecimal a binario

Figura 2.5.

La notación hexadecimal es muy utilizada para representar números binarios. Las personas que utilizan la notación hexadecimal deben memorizar la tabla mostrada en la Figura 2.4.

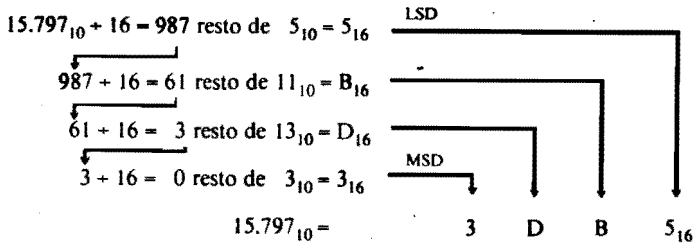
Convertir el número hexadecimal 2C6E a su equivalente decimal. El procedimiento se muestra en la Figura 2.6*a*. Los valores de posición para los cuatro primeros dígitos decimales son 4096, 256, 16 y 1. El número hexadecimal contiene catorce (E_{16}) 1, seis 16, doce (C_{16}) 256 y dos 4096. Cada valor de posición se multiplica y los productos se suman para obtener 11.374_{10}.

Convertir el número decimal 15.797 a su equivalente hexadecimal. El procedimiento se muestra en la Figura 2.6*b*. La primera línea indica 15.797_{10} dividido por 16, dando un cociente de 987_{10} con un resto de 5_{10}. El resto se convierte entonces a su equivalente hexadecimal. Por tanto 5_{10} es igual a 5_{16}. El resto hexadecimal (5_{16}) se convierte en el dígito menos significativo (LSD) del número hexadecimal. El primer cociente (987) es el dividendo en la segunda línea y se divide por 16. El segundo cociente es 61 con un resto de 11_{10}, o B hexadecimal. La línea 3 muestra 61 dividido por 16, dando un cociente de 3 con un resto de 13_{10}, o D_{16}. La cuarta línea de la Figura 2.6*b* muestra el dividendo (3) dividido por 16, dando un cociente de 0 con un resto de 3_{10}, ó 3_{16}. Cuando el cociente se hace 0, como en la línea 4, se termina el cálculo. El 3_{16} es el dígito más significativo (MSD). El procedimiento mostrado en la Figura 2.6*b* convierte el número decimal 15.797 en su equivalente hex de $3DB5_{16}$.

Potencias de 16	16^3	16^2	16^1	16^0
Valor de posición	4096	256	16	1

Hexadecimal	**2**		**C**		**6**		**E**		
	↓		↓		↓		↓		
	4096		256		16		1		
	×2		×12		×6		×14		
Decimal	8192	+	3072	+	96	+	14	=	11.374_{10}

(a) Conversión de hexadecimal a decimal



(b) Conversión de decimal a hexadecimal

Figura 2.6.

PROBLEMAS RESUELTOS

2.9. La notación hexadecimal es muy utilizada, para trabajar con las microcomputadoras, como método «abreviado» de representar números _____ (binarios, decimales).

Solución:

La notación hexadecimal es muy utilizada para representar números binarios.

2.10. El sistema de numeración hexadecimal a veces se denomina sistema de base _____.

Solución:

El sistema de numeración hexadecimal a veces se denomina sistema de base 16 debido al uso de 16 símbolos únicos.

2.11. Convertir los siguientes números hexadecimales a sus equivalentes binarios:

(*a*) C (*c*) F (*e*) 1A
(*b*) 6 (*d*) E2 (*f*) 3D

Solución:

Utilizando la tabla de la Figura 2.4, seguir el procedimiento de la Figura 2.5*b*. Los equivalentes binarios para los números hexadecimales son los siguientes:

(*a*) $C_{16} = 1100_2$ (*c*) $F_{16} = 1111_2$ (*e*) $1A_{16} = 00011010_2$
(*b*) $6_{16} = 0110_2$ (*d*) $E2_{16} = 11100010_2$ (*f*) $3D_{16} = 00111101_2$

2.12. Convertir los siguientes números binarios en sus equivalentes hexadecimales:

(*a*) 1001 (*c*) 1101 (*e*) 10000000
(*b*) 1100 (*d*) 1111 (*f*) 01111110

Solución:

Utilizando la tabla de la Figura 2.4, seguir el procedimiento de la Figura 2.5*a*. Los equivalentes hexadecimales para los números binarios son los siguientes:

(*a*) $1001_2 = 9_{16}$ (*c*) $1101_2 = D_{16}$ (*e*) $10000000_2 = 80_{16}$
(*b*) $1100_2 = C_{16}$ (*d*) $1111_2 = F_{16}$ (*f*) $01111110_2 = 7E_{16}$

2.13. Convertir los siguientes números hexadecimales en sus equivalentes decimales:

(*a*) 7E (*b*) DB (*c*) 12A3 (*d*) 34CF

Solución:

Utilizando la tabla de la Figura 2.4, seguir el procedimiento de la Figura 2.6*a*. Los equivalentes decimales a los números hexadecimales son los siguientes:

(*a*) $7E_{16} = (16 \times 7) + (1 \times 14) = 126_{10}$
(*b*) $DB_{16} = (16 \times 13) + (1 \times 11) = 219_{10}$
(*c*) $12A3_{16} = (4096 \times 1) + (256 \times 2) + (16 \times 10) + (1 \times 3) = 4771_{10}$
(*d*) $34CF_{16} = (4096 \times 3) + (256 \times 4) + (16 \times 12) + (1 \times 15) = 13.519_{10}$

2.14. $48.373_{10} =$ _____ $_{16}$.

Solución:

Utilizando la tabla de la Figura 2.4, seguir el procedimiento de la Figura 2.6*b*.

$48.373_{10} \div 16 = 3023$ resto de $5_{10} = 5_{16}$ (LSD)
$3023 \div 16 = 188$ resto de $15_{10} = F_{16}$
$188 \div 16 = 11$ resto de $12_{10} = C_{16}$
$11 \div 16 = 0$ resto de $11_{10} = B_{16}$ (MSD)

$$48.373_{10} = BCF5_{16}$$

2.3. NUMEROS BCD

Los números binarios puros se representan en notación hexadecimal para hacer más fácil la conversión. Sin embargo, la conversión binario a decimal es bastante difícil. En calculadoras, juegos e instrumentos digitales, donde son frecuentes las entradas y salidas del usuario en decimal, se utiliza un código especial para representar los números decimales. Este código se denomina *BCD (decimal-codificado-binario)*. Las equivalencias entre decimal y BCD se muestran en la tabla de la Figura 2.7*a*. Técnicamente, esta tabla detalla el *código BCD 8421*. La parte del nombre 8421 da el valor de la posición a los 4 bits del código BCD. También se utilizan otros códigos BCD, como por ejemplo el código BCD 5421 y el código de exceso 3.

Decimal	BCD			
	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

(a) Código BCD 8421

Decimal	3	6	9	1
	↓	↓	↓	↓
BCD	0011	0110	1001	0001

(b) Conversión de decimal a BCD

BCD	1000	0000	0111	0010
	↓	↓	↓	↓
Decimal	8	0	7	2

(c) Conversión de BCD a decimal

Figura 2.7.

Convertir el número decimal 3691 a su equivalente BCD 8421. El procedimiento se muestra en la Figura 2.7*b*. Cada dígito decimal se traduce directamente a su equivalente BCD de 4 bits. Este ejemplo muestra que 3691_{10} es igual a $0011\ 0110\ 1001\ 0001_{BCD}$.

Convertir el número BCD 1000 0000 0111 0010 a su equivalente decimal. El procedimiento se detalla en la Figura 2.7*c*. Cada grupo de 4 bits se traduce directamente a su equivalente decimal. Este ejemplo muestra que $1000\ 0000\ 0111\ 0010_{BCD}$ es igual a 8072_{10}.

Los microprocesadores suman los números binarios puros. Sin embargo, muchos microprocesadores tienen instrucciones especiales para cambiar el resultado de las sumas a notación BCD. El número BCD se interpreta fácilmente entonces como número decimal utilizando los sencillos procedimientos mostrados en las Figuras 2.7*b* y *c*.

PROBLEMAS RESUELTOS

2.15. Las letras BCD significan _____ _____ _____ .

Solución:

Las letras BCD significan decimal codificado binario.

2.16. La notación BCD más común es el código _____ (5421, 8421).

Solución:

La notación BCD más común es el código BCD 8421.

2.17. Convertir los siguientes números decimales en sus equivalentes 8421 BCD:

(*a*) 39 (*c*) 40 (*e*) 82
(*b*) 65 (*d*) 17 (*f*) 99

Solución:

Seguir el procedimiento de la Figura 2.7*b*. Los equivalentes BCD para los números decimales son los siguientes:

(*a*) $39_{10} = 0011\ 1001_{BCD}$ (*c*) $40_{10} = 0100\ 0000_{BCD}$ (*e*) $82_{10} = 1000\ 0010_{BCD}$
(*b*) $65_{10} = 0110\ 0101_{BCD}$ (*d*) $17_{10} = 0001\ 0111_{BCD}$ (*f*) $99_{10} = 1001\ 1001_{BCD}$

2.18. Convertir los siguientes números BCD 8421 en sus equivalentes decimales:

(*a*) 1000 0000 (*c*) 1001 0010 (*e*) 0100 0011
(*b*) 0000 0001 (*d*) 0111 0110 (*f*) 0101 0101

Solución:

Seguir el procedimiento mostrado en la Figura 2.7*c*. Los equivalentes decimales para los números BCD son los siguientes:

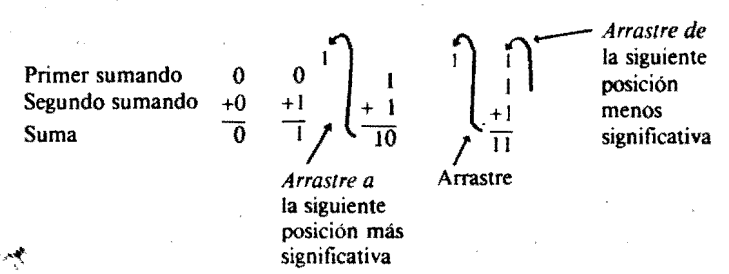
(*a*) $1000\ 0000_{BCD} = 80_{10}$ (*c*) $1001\ 0010_{BCD} = 92_{10}$ (*e*) $0100\ 0011_{BCD} = 43_{10}$
(*b*) $0000\ 0001_{BCD} = 1_{10}$ (*d*) $0111\ 0110_{BCD} = 76_{10}$ (*f*) $0101\ 0101_{BCD} = 55_{10}$

2.4. ARITMETICA BINARIA

Sumar, restar o multiplicar números binarios se realiza de forma similar a la aritmética decimal. La mayoría de los microprocesadores tienen instrucciones para sumar y restar números binarios. Los microprocesadores más avanzados tienen incluso instrucciones para multiplicar y dividir, por ejemplo, los 8086, 8088, 80286, 80386 y 68000.

Las sencillas reglas para la suma binaria se muestran en la Figura 2.8*a*. Las dos primeras reglas de la parte izquierda son obvias. La tercera regla 1 + 1 = 10 muestra que el bit más significativo es *arrastrado* a la siguiente posición de orden superior. La cuarta regla muestra que en binario 1 + 1 + 1 = 11. Aquí los sumandos y el arrastre son todos unos. El resultado es una suma de 1 con un arrastre de 1.

Sumar los números binarios 00111011 y 00101010. Este problema se ilustra en la Figura 2.8*b*. Comprobar este procedimiento. Los equivalentes decimales de los números binarios se muestran, por conveniencia, a la derecha. La suma de 00111011 y 00101010 se muestra en la Figura 2.8*b* y es 01100101_2.



(a) Reglas para la suma binaria

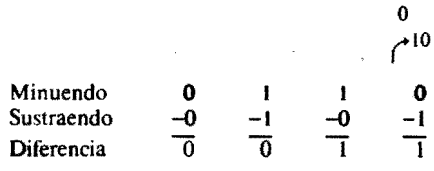
Arrastres

	1 1 1 1	1 1
Primer sumando	00 1 1 10 11	59
Segundo sumando	+ 00 1 0 10 10	+ 42
Suma	01 1 00 1 01$_2$	101_{10}

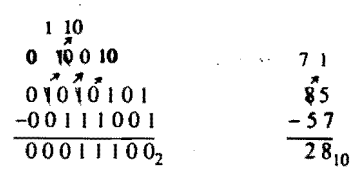
(b) Problema de suma binaria

Figura 2.8.

Las reglas para la resta binaria se muestran en la Figura 2.9*a*. Las tres primeras reglas son iguales que en la resta decimal. La última regla requiere un *préstamo* de la siguiente posición más significativa (la posición del 2). Con el préstamo, el minuendo se convierte en el binario 10, como el sustraendo es 1 la diferencia es 1.



(a) Reglas para la sustracción binaria



(b) Problema de sustracción binaria

Figura 2.9.

Restar el número binario 00111001 de 01010101. Este problema ejemplo se detalla en la Figura 2.9*b*. Las columnas del 4, 2 y 1 del problema de resta binaria son bastante sencillas de seguir utilizando las tres primeras reglas de la Figura 2.9*a*. La columna del 8 muestra un 1 restándose de un 0. Se toma prestado un 1 de la columna del 16 y se resta 1 de 10_2, obteniéndose una diferencia de 1 de acuerdo con la cuarta regla de la Figura 2.9*a*. Después del préstamo, la columna del 16 muestra un 1 restado del nuevo minuendo 0. Según la regla 4, hay que tomar prestado un 1 de la siguiente posición más significativa (la posición del 32). El minuendo de la posición 32 es 0, por lo que entonces hay que tomar prestado el 1 de la posición 64. La posición del 32 toma un préstamo de la posición del 64. Finalmente, la posición del 16 puede tomar prestado de la posición del 32. El minuendo de la posición del 16 es entonces 10_2, y el sustraendo es 1 dando una diferencia de 1. La posición del 32 ahora muestra 1 − 1 dando una diferencia de 0. La posición del 64 muestra 0 − 0 dando una diferencia de 0. La posición del 128 muestra 0 − 0 dando también una diferencia de 0. En resumen, el problema ejemplo de la Figura 2.9*b* muestra que el binario 00111001 restado de 01010101_2 da una diferencia de 00011100_2. El problema también se muestra en forma decimal a la derecha.

Las reglas para la multiplicación binaria se muestran en la Figura 2.10*a*. Las dos primeras reglas no necesitan explicación. El multiplicador es 1 en las dos últimas reglas. Cuando el multiplicador es 1 en la multiplicación binaria, el *multiplicando se copia* como producto. Cuando el multiplicador es 0, el producto siempre es 0.

Multiplicando	0	1	0	1
Multiplicador	×0	×0	×1	×1
Producto	0	0	0	1

(a) Reglas para la multiplicación binaria

Multiplicando	1101	13
Multiplicador	× 101	× 5
Primer producto parcial	1101	65_{10}
Segundo producto parcial	0000	
Tercer producto parcial	1101	
Producto final	1000001_2	

(b) Problema de multiplicación binaria

Figura 2.10.

Multiplicar los números binarios 1101 y 101. Este problema ejemplo se muestra en la Figura 2.10*b*. Como en la multiplicación decimal, el multiplicando se multiplica primero por el dígito menos significativo (en este caso el bit del 1). El bit del 1 del multiplicador es 1, por tanto el multiplicando se copia como primer producto parcial. El bit del 2 del multiplicador es un 0, por tanto el segundo producto parcial es 0000. Observar que éste se *desplaza* una posición a la izquierda. El bit del 4 del multiplicador es 1, por tanto el multiplicando se copia como tercer producto parcial. Observar que 1101 se copia después del segundo desplazamiento a la izquierda. Los productos parciales primero, segundo y tercero se suman, dando el producto final de 1000001_2. En resumen, la Figura 2.10*b* muestra que $1101_2 \times 101_2 = 1000001_2$ o que $13_{10} \times 5_{10} = 65_{10}$.

PROBLEMAS RESUELTOS

2.19. Resolver los siguientes problemas de suma binaria:

(*a*) 1010 + 0101 (*b*) 1101 + 0101 (*c*) 01011011 + 00001111 (*d*) 00111111 + 00011111

Solución:

Acudir a la Figura 2.8. Las sumas binarias de los problemas son las siguientes:

(*a*) 1111 (*b*) 10010 (*c*) 01101010 (*d*) 01011110

2.20. Resolver los siguientes problemas de resta binaria:

(*a*) 1110 − 1000 (*b*) 1010 − 0101 (*c*) 01100110 − 00011010 (*d*) 01111000 − 00111111

Solución:

Acudir a la Figura 2.9. Las diferencias binarias de los problemas son las siguientes:

(*a*) 0110 (*b*) 0101 (*c*) 01001100 (*d*) 00111001

2.21. En un problema de multiplicación el número de arriba se denomina _____ mientras que el de abajo se denomina multiplicador y el resultado se denomina _____.

Solución:

En un problema de multiplicación el número de arriba se llama multiplicando, mientras que el de abajo se llama multiplicador y el resultado se llama producto.

2.22. Resolver los siguientes problemas de multiplicación binaria:

(*a*) 1001 ×11 (*b*) 1101 ×1001 (*c*) 1111 ×101 (*d*) 1110 ×1110

Solución:

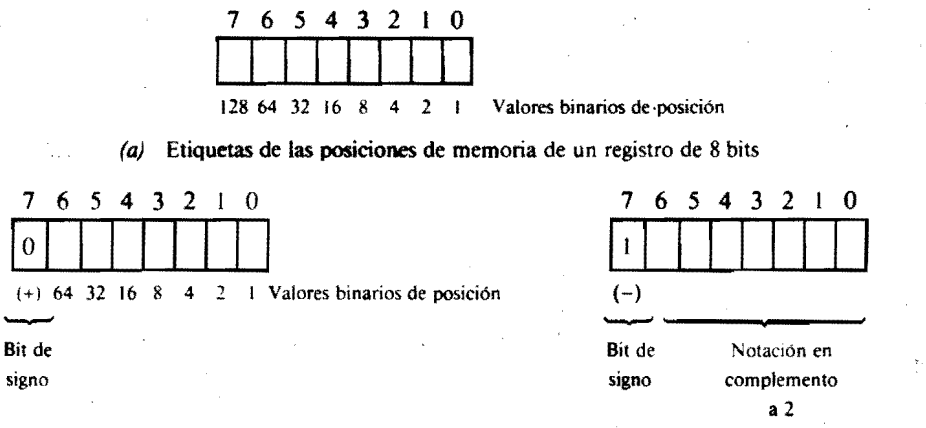
Acudir a la Figura 2.10. Los productos de los problemas son los siguientes:

(*a*) 11011 (*b*) 1110101 (*c*) 1001011 (*d*) 11000100

2.5. NOTACION EN COMPLEMENTO A 2

Generalmente, en las computadoras se utilizan los números binarios. Sin embargo, a veces se utiliza un código especial denominado *notación en complemento a 2* cuando se necesitan *números con signo.* Este sistema simplifica la circuitería de la computadora.

Un registro o posición de almacenamiento en un microprocesador puede ser como el de la Figura 2.11*a*. Este registro tiene espacio para datos de 8 bits. Las posiciones de los bits se numeran del 7 al 0. Los valores de las posiciones binarias se muestran en la parte inferior del registro. El bit 7 será el de la posición del 128, el bit 6 el de la posición del 64, etc.



(*a*) Etiquetas de las posiciones de memoria de un registro de 8 bits

(*b*) Los números positivos se identifican por un 0 en la posición del bit del signo del registro

(*c*) Los números negativos se identifican por un 1 en la posición del bit de signo del registro

Figura 2.11.

La organización más frecuente de un registro de 8 bits utilizado para almacenar *números con signo* se muestra en las Figuras 2.11*b* y *c*. El bit 7 en ambos registros es el *bit de signo.* Este bit dice si el número es (+) positivo o (−) negativo. Un 0 en la posición del bit de signo significa que el número es positivo, mientras que un 1 indica que el número es negativo.

Si el número con signo es positivo como en la Figura 2.11*b*, las restantes posiciones de memoria (6-0) contienen un número binario de 7 bits. Por ejemplo, si el contenido del registro de la Figura 2.11*b* fuese 01000001, significaría el decimal +65 (bit de signo positivo + 64 + 1). Si el contenido del registro de la Figura 2.11*b* fuese 01111111, sería $+127_{10}$ (bit de signo positi-

Decimal	Representación de números con signo	
+127	0111 1111	Números positivos representados igual que en binario puro
⋮	⋮	
+8	0000 1000	
+7	0000 0111	
+6	0000 0110	
+5	0000 0101	
+4	0000 0100	
+3	0000 0011	
+2	0000 0010	
+1	0000 0001	
+0	0000 0000	
−1	1111 1111	Números negativos representados en forma de complemento a 2
−2	1111 1110	
−3	1111 1101	
−4	1111 1100	
−5	1111 1011	
−6	1111 1010	
−7	1111 1001	
−8	1111 1000	
⋮	⋮	
−128	1000 0000	

Figura 2.12. Números decimales con signo y sus equivalentes en la notación de complemento a 2.

vo + 64 + 32 + 16 + 8 + 4 + 2 + 1). Este es el mayor número positivo que puede ser representado en este registro de 8 bits.

Si el número con signo es *negativo* como en la Figura 2.11*c*, el registro contendrá la *forma en complemento a 2* de ese número. La tabla de la Figura 2.12 muestra la notación en complemento a 2 para números positivos y negativos. Observar que los números positivos tienen un 0 en el MSB, mientras que el resto de los números corresponden a un número binario. Los números negativos tienen un 1 en el MSB. Observar la línea +0 de la tabla de la Figura 2.12. La notación en complemento a 2 para +0 es 00000000. En la línea siguiente, observar que 11111111 es la notación en complemento a 2 de −1. Imaginar que el sistema es un odómetro que cuenta hacia atrás a medida que se avanza de 00000000 a 11111111.

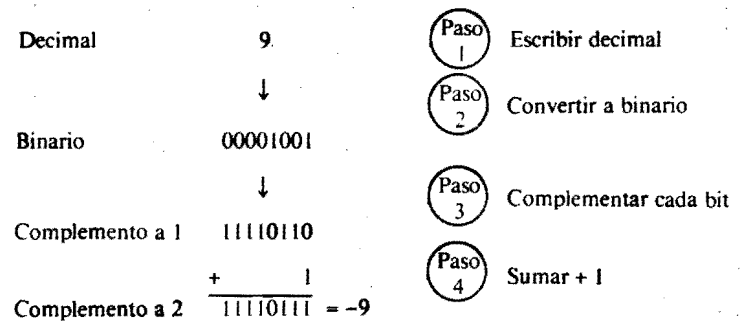
¿Cuál sería la notación en complemento a 2 de −9? Los pasos para hacer esta conversión se esbozan en la Figura 2.13*a* y son los siguientes:

Paso 1. Listar el número decimal sin signo. Escribir 9 en este ejemplo.

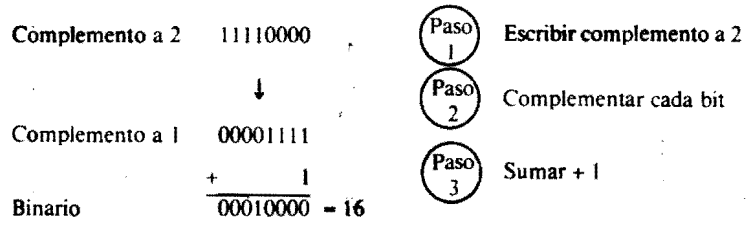
Paso 2. Convertir el número decimal a binario. Escribir el número binario 00001001 en este ejemplo.

Paso 3. Complementar cada bit formando el *complemento a 1*. En este ejemplo escribir 11110110 como complemento a 1.

Paso 4. Sumar 1 al número en complemento a 1. En este ejemplo sumar 1 a 11110110.



(a) Formación del complemento a 2 de un número negativo



(b) Cálculo del decimal equivalente para un número en complemento a 2

Figura 2.13.

El resultado es la notación en complemento a 2 para el número decimal negativo. En este ejemplo, en la Figura 2.13*a*, −9 es igual a 11110111 en forma de complemento a 2. Observar que el bit de signo (11110111) es 1, lo que significa que se trata de un número negativo.

¿Cuál es el equivalente decimal del número en complemento a 2 11110000? El procedimiento para hacer esta conversión se detalla en la Figura 2.13*b*. El procedimiento de complementar y sumar 1 es el mismo que se utiliza para convertir de binario a complemento a 2. El procedimiento de la Figura 2.13*b* muestra como se cambia cada bit del número en complemento a 2, formando el complemento a 1. Entonces se suma un 1 al número en complemento a 1, formando el número binario 00010000, que es igual a 16. Esto significa que la notación en complemento a 2 de 11110000 es igual a -16_{10}. El 16 debe ser negativo porque el bit de signo (MSB) del número en complemento a 2 es un 1.

PROBLEMAS RESUELTOS

2.23. Cuando los números con signo se almacenan en un registro de 8 bits de un microprocesador, el MSB (bit 7) se denomina bit de _____ .

Solución:

Cuando los números con signo se almacenan en un registro de 8 bits, el MSB (bit 7) se denomina bit de signo.

2.24. Determinar si los siguientes números en notación de complemento a 2 son positivos o negativos.

(*a*) 01110000 (*b*) 11001111 (*c*) 10001111 (*d*) 01010101

Solución:

(*a*) El complemento a 2 de 01110000 es un número positivo porque el bit de signo vale 0.
(*b*) El complemento a 2 de 11001111 es un número negativo porque el bit de signo vale 1.
(*c*) El complemento a 2 de 10001111 es un número negativo porque el bit de signo vale 1.
(*d*) El complemento a 2 de 01010101 es un número positivo porque el bit de signo es igual a 0.

2.25. Utilizando la tabla de la Figura 2.12, dar la notación en complemento a 2 de los siguientes números decimales con signo:

(*a*) + 1 (*b*) + 5 (*c*) + 127 (*d*) − 1 (*e*) − 2 (*f*) − 128

Solución:

Acudir a la Figura 2.12.

(*a*) + 1 = 00000001 (notación en complemento a 2)
(*b*) + 5 = 00000101 (notación en complemento a 2)
(*c*) + 127 = 01111111 (notación en complemento a 2)
(*d*) − 1 = 11111111 (notación en complemento a 2)
(*e*) − 2 = 11111110 (notación en complemento a 2)
(*f*) − 128 = 10000000 (notación en complemento a 2)

2.26. Utilizando el procedimiento de la Figura 2.13*a*, convertir los siguientes números decimales con signo en su forma de complemento a 1:

(*a*) − 10 (*b*) − 21 (*c*) − 34 (*d*) − 96

Solución:

Acudir a la Figura 2.13*a*. Las notaciones en complemento a 2 de los números decimales con signo son como sigue:

(*a*) Primer paso (convertir el decimal a binario): $10_{10} = 00001010_2$
Segundo paso (complementar): $00001010_2 \rightarrow 11110101$
Tercer paso (sumar 1): 11110101 + 1 = 11110110 (complemento a 2) = -10_{10}

(*b*) Primer paso (convertir el decimal a binario): $21_{10} = 00010101_2$
Segundo paso (complementar): $00010101_2 \rightarrow 11101010$
Tercer paso (sumar 1): 11101010 + 1 = 11101011 (complemento a 2) = -21_{10}

(*c*) Primer paso (convertir el decimal a binario): $34_{10} = 00100010_2$
Segundo paso (complementar): $00100010_2 \rightarrow 11011101$
Tercer paso (sumar 1): 11011101 + 1 = 11011110 (complemento a 2) = -34_{10}

(*d*) Primer paso (convertir el decimal a binario): $96_{10} = 01100000_2$
Segundo paso (complementar): $01100000_2 \rightarrow 10011111$
Tercer paso (sumar 1): 10011111 + 1 = 10100000 (complemento a 2) = -96_{10}

2.27. El complemento a 2 y los patrones de bits _____ (BCD, binario) son los mismos para los números decimales *positivos*.

Solución:

El complemento a 2 y los patrones de bits binarios son los mismos para los números decimales positivos.

2.28. Utilizando el procedimiento de la Figura 2.13*b*, traducir las siguientes notaciones en complemento a 2 a sus decimales equivalentes con signo:
(*a*) 11111011 (*b*) 00001111 (*c*) 10001111 (*d*) 01110111

Solución:

Los decimales equivalentes con signo de las notaciones en complemento a 2 son los siguientes:

(*a*) 11111011 (complemento a 2) = -5_{10}

$$11111011 \xrightarrow{\text{complementa}} \begin{array}{r} 00000100 \\ +\quad\ 1 \\ \hline 00000101 = 5 \end{array}$$

(*b*) 00001111 (complemento a 2) = + 15

(*c*) 10001111 (complemento a 2) = − 113

$$10001111 \xrightarrow{\text{complementa}} \begin{array}{r} 01110000 \\ +\quad\ 1 \\ \hline 01110001 = 113 \end{array}$$

(*d*) 01110111 (complemento a 2) = + 119

2.6. ARITMETICA EN COMPLEMENTO A 2

Un microprocesador puede utilizar números en complemento a 2 porque puede *complementar, incrementar* (sumar +1 a un número) y *sumar* números binarios. Los microprocesadores no tienen circuitería para restar, en su lugar utilizan un sumador y números en complemento a 2 para realizar la sustracción.

Sumar los números decimales con signo +5 y +3. El proceso se muestra en la Figura 2.14*a* en decimal y en la forma en complemento a 2. A partir de la tabla de la Figura 2.12 se encuentra que +5 es igual a 00000101 en su notación en complemento a 2, mientras que +3 es igual a 00000011. Los números en complemento a 2 00000101 y 00000011 se suman entonces igual que los números binarios regulares, obteniéndose la suma en complemento a 2 de 00001000. La suma 00001000 es igual a $+8_{10}$.

Primer sumando	(+5)	00000101
Segundo sumando	+ (+3)	+00000011
Suma	(+8)	00001000

(a) Problema de suma en complemento a 2

Primer sumando	(+3)	00000011
Segundo sumando	+ (−8)	+11111000
Suma	(−5)	11111011

(c) Problema de suma en complemento a 2

Primer sumando	(+7)	00000111
Segundo sumando	+ (−3)	+ 11111101
Suma	(+4)	(1) 00000100

Descarta overflow

(b) Problema de suma en complemento a 2

Primer sumando	(−2)	11111110
Segundo sumando	+ (−5)	+ 11111011
Suma	(−7)	(1) 11111001

Descarta overflow

(d) Problema de suma en complemento a 2

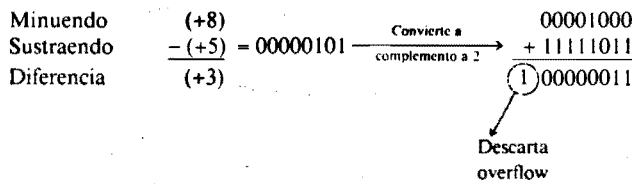
Figura 2.14.

Sumar los números decimales con signo +7 y −3. El procedimiento utilizando la suma en decimal y en complemento a 2 se muestra en la Figura 2.14*b*. A partir de la tabla de la Figura 2.12 se encuentra que +7 = 00000111 y −3 = 11111101 en la notación en complemento a 2. Los números en complemento a 2 se suman entonces (00000111 y 11111101) como si fuesen números binarios regulares, obteniéndose la suma de 100000100. El MSB es un arrastre de «overflow» del registro de 8 bits y se descarta. Por tanto, la suma en complemento a 2 es igual a 00000100, ó $+4_{10}$.

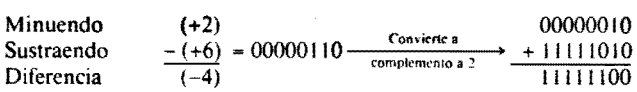
Sumar los números decimales con signo +3 y −8. El procedimiento se detalla en la Figura 2.14*c*. A partir de la tabla de la Figura 2.12 se encuentra que +3 = 00000011 y −8 = 11111000 en notación en complemento a 2. Entonces se suman estos números en complemento a 2 como si fuesen números binarios regulares, obteniéndose una suma de 11111011. De nuevo de la Figura 2.12 se determina que la suma en complemento a 2 de 11111011 es igual -5_{10}.

Sumar los decimales −2 y −5. El procedimiento se detalla en la Figura 2.14*d*. A partir de la tabla de la Figura 2.12 se encuentra que la notación en complemento a 2 de −2 = 11111110 y la de −5 = 11111011. Estos números se suman entonces como si fuesen números binarios regulares. La suma es 111111001. El MSB es un arrastre de «overflow» del registro de 8 bits y se descarta. La suma de los números en complemento a 2 (11111110 y 11111011) es entonces 11111001 (en complemento a 2). Utilizando la Figura 2.12 se determina que la suma en complemento a 2 de 11111001 es igual a -7_{10}.

Restar los números decimales con signo +5 de +8. El procedimiento se esboza en la Figura 2.15*a*. El minuendo (+8) es igual a 00001000. El sustraendo es +5, ó 00000101. El 00000101 debe convertirse a su forma en complemento a 2 (complementar y sumar 1), dando 11111011. El «minuendo» de 00001000 se suma al complemento a 2 del sustraendo 11111011 como si fuesen números binarios. La suma es igual a 100000011. El MSB es un arrastre de «overflow» del registro y se descarta, obteniéndose la suma de 00000011. A partir de la tabla de la Figura 2.12 se determina que la suma en complemento a 2 de 00000011 es igual a $+3_{10}$. Observar que al restar, el sustraendo se convierte a su forma en complemento a 2 y después se suma al minuendo. *Utilizando la representación en complemento a 2 y un sumador, el microprocesador puede realizar la sustración.*



(a) Problema de resta en complemento a 2 utilizando suma



(b) Problema de resta en complemento a 2 utilizando suma

Figura 2.15.

Restar el número decimal +6 de +2. El procedimiento se muestra en la Figura 2.15*b*. El minuendo +2 es igual 00000010. El sustraendo +6 es igual 00000110. Este sustraendo se convierte a su forma en complemento a 2 (complementar y sumar 1) dando 11111010. Los números

(00000010 y 11111010) se suman como si fuesen números binarios dando una suma de 11111100. De la tabla de la Figura 2.12 se determina que la suma en complemento a 2 de 11111100 es igual a -4_{10}.

PROBLEMAS RESUELTOS

2.29. Sumar los siguientes números decimales con signo utilizando números en complemento a 2:

(*a*) $(+7) + (+1)$ (*b*) $(+31) + (+26)$

Solución:

Seguir el procedimiento de la Figura 2.14*a*. Las sumas son las siguientes:

(*a*) $(+7)$ 00000111
$+(+1)$ +00000001
$(+8) =$ 00001000 (complemento a 2)

(*b*) $(+31)$ 00011111
$+(+26)$ +00011010
$(+57) =$ 00111001 (complemento a 2)

2.30. Sumar los siguientes números decimales con signo utilizando números en complemento a 2:

(*a*) $(+8) + (-5)$ (*b*) $(+89) + (-46)$

Solución:

Seguir el procedimiento de la Figura 2.14*b*. Las sumas son las siguientes:

(*a*) $(+8)$ 00001000
$+(-5)$ +11111011
$(+3) =$ (1)00000011 (complemento a 2)
descarta overflow

(*b*) $(+89)$ 01011001
$+(-46)$ +11010010
$(+43) =$ (1)00101011 (complemento a 2)
descarta overflow

2.31. Sumar los siguientes números decimales con signo utilizando números en complemento a 2:

(*a*) $(+1) + (-6)$ (*b*) $(+20) + (-60)$

Solución:

Seguir el procedimiento de la Figura 2.14*c*. Las sumas son las siguientes:

(*a*) $(+1)$ 00000001
$+(-6)$ +11111010
$(-5) =$ 11111011 (complemento a 2)

(*b*) $(+20)$ 00010100
$+(-60)$ +11000100
$(-40) =$ 11011000 (complemento a 2)

2.32. Sumar los siguientes números decimales con signo utilizando números en complemento a 2:

(*a*) $(-3) + (-4)$ (*b*) $(-13) + (-41)$

Solución:

Seguir el procedimiento de la Figura 2.14*d*. Las sumas son las siguientes:

(*a*) (-3) 11111101
$+(-4)$ $+11111100$
$(-7) = (1)11111001$ (complemento a 2)

descarta overflow

(*b*) (-13) 11110011
$+(-41)$ $+11010111$
$(-54) = (1)11001010$ (complemento a 2)

descarta overflow

2.33. Restar los siguientes números decimales con signo utilizando números en complemento a 2:

(*a*) $(+7)$ (*b*) $(+113)$
$-(+2)$ $-(+50)$

Solución:

Seguir el procedimiento de la Figura 2.15*a*. Los resultados de los problemas de resta son los siguientes:

(*a*)
$(+7)$ 00000111
$-(+2) = 00000010$ —complementa a 2 y suma→ $+11111110$
$(+5) =$ $(1)00000101$ (complemento a 2)

descarta overflow

(*b*)
$(+113)$ 01110001
$-(+50)$ 00110010 —complementa a 2 y suma→ $+11001110$
$(+63)$ $(1)00111111$ (complemento a 2)

descarta overflow

2.34. Restar los siguientes números decimales con signo utilizando números en complemento a 2:

(*a*) $(+3)$ (*b*) $(+12)$
$-(+8)$ $-(+63)$

Solución:

Seguir el procedimiento de la Figura 2.15*b*. Los resultados de los problemas de resta son los siguientes:

(*a*)
$(+3)$ 00000011
$-(+8) = 00001000$ —complementa a 2 y suma→ $+11111000$
$(-5) =$ $+11111011$ (complemento a 2)

(*b*)
$(+12)$ 00001100
$-(+63) = 00111111$ —complementa a 2 y suma→ $+11000001$
$(-51) =$ 11001101 (complemento a 2)

2.7. AGRUPACIONES DE BITS

Un simple dígito binario se denomina *bit*. Cuatro bits agrupados se denominan *nibble*. Ocho bits agrupados se denominan *byte*.

Una característica muy importante de cualquier microprocesador es el tamaño del acumulador. Los microprocesadores sencillos, comúnmente utilizan acumuladores de 8 bits. El *tamaño de la palabra* del microprocesador entonces es de 8 bits. En este caso 1 byte forma una palabra. Los microprocesadores tienen longitudes de palabra de 4, 8, 16 o incluso 32 bits. Un microprocesador de 16 bits tiene por tanto una longitud de palabra de 2 bytes, o de 16 dígitos binarios. Una *palabra* es un grupo de bits que es procesada como un simple número o instrucción por el microprocesador. Un microprocesador de 8 bits transfiere y almacena todos los datos en grupos de 8 bits, vía ocho conductores paralelos denominados bus de datos.

El contenido de la memoria de una microcomputadora de 8 bits puede ser el reflejado en la Figura 2.16*a*. Observar que cada posición de dirección (contenido etiquetado) contiene información en grupos de 8 bits. Cada byte se denomina *palabra de memoria*, ya que el microprocesador es una unidad de 8 bits. Cada palabra de memoria tiene un significado específico cuando es buscada y decodificada por el microprocesador. El contenido binario de la memoria de la Figura 2.16*a* puede representar:

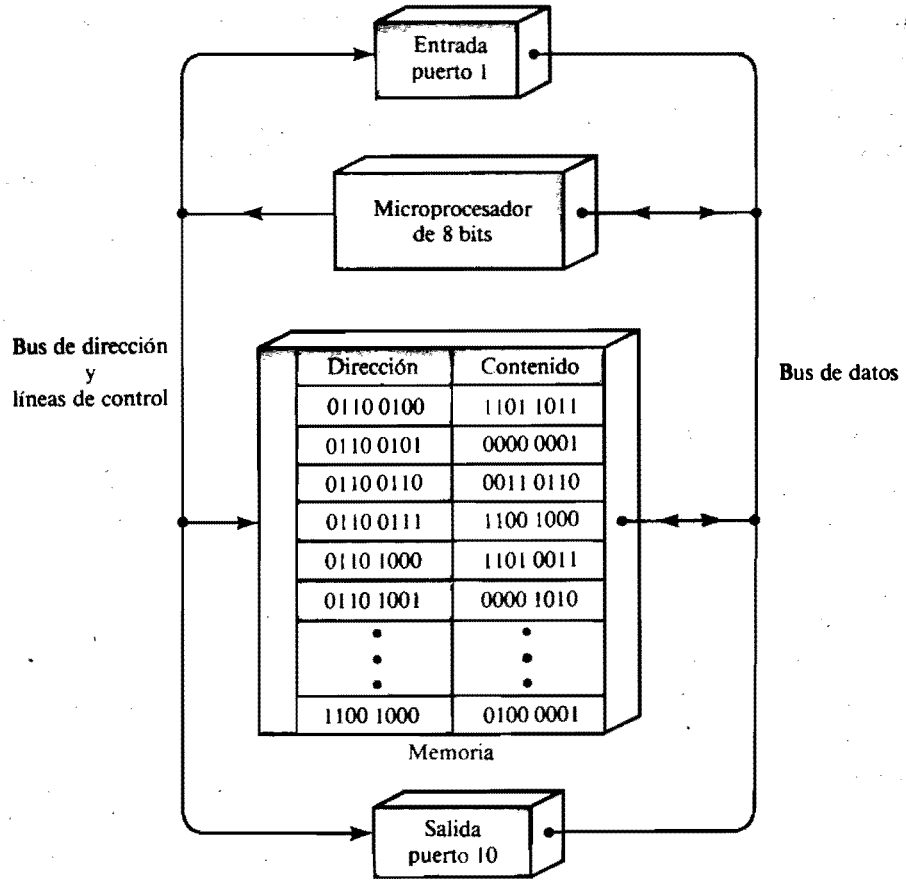
1. Un número binario.
2. Un número binario con signo.
3. Un número BCD.
4. Un carácter (una letra del alfabeto).
5. Una instrucción.
6. Una dirección de memoria.
7. Una dirección de un puerto de entrada o salida.

Considerar la posición superior de la memoria (01100100_2) de la Figura 2.16*a*. El contenido de esta posición de memoria es 11011011. Esta palabra de memoria puede interpretarse como sigue:

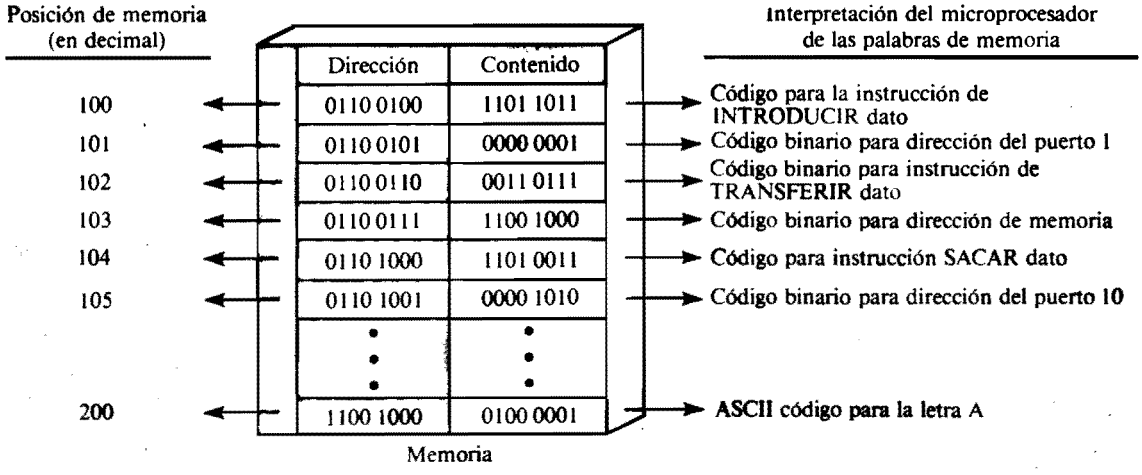
1. Como un número binario —$11011011_2 = 219_{10}$.
2. Como un número binario con signo —11011011 (complemento a 2) = -37_{10}.
3. Como un número BCD —esto no puede ser un número decimal codificado binario porque ni 1101 ni 1011 son códigos BCD.
4. Como un carácter —esto no puede ser ningún carácter ASCII (ASCII es un código alfanumérico popular especial).
5. Como una instrucción —11011011 = la instrucción INPUT (ENTRAR) del popular microprocesador Intel 8080/8085.
6. Como una dirección de memoria —$11011011_2 = DB_{16}$ = posición de memoria 219_{10}.
7. Como una dirección para un puerto de entrada o salida —11011011_2 = puerto 219_{10}.

La palabra superior de la memoria de la Figura 2.16*a* puede ser el número binario 219_{10}, el número binario con signo -37_{10}, la instrucción INPUT (ENTRAR) del microprocesador 8085, la dirección de la posición de memoria DB_{16} o la dirección del puerto de entrada o salida 219_{10}. En este ejemplo la palabra superior de la memoria (11011011) puede significar una de cinco posiciones diferentes, instrucciones o cantidades.

Si el operador de la microcomputadora 8085 arrancase el contador de programa en la dirección 100_{10} (01100100_2), entonces el microprocesador buscaría y decodificaría la palabra de memoria 11011011 como una instrucción de INTRODUCIR dato (INPUT). El microprocesador continuaría posteriormente en la siguiente dirección: 101_{10} (01100101_2). El contenido de la me-



(a) Contenido típico de una memoria binaria de una microcomputadora



(b) Interpretación por el microprocesador del contenido de la memoria

Figura 2.16.

moria en la Figura 2.16*a* es el mismo que el de la **Figura 1.4.** Recordar que el programa de la Figura 1.4 ejecuta las siguientes instrucciones:

1. INTRODUCIR (INPUT) el dato del puerto 1.
2. ALMACENAR (STORE) el dato en la posición de memoria 200.
3. SACAR (OUTPUT) el dato al puerto 10.

La interpretación que el microprocesador hace del contenido de la memoria se detalla en la Figura 2.16*b*. Las instrucciones del programa se listan en las seis posiciones superiores de memoria ($100\text{-}105_{10}$). La posición inferior de memoria (200_{10}) es la posición donde se almacena el dato. En este caso, el código ASCII de la letra A se almacena en esta posición de memoria.

En resumen, es importante observar que en las microcomputadoras, los bits se agrupan en palabras. Estas palabras de la memoria de programa son interpretadas por el microprocesador secuencialmente, *una cada vez*. Es importante para el programador saber cómo el microprocesador realiza las secuencias e interpreta los datos. Cada microprocesador tiene su propio código, único, de instrucciones; sin embargo, todos los microprocesadores realizan el secuenciamiento de las posiciones de memoria de forma similar.

PROBLEMAS RESUELTOS

2.35. Cuatro bits agrupados se denominan nibble, mientras que un grupo de 8 bits se denomina _____ .

Solución:

Una agrupación de 8 bits se denomina byte. Bajo circunstancias especiales esta agrupación de 8 bits también puede denominarse palabra.

2.36. El tamaño de _____ es una característica importante de un microprocesador y está relacionada con el número de bits transferidos y procesados como un grupo.

Solución:

El tamaño de palabra es una de las características más importantes de un microprocesador.

2.37. Acudir a la Figura 2.16*a*. El byte de datos almacenados en cualquier posición de memoria se denomina palabra de _____ .

Solución:

El byte de datos almacenados en cualquier posición de memoria se denomina palabra de memoria.

2.38. Citar siete interpretaciones posibles de una palabra de memoria de 8 bits.

Solución:

El contenido de una posición de memoria puede interpretarse como un número binario, un número binario con signo (notación en complemento a 2), un número BCD, un carácter, una instrucción, una dirección de memoria y una dirección de un puerto de entrada o salida.

2.39. Acudir a la Figura 2.16*b*. ¿Cómo interpretaría el microprocesador la palabra de memoria (00000001) de la dirección 101_{10}?

Solución:

Acudir a la Figura 2.16*b*. El microprocesador busca la palabra de memoria (00000001) esperando que le diga por *qué puerto* va a entrar el dato. La palabra de memoria indica al microprocesador que va a entrar un dato por el puerto 1.

2.40. Acudir a la Figura 2.16*b*. ¿Cómo interpretaría el microprocesador, la palabra de memoria (00110111) de la dirección 102_{10}?

Solución:

Acudir a la Figura 2.16*b*. El microprocesador busca la palabra de memoria (00110111) esperando que sea una nueva instrucción. La palabra 00110111 es decodificada por el microprocesador y significa TRANSFERIR (MOVE) dato desde el acumulador hasta la posición de memoria cuya dirección se encuentra en la siguiente posición de memoria.

2.41. Los códigos utilizados para las instrucciones son los mismos para todos los microprocesadores, y por tanto un programa escrito para un microprocesador Intel funcionará en una unidad de Motorola (verdadero o falso).

Solución:

Falso. Cada microprocesador tiene su propio código único de instrucciones.

2.8. CODIGOS ALFANUMERICOS

Los códigos que contienen caracteres alfabéticos y numéricos son necesarios cuando los microprocesadores se comunican con dispositivos como teletipos o terminales CRT (tubo de rayos catódicos). Estos códigos se denominan *códigos alfanuméricos.*

El código alfanumérico más utilizado en los sistemas microcomputadores es el *American Standard Code for Information Interchange* o ASCII (pronunciado «ask-i») *(Código Estándar Americano para Intercambio de Información).* Un listado parcial del código de 7 bits se muestra en la Figura 2.17. Este listado contiene códigos de 7 bits para números, letras mayúsculas y caracteres de puntuación. El código completo tiene también códigos para las letras minúsculas y los caracteres de control.

PROBLEMAS RESUELTOS

2.42. Los códigos binarios que representan números y letras se denominan códigos _____.

Solución:

Los códigos binarios utilizados para representar números y letras son los códigos alfanuméricos.

2.43. La representación ASCII del número 0 es 011 0000. El número 9 se representa en ASCII por _____ (7 bits).

Solución:

Acudir a la Figura 2.17. El número 9 se representa en ASCII por 011 1001.

Carácter	ASCII	Carácter	ASCII
Espacio	010 0000	A	100 0001
!	010 0001	B	100 0010
"	010 0010	C	100 0011
#	010 0011	D	100 0100
$	010 0100	E	100 0101
%	010 0101	F	100 0110
&	010 0110	G	100 0111
'	010 0111	H	100 1000
(	010 1000	I	100 1001
)	010 1001	J	100 1010
*	010 1010	K	100 1011
+	010 1011	L	100 1100
,	010 1100	M	100 1101
-	010 1101	N	100 1110
.	010 1110	O	100 1111
/	010 1111	P	101 0000
0	011 0000	Q	101 0001
1	011 0001	R	101 0010
2	011 0010	S	101 0011
3	011 0011	T	101 0100
4	011 0100	U	101 0101
5	011 0101	V	101 0110
6	011 0110	W	101 0111
7	011 0111	X	101 1000
8	011 1000	Y	101 1001
9	011 1001	Z	101 1010

Figura 2.17. Lista parcial del conjunto de caracteres ASCII.

2.44. Al eliminar los tres bits más significativos de la representación ASCII del 0 al 9 se obtendrá el _____ equivalente de ese número.

Solución:

Acudir a la Figura 2.17. Al eliminar los tres bits más significativos en la representación ASCII del 0 al 9 se obtendrá el equivalente binario o BCD de ese número.

PROBLEMAS SUPLEMENTARIOS

2.45. Mentalmente, convertir los siguientes números binarios a sus equivalentes decimales:
(*a*) 0000 (*b*) 0010 (*c*) 0011 (*d*) 0111 (*e*) 1001 (*f*) 1100
Res. (*a*) 0 (*b*) 2 (*c*) 3 (*d*) 7 (*e*) 9 (*f*) 12

2.47. $01101001_2 =$ _____ $_{10}$.
Res. 105_{10}.

2.47. $60_{10} =$ _____ $_2$.
Res. 111100_2.

2.48. El número binario 10011100 se representa como 9C en notación _____.
Res. hexadecimal.

2.49. $8D_{16} =$ _____ $_2$.
Res. 10001101_2.

2.50. $01011111_2 =$ _____ $_{16}$.
Res. $5F_{16}$.

2.51. $3C_{16} =$ _____ $_{10}$.
Res. 60_{10}.

2.52. $90_{10} =$ _____ $_{16}$.
Res. $5A_{16}$.

2.53. $92_{10} =$ _____ $_{BCD}$.
Res. $1001\ 0010_{BCD}$.

2.54. $1000\ 0110_{BCD} =$ _____ $_{10}$.
Res. 86_{10}.

2.55. Resolver los siguientes problemas de suma binaria:

(*a*)
$$\begin{array}{r} 11000011 \\ +\ 00111100 \\ \hline \end{array}$$

(*b*)
$$\begin{array}{r} 01101110 \\ +\ 00111101 \\ \hline \end{array}$$

Res. (*a*) 11111111_2 (*b*) 10101011_2

2.56. $11011000_2 - 00110011_2 =$ _____ $_2$.
Res. 10100101_2.

2.57. $1001_2 \times 1101_2 =$ _____ $_2$.
Res. 1110101_2.

2.58. Cuando se almacenan números positivos y negativos en un registro del microprocesador, si el bit de signo (MSB) es 1, el número es _____ (negativo, positivo).
Res. negativo.

2.59. 01111110 en notación en complemento a 2 representa un número _____ (negativo, positivo).

Res. positivo.

2.60. Traducir los siguientes números decimales con signo a su forma en complemento a 2 de 8 bits:
(*a*) + 12 (*b*) − 12.

Res. (*a*) 00001100 (complemento a 2) (*b*) 11110100 (complemento a 2).

2.61. Traducir los siguientes números en complemento a 2 a sus decimales con signo equivalentes:
(*a*) 01110100 (*b*) 11011101.

Res. (*a*) $+116_{10}$ (*b*) -35_{10}.

2.62. Sumar los siguientes números decimales con signo utilizando números de 8 bits en complemento a 2:

(*a*) (+ 13) (*b*) (+ 17) (*c*) (− 6)
+ (+ 8) + (− 8) + (− 14)

Res. (*a*) 00010101 (complemento a 2) (*b*) 00001001 (complemento a 2)
(*c*) 11101100 (complemento a 2)

2.63. Restar los siguientes decimales con signo utilizando números en complemento a 2 de 8 bits:

(*a*) (+ 13) (*b*) (+ 19)
− (+ 5) − (+ 29)

Res. (*a*) 00001000 (complemento a 2) (*b*) 11110110 (complemento a 2)

2.64. Un byte es un grupo que contiene _____ bits.

Res. 8.

2.65. Un nibble es un grupo que contiene _____ bits.

Res. 4.

2.66. El tamaño de palabra más popular para los sencillos microprocesadores es de _____ (8,48) bits.

Res. 8.

2.67. Acudir a la Figura 2.16*a*. Se trata de un diagrama de bloques básicos de _____ (una microcomputadora, un microprocesador).

Res. una microcomputadora.

2.68. Acudir a la Figura 2.16*b*. ¿Cómo interpreta el microprocesador la palabra de memoria (11010011) de la dirección 104_{10}?

Res. Acudir a la Figura 2.16*b*. El microprocesador busca la palabra de memoria (11010011) esperando que sea una nueva instrucción. La palabra (11010011) es decodificada por el microprocesador y significa SACAR (OUTPUT) dato del acumulador al puerto de salida cuyo número está en la siguiente posición de memoria.

2.69. Las letras ASCII significan _____ .

Res. American Standard Code for Information Interchange (Código Americano Estándar para Intercambio de Información).

2.70. Un código _____ se utiliza, probablemente, para traducir información desde un dispositivo de entrada (teclado) a un sistema microcomputador.

Res. alfanumérico o ASCII.

Capítulo 3

DISPOSITIVOS DIGITALES BASICOS

3.1. PUERTAS LOGICAS

Los circuitos utilizados para procesar señales digitales se denominan *puertas lógicas.* Para identificar estos circuitos se utilizan *símbolos lógicos.* Las siete puertas que son los elementos lógicos fundamentales en los sistemas digitales aparecen en la Figura 3.1.

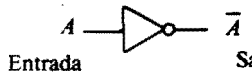
Los nombres de los siete elementos lógicos se dan en la columna izquierda de la Figura 3.1. Los símbolos lógicos estándares gráficos se ilustran en la segunda columna. La tercera muestra la acción exacta (salida) del circuito con entradas dadas. En la tabla de verdad, un 0 significa un nivel de tensión BAJO mientras que un 1 significa un nivel de tensión ALTO. La columna derecha de la Figura 3.1 da la *expresión booleana* para cada elemento lógico. Las expresiones booleanas son una especie de «taquigrafía» utilizada para representar una función lógica.

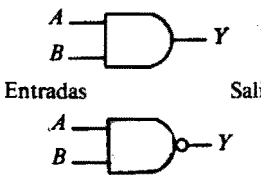
Observar que cada símbolo lógico de la Figura 3.1 tiene una forma distintiva. Algunos fabricantes están comenzando a utilizar símbolos rectangulares para todos los dispositivos lógicos. Estos nuevos símbolos lógicos están basados en el no ensayado estándar (IEEE Standard 91-1984) desarrollado por el Instituto de Electricidad e Ingeniería Electrónica. Sin embargo, por ahora, todos los trabajadores de esta área reconocerán los símbolos lógicos dibujados en la Figura 3.1.

El ejemplo de la Figura 3.2 es práctico con la utilización de información con las puertas lógicas. ¿Cuáles son las salidas del inversor (a veces denominado puerta NOT) de la Figura 3.2 con el pulso *a* en la entrada? De acuerdo con la segunda línea de la tabla de verdad de la Figura 3.1, la salida será 0, la opuesta a la entrada. Cuando el pulso *b* (un 0, o BAJO) alcanza la entrada del inversor, la salida será un nivel ALTO o un 1. El pulso de entrada *c* producirá una salida de nivel BAJO, mientras que el pulso *d* producirá una salida de nivel ALTO. El proceso de inversión también se denomina *complementación* o *negación.* La expresión booleana para la complementación es $A = \overline{A}$ (leer A complementado es igual a no A o A no). La barra superior se llama *vinculo* y se utiliza para invertir o complementar la variable o expresión que está debajo.

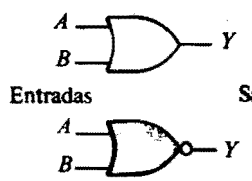
Otro ejemplo se muestra en la Figura 3.3*a*, donde aparece una puerta AND de 2 entradas. Los pulsos de entrada en *a* son 0 y 1, que según la tabla de verdad de la Figura 3.1 producirán una salida de nivel BAJO ó 0. Los pulsos *a, b* y *c* producirán todos salidas de nivel BAJO. Cuando todas las entradas de la puerta AND estén en el nivel ALTO (ver pulso *d* en la Fig. 3.3*a*), la salida será un nivel ALTO ó 1.

Considerar el problema mostrado en la Figura 3.3*b*. Aquí los «nibbles» 1010 (en la entrada *A*) y 1001 (en la entrada *B*) van a realizar la operación OR. El «nibble» de salida se determinará utilizando la tabla de verdad de la Figura 3.1. El «nibble» de salida será 1011 como resultado de la operación OR de 1010 con 1001. Observar que primero realizan la operación OR los pulsos *a*, después los *b*, etc.

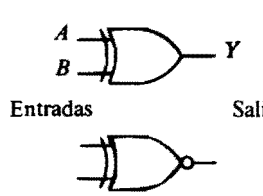
<table>
<tr><th>Función lógica</th><th>Símbolo lógico de puerta</th><th>Tabla de verdad</th><th>Expresión booleana</th></tr>
<tr><td>Inversor</td><td> 
</td><td>
<table>
<tr><th>Entrada</th><th>Salida</th></tr>
<tr><td>A</td><td>$\overline{A}$</td></tr>
<tr><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td></tr>
</table>
</td><td>$A = \overline{A}$</td></tr>
<tr><td>AND

NAND</td><td> 
</td><td>
<table>
<tr><th colspan="2">Entradas</th><th colspan="2">Salidas</th></tr>
<tr><th>B</th><th>A</th><th>AND</th><th>NAND</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td></tr>
</table>
</td><td>$A \cdot B = Y$

$\overline{A \cdot B} = Y$</td></tr>
<tr><td>OR

NOR</td><td> 
</td><td>
<table>
<tr><th colspan="2">Entradas</th><th colspan="2">Salidas</th></tr>
<tr><th>B</th><th>A</th><th>OR</th><th>NOR</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td></tr>
</table>
</td><td>$A + B = Y$

$\overline{A + B} = Y$</td></tr>
<tr><td>OR exclusiva

NOR exclusiva</td><td> 
</td><td>
<table>
<tr><th colspan="2">Entradas</th><th colspan="2">Salidas</th></tr>
<tr><th>B</th><th>A</th><th>XOR</th><th>XNOR</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td></tr>
</table>
</td><td>$A \oplus B = Y$

$\overline{A \oplus B} = Y$</td></tr>
</table>

Figura 3.1. Las siete funciones lógicas comparadas.

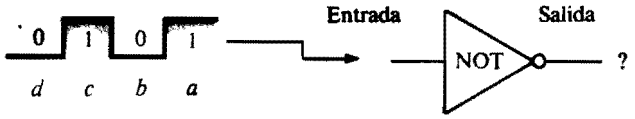
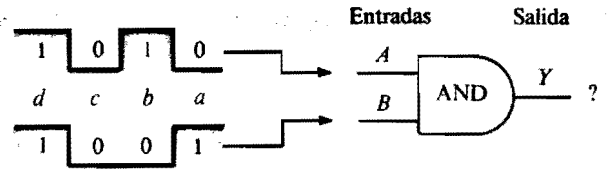
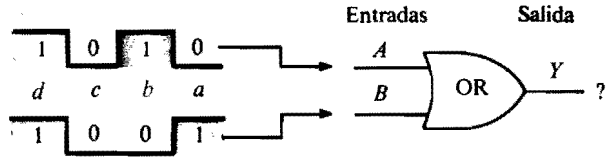


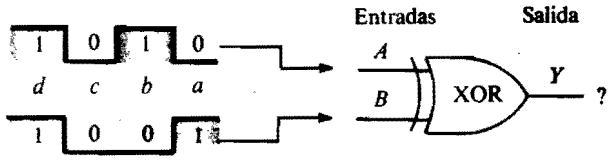
Figura 3.2. Problema del inversor.



(a) Problema de la puerta AND



(b) Problema de la puerta OR



(c) Problema de la puerta XOR

Figura 3.3.

¿Cuál es el «nibble» de salida después que 1010 y 1001 realicen la operación OR exclusiva (XOR) en la Figura 3.3*c*. Utilizando la tabla de verdad XOR de la Figura 3.1, se determina que el resultado de la operación XOR de 1010 con 1001 es 0011.

Los microprocesadores pueden realizar operaciones lógicas. Normalmente un microprocesador tendrá instrucciones NOT (complementación o negación), AND, OR y XOR. Estas instrucciones operan sobre grupos de bits similares a los de las puertas lógicas de las Figuras 3.2 y 3.3.

PROBLEMAS RESUELTOS

3.1. Citar los nombres de siete funciones lógicas.

Solución:

Acudir a la Figura 3.1. Las siete funciones lógicas se denominan NOT (inversor) AND, NAND, OR, NOR, OR exclusiva (XOR) y NOR exclusiva (XNOR).

3.2. Citar los nombres de cuatro funciones lógicas que pueden ser realizadas por las sencillas instrucciones de un microprocesador.

Solución:

Muchos microprocesadores tienen instrucciones que realizan las funciones lógicas NOT, AND, OR y XOR.

3.3. Si un microprocesador realiza la operación AND de 1100 con 1011, el «nibble» de salida será _____ .

Solución:

Acudir a la tabla de verdad AND de la Figura 3.1. La salida resultante de la operación AND de 1100 con 1011 será 1000.

3.4. Si un microprocesador realiza la operación OR de 0011 con 1000, el «nibble» de salida será _____ .

Solución:

Acudir a la tabla de verdad OR de la Figura 3.1. El resultado de salida de la operación OR de 0011 con 1000 será 1011.

3.5. Si un microprocesador complementa (NOT) 1001, el «nibble» de salida será _____ .

Solución:

Acudir a la tabla de verdad del inversor de la Figura 3.1. La salida resultante de complementar 1001 será 0110.

3.6. Si un microprocesador realiza la operación XOR de 0011 con 0110, el «nibble» de salida será _____ .

Solución:

Acudir a la tabla de verdad OR exclusiva de la Figura 3.1. La salida resultante será 0101.

3.7. Describir las salidas de la puerta NAND de la Figura 3.4.

Solución:

Acudir a la tabla de verdad NAND de la Figura 3.1. Los pulsos de salida de la Figura 3.4 serán los siguientes:

pulso $a = 1$ pulso $c = 1$
pulso $b = 0$ pulso $d = 1$

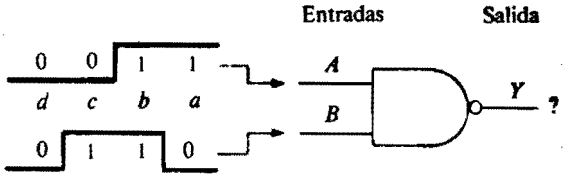


Figura 3.4. Problema del tren de pulsos.

3.8. Describir la salida de la puerta XNOR de la Figura 3.5.

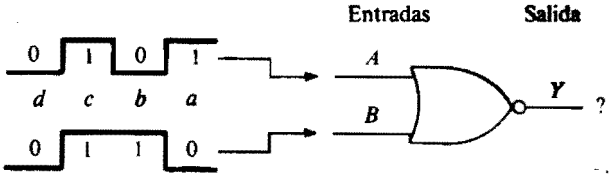


Figura 3.5. Problema del tren de pulsos.

Solución:

Acudir a la tabla de verdad NOR exclusiva de la Figura 3.1. Los pulsos de salida de la Figura 3.5 serán los siguientes:

pulso $a = 0$ pulso $c = 1$
pulso $b = 0$ pulso $d = 1$

3.2. COMBINACION DE PUERTAS LOGICAS

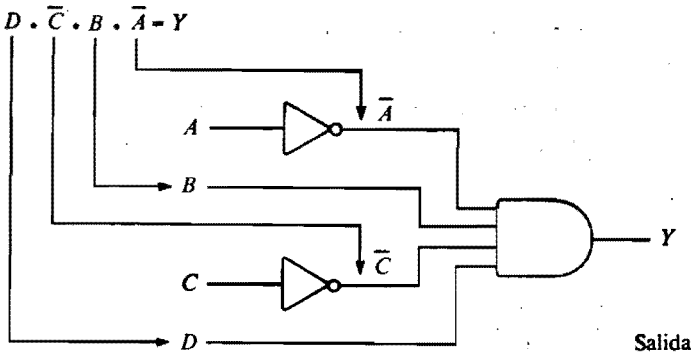
Los sistemas digitales están formados por combinaciones de puertas lógicas, las cuales pueden ser descritas por una tabla de verdad, expresiones booleanas o diagramas de símbolos lógicos.

Considerar la tabla de verdad de la Figura 3.6*a*, que muestra todas las posibles combinaciones de cuatro entradas (*D*, *C*, *B* y *A*). Observar que sólo la combinación 1010 producirá una salida 1,

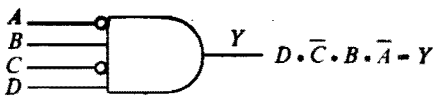
Entradas				Salida	Entradas				Salida
D	*C*	*B*	*A*	*Y*	*D*	*C*	*B*	*A*	*Y*
0	0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	1	0
0	0	1	0	0	1	0	1	0	1
0	0	1	1	0	1	0	1	1	0
0	1	0	0	0	1	1	0	0	0
0	1	0	1	0	1	1	0	1	0
0	1	1	0	0	1	1	1	0	0
0	1	1	1	0	1	1	1	1	0

$\rightarrow D \cdot \overline{C} \cdot B \cdot \overline{A} = Y$

(a) **Conversión de una tabla de verdad a la expresión booleana equivalente**



(b) **Conversión de una expresión booleana a su diagrama de símbolos lógicos**



(c) Diagrama simplificado de los símbolos lógicos

Figura 3.6.

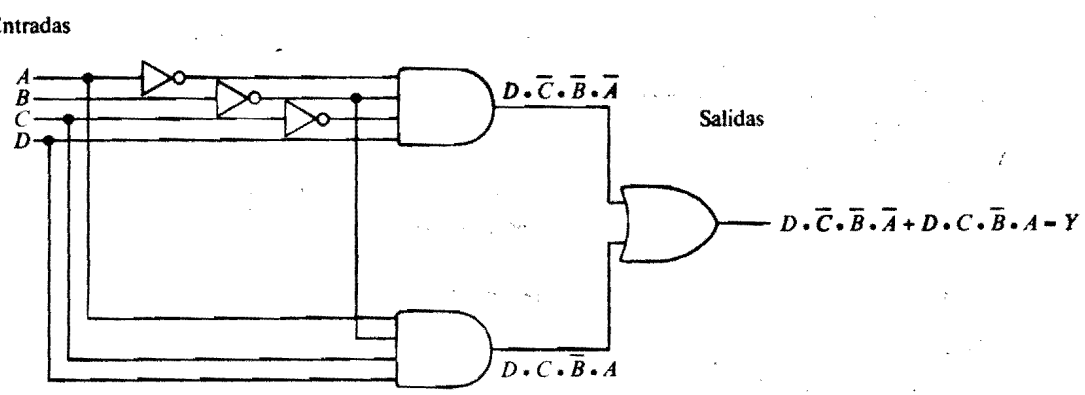
o de nivel ALTO. Una expresión booleana equivalente para esta tabla de verdad se indica en la parte derecha de la tabla de la Figura 3.6*a*. Las entradas realizan la operación AND, formando la expresión booleana $D \cdot \overline{C} \cdot B \cdot \overline{A} = Y$ (se lee D AND no C AND B AND no A igual a la salida Y).

A continuación se obtiene un diagrama de símbolos lógicos para la expresión booleana. Este procedimiento está esbozado en la Figura 3.6*b*. Las entradas A y C deben de complementarse utilizando un inversor. La salida es una puerta AND de 4 entradas.

Una versión simplificada del mismo diagrama lógico se muestra en la Figura 3.6*c*. En este diagrama los inversores aparecen como *burbujas inversoras*. Estas burbujas inversoras también pueden considerarse como *entradas activas en el nivel BAJO* (o en *BAJA*). En otras palabras, para activar la puerta AND de la Figura 3.6*c*, las entradas A y C deben estar en el nivel BAJO y las B y D en el nivel ALTO. Como las entradas B y D deben de estar en el nivel ALTO para activar la puerta AND, se consideran *entradas activas* en el nivel *ALTO* (o en *ALTA*).

Entradas				Salida	Entradas				Salida	
D	C	B	A	Y	D	C	B	A	Y	
0	0	0	0	0	1	0	0	0	1	$\rightarrow D \cdot \overline{C} \cdot \overline{B} \cdot \overline{A}$
0	0	0	1	0	1	0	0	1	0	
0	0	1	0	0	1	0	1	0	0	
0	0	1	1	0	1	0	1	1	0	o
0	1	0	0	0	1	1	0	0	0	
0	1	0	1	0	1	1	0	1	1	$\rightarrow D \cdot C \cdot \overline{B} \cdot A$
0	1	1	0	0	1	1	1	0	0	
0	1	1	1	0	1	1	1	1	0	

(a) Conversión de una tabla de verdad a la expresión booleana equivalente en minterms



(b) Conversión de una expresión booleana en minterms a su diagrama de símbolos lógicos

Figura 3.7.

Considerar la tabla de verdad de la Figura 3.7*a*. Dos combinaciones de entrada producirán un 1 ó salida ALTA. La expresión booleana de esta tabla de verdad es entonces $D \cdot \overline{C} \cdot \overline{B} \cdot \overline{A} + D \cdot C \cdot \overline{B} \cdot A = Y$ (se lee *D* AND no *C* AND no *B* AND no *A* OR *D* AND *C* AND no *B* AND *A* igual a la salida *Y*).

La expresión booleana es convertida a continuación a un diagrama de símbolos lógicos. Este procedimiento se ilustra en la Figura 3.7*b*. Observar que este tipo de expresión booleana crea patrones de puertas lógicas AND-OR, siendo la puerta OR la más cercana a la salida. El patrón de esta expresión booleana se denomina forma de *suma de productos* o *minterms*. Las expresiones booleanas son generadas a partir de los unos de la columna de salida de la tabla de verdad de la forma mostrada en la Figura 3.7*a*.

PROBLEMAS RESUELTOS

3.9. Acudir a la Figura 3.6*c*. Las entradas *B* y *D* son entradas activas en el nivel _____ (ALTO, BAJO), mientras que las entradas *A* y *C* son activas en el nivel _____ (ALTO, BAJO).

Solución:

En la Figura 3.6, las entradas *B* y *D* son activas en el nivel ALTO, mientras que *A* y *C* son activas en el nivel BAJO. En otras palabras, la puerta AND de 4 entradas puede ser activada (la salida está en el nivel ALTO) solamente cuando las entradas *A* y *C* son BAJAS y las *B* y *D* son ALTAS.

3.10. Escribir la expresión booleana de la tabla de verdad de la Figura 3.8.

Solución:

Hay dos combinaciones de entradas que producen una salida ALTA en la tabla de verdad de la Figura 3.8. A partir de estas combinaciones se desarrolla la expresión booleana en minterms: $\overline{D} \cdot \overline{C} \cdot \overline{B} \cdot A + D \cdot C \cdot B \cdot A = Y$.

Entradas				Salida	Entradas				Salida
D	*C*	*B*	*A*	*Y*	*D*	*C*	*B*	*A*	*Y*
0	0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	1	1
0	0	1	0	0	1	0	1	0	0
0	0	1	1	0	1	0	1	1	0
0	1	0	0	0	1	1	0	0	0
0	1	0	1	0	1	1	0	1	0
0	1	1	0	0	1	1	1	0	0
0	1	1	1	0	1	1	1	1	1

Figura 3.8.

3.11. Dibujar el diagrama de símbolos lógicos de un circuito que genere la tabla de verdad de la Figura 3.8.

Solución:

La expresión booleana para la tabla de verdad mostrada en la Figura 3.8 es $D \cdot \overline{C} \cdot \overline{B} \cdot A + D \cdot C \cdot B \cdot A = Y$. A partir de esta expresión se dibuja el diagrama de símbolos lógicos AND-OR mostrado en la Figura 3.9.

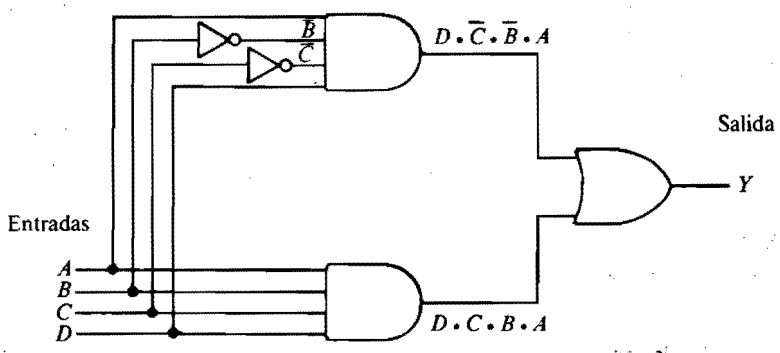


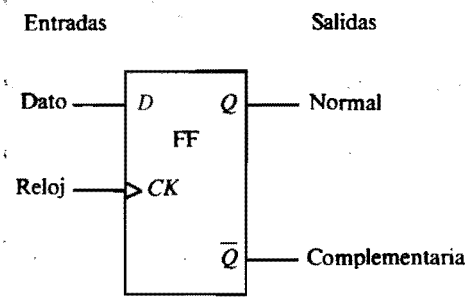
Figura 3.9. Solución del diagrama lógico.

3.3. FLIP-FLOPS Y CERROJOS

Los circuitos lógicos pueden clasificarse en dos grandes categorías. El primer grupo es el de los *circuitos lógicos combinacionales,* que está compuesto de puertas. El segundo grupo es el de los *circuitos lógicos secuenciales,* que incluye unos dispositivos denominados «flip-flops». Los flip-flops están interconectados para formar circuitos lógicos secuenciales para almacenamiento de datos, temporización, conteo y secuenciamiento. Los flip-flops tienen una *característica de memoria* extremadamente valiosa. Un flip-flop «recordará» sus entradas aún después de que éstas se hayan eliminado. Una puerta lógica sin embargo, «no recordará» su estado de salida después que se eliminen las entradas.

El símbolo lógico para el *flip-flop D* (también llamado *flip-flop de datos*) aparece en la Figura 3.10*a*. El símbolo lógico muestra dos entradas etiquetadas *D* (para datos) y *CK* (para reloj). Los flip-flops habitualmente tienen salidas complementarias etiquetadas Q y $\overline{Q}$ (no Q). La salida Q es la utilizada con más frecuencia y se considera la salida *normal.* La salida $\overline{Q}$ se denomina la salida *complementaria* o invertida del flip-flop. El > en la entrada *CK* del símbolo lógico *D* denota que este flip-flop transfiere datos de la entrada a la salida en el flanco positivo (↑) del pulso de reloj.

Los modos de operación del flip-flop *D* se muestran en la columna izquierda de la tabla de la Figura 3.10*b*. Poner en *set* el flip-flop significa cargar un 1 en la salida normal (Q). La primera línea de la tabla de verdad de la Figura 3.10*b* muestra que colocando un 1 en la entrada *D* y pulsando una vez la entrada *CK* (reloj) la salida Q se pone a 1. La segunda línea muestra la reinicialización (*reset*) del flip-flop. *Reinicializar* significa borrar la salida Q (ponerla a 0). *Mantener (hold)* significa almacenar el dato de salida. Cuando el flip-flop está en el modo de mantenimiento, los cambios en el estado lógico de los datos de entrada no ocasionan ningún cambio en



(a) Símbolo lógico para el flip-flop *D*

Modo de operación	Entradas		Salidas	
	D	*CK*	*Q*	$\overline{Q}$
Set	1	↑	1	0
Reinicialización	0	↑	0	1
Mantenimiento	X	No pulso de reloj	Igual que antes	

0 = BAJO
1 = ALTO
X = irrelevante
↑ = transición BAJA-ALTA del pulso de reloj

(b) Tabla de verdad para el flip-flop *D*

Figura 3.10.

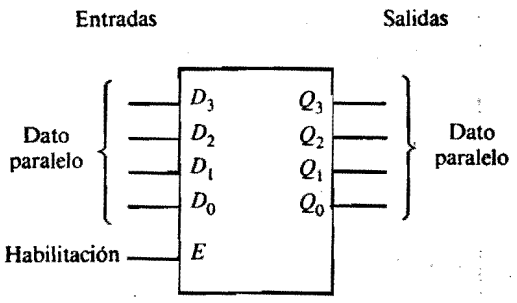
las salidas. La condición de mantenimiento ilustra la característica de memoria del flip-flop *D*. Observar que las operaciones de set y reinicialización se refieren a la salida *Q*.

Considerar el símbolo lógico de la Figura 3.11*a* correspondiente a un *cerrojo transparente de 4 bits*. Cada cerrojo (flip-flop) dentro del símbolo es un dispositivo de almacenamiento o memoria. Pensar que el cerrojo de 4 bits contiene 4 flip-flops *D* con sus entradas de reloj conectadas a la entrada de habilitación *(enable)* (*E*) mostrada en el símbolo de la Figura 3.11*a*. De acuerdo con la tabla de verdad del cerrojo de la Figura 3.11*b*, cuando la entrada de habilitación (*E*) está en ALTA (a 1), los datos de las entradas *D* son transferidos a sus respectivas salidas (Q_0–Q_3). La palabra de 4 bits llega en forma paralela (opuesta a la forma serie) y también sale en forma paralela. A esto se denomina entrada paralela y salida paralela. El dispositivo de almacenamiento cerrojo es una aplicación realizada con flip-flop.

El flip-flop *JK* es probablemente el más utilizado en los circuitos lógicos secuenciales, debido a su adaptabilidad. El símbolo lógico de un flip-flop *JK* se muestra en la Figura 3.12*a*. Este flip-flop tiene dos entradas de datos etiquetadas *J* y *K* más una entrada de reloj (*CK*) y las salidas *Q* (normal) y $\overline{Q}$ (complementaria).

Un flip-flop *JK* tiene cuatro modos de operación, que se describen en la Figura 3.12*b*. El *modo* de operación *de conmutación (toggle)* significa que con cada pulso, un tren de pulsos de reloj, las salidas cambian a su estado lógico opuesto. En este modo, la salida *Q* de un flip-flop *JK* será ALTA-BAJA-ALTA-BAJA, etc., con pulsos de reloj consecutivos.

La tabla de verdad de la Figura 3.12*b* muestra que el flip-flop *JK* conmutará cuando ambas entradas de datos, *J* y *K*, estén en el nivel ALTO, siempre que un pulso de reloj llegue a la entrada *CK*. La acción de conmutación real tiene lugar cuando el pulso de reloj cambia del nivel ALTO al BAJO, como muestra la flecha (↓) en la tabla de verdad.



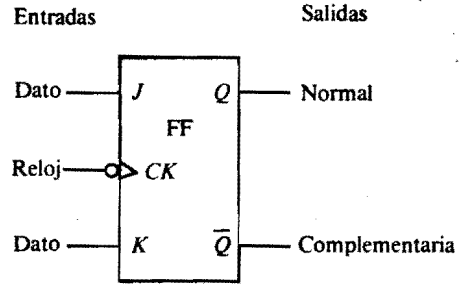
(a) Símbolo lógico para el cerrojo transparente de 4 bits

Modo de operación	Entradas		Salidas
	D	*E*	*Q*
Dato habilitado	0	1	0
	1	1	1
Dato encerrado	X	0	Igual que antes

0 = BAJO
1 = ALTO
X = irrelevante

(b) Tabla de verdad para el cerrojo

Figura 3.11.



(a) Símbolo lógico para el flip-flop *JK*

Modo de operación	Entradas			Salidas	
	J	*K*	*CK*	*Q*	$\overline{Q}$
Conmutación	1	1	↓	Estado opuesto	
Set	1	0	↓	1	0
Reinicialización	0	1	↓	0	1
Mantenimiento	0	0	↓	No cambia	

0 = BAJO
1 = ALTO
↓ = transición ALTA-BAJA del pulso de reloj

(b) Tabla de verdad para el flip-flop *JK*

Figura 3.12.

El flip-flop *JK* está en el *modo set* cuando las entradas de datos son: $J = 1$ y $K = 0$. La segunda línea de la tabla de verdad de la Figura 3.12*b* muestra que la transición de ALTO a BAJO del pulso de reloj pone la salida Q a 1. El *modo de reinicialización* (poner Q a 0) se muestra en la tercera línea de la tabla de verdad. La última línea de la tabla de verdad de la Figura 3.12*b* describe el *modo de mantenimiento* (modo de no hacer nada) del flip-flop *JK*. Cuando ambas entradas de datos (J y K) son BAJAS, un pulso de reloj en la entrada *CK* no tendrá efecto sobre las salidas.

El disparo es una característica importante de los flip-flops. Los flip-flops pueden clasificarse en dispositivos *disparados por flanco* o dispositivos *disparados por nivel*. Los símbolos lógicos de las Figuras 3.10*a* y 3.12*a* sugieren que estos flip-flops son disparados por flanco, por el pequeño > (mayor que) que está junto a la entrada de reloj. El flip-flop *D* es disparado en la transición L a H (BAJA a ALTA) del pulso de reloj. Esto se muestra en la tabla de verdad y en el símbolo lógico (la no existencia de circulito en la entrada *CK* implica que es necesario un 1 para activar el reloj). El flip-flop *D* también se denomina flip-flop *disparado por flanco positivo* ya que es la transición positiva del pulso de reloj la que dispara al flip-flop.

El flip-flop *JK* es una flip-flop *disparado por flanco negativo*. Esto se muestra en la tabla de verdad y en el símbolo lógico de la Figura 3.12. El circulito en la entrada *CK*, del símbolo lógico del flip-flop *JK*, implica que es necesario un nivel BAJO para activar el reloj. Debido a que el flip-flop *JK* es disparado por flanco, realmente es en la transición H a L (ALTA a BAJA) (negativa) del pulso de reloj cuando se dispara el flip-flop.

El cerrojo transparente de 4 bits de la Figura 3.11 es un dispositivo disparado por nivel. Esto significa que cuando la entrada de habilitación (análoga a las entradas *CK* de los flip-flops) está en ALTA, cualquier dato binario en las entradas (D_3–D_0) aparecerá inmediatamente en las salidas (Q_3–Q_0). Por esta razón se dice que el cerrojo es transparente.

PROBLEMAS RESUELTOS

3.12. Los circuitos lógicos secuenciales contienen dispositivos llamados _____.

Solución:

Los circuitos lógicos secuenciales contienen flip-flops.

3.13. Una característica de memoria utilizada para almacenar datos es típica de _____ (flip-flops, puertas).

Solución:

Los flip-flops tienen una característica de memoria y pueden utilizarse para almacenar datos.

3.14. El flip-flop *D* también se denomina flip-flop _____.

Solución:

El flip-flop *D* también se denomina flip-flop de datos. A veces se llama flip-flop de retardo.

3.15. La salida normal de un flip-flop es la salida _____ (Q, $\overline{Q}$).

Solución:

La salida normal de un flip-flop es la salida Q). La salida $\overline{Q}$ es la invertida, o complementaria.

3.16. Listar los tres modos de operación del flip-flop *D* mostrado en la Figura 3.10.

Solución:

Acudir a la tabla de verdad de la Figura 3.10*b*. El flip-flop *D* puede operar en los modos de set, reinicialización o mantenimiento.

3.17. Listar el modo de operación del flip-flop *D* durante cada uno de los cinco pulsos de reloj de la Figura 3.13.

Solución:

Acudir a la tabla de verdad de la Figura 3.10*b*. Según esta tabla, los modos del flip-flop *D* durante cada pulso de la Figura 3.13 son los siguientes:
pulso *a* = modo set pulso *d* = modo set
pulso *b* = modo reset pulso *e* = modo reset
período de tiempo *c* (no pulso) = modo mantenimiento

3.18. Listar los valores binarios de la salida normal (*Q*) del flip-flop *D* de la Figura 3.13 después de cada uno de los pulsos de reloj.

Solución:

Acudir a la tabla de verdad de la Figura 3.10*b*. Las salidas binarias en *Q* del flip-flop *D* de la Figura 3.13 son:
después del pulso *a* = 1 después del pulso *d* = 1
después del pulso *b* = 0 después del pulso *e* = 0
período de tiempo *c* (no pulso) = 0 (igual que después del pulso *b*)

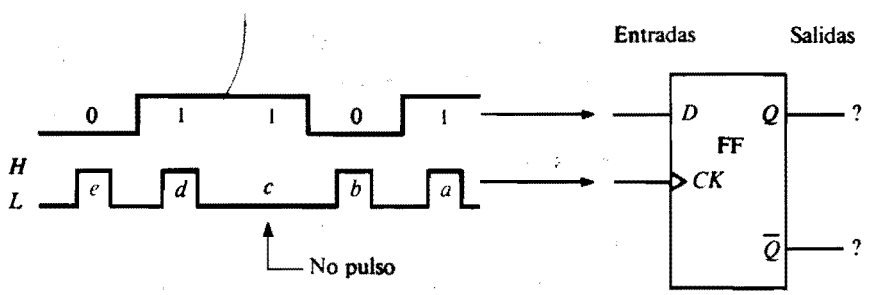


Figura 3.13. Problema del flip-flop *D*.

3.19. Un cerrojo es un dispositivo de _____ (cuenta, almacenamiento).

Solución:

La función principal de un cerrojo es la de almacenar datos.

3.20. Acudir a la Figura 3.11. La entrada de habilitación de este cerrojo de 4 bits es activada por un _____ (0, 1) lógico.

Solución:

Acudir a la tabla de verdad de la Figura 3.11*b*. De acuerdo con esta tabla, la entrada de habilitación de este cerrojo de 4 bits se activa por un 1 lógico.

3.21. Listar los cuatro modos de operación del flip-flop *JK* de la Figura 3.12.

Solución:

Acudir a la tabla de verdad de la Figura 3.12*b*. Los modos de operación del flip-flop *JK* son conmutación, set, reinicialización y mantenimiento.

3.22. Listar el modo de operación del flip-flop *JK* durante cada uno de los pulsos de la Figura 3.14.

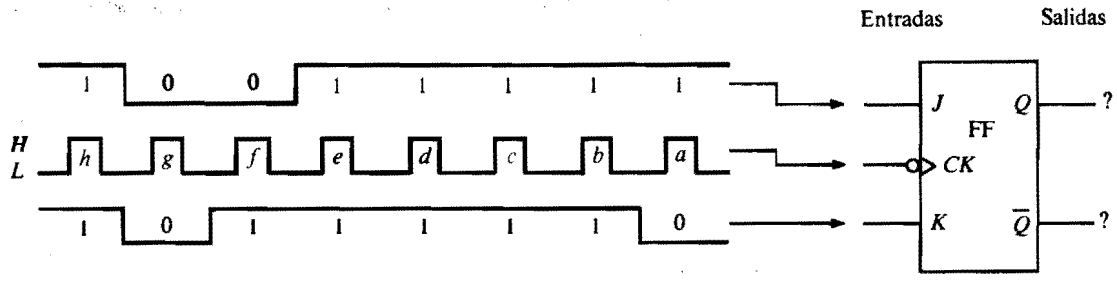


Figura 3.14. Problema del flip-flop *JK*.

Solución:

Acudir a la tabla de verdad de la Figura 3.12*b*. De acuerdo con esta tabla los modos del flip-flop *JK* durante cada pulso de la Figura 3.14 son:

pulso *a* = set	pulso *e* = conmutación
pulso *b* = conmutación	pulso *f* = reset
pulso *c* = conmutación	pulso *g* = mantenimiento
pulso *d* = conmutación	pulso *h* = conmutación

3.23. Listar los valores binarios de la salida normal *Q* del flip-flop *JK* de la Figura 3.14 *después* de cada pulso de reloj.

Solución:

Acudir a la tabla de verdad de la Figura 3.12*b*. Las salidas binarias en *Q* del flip-flop *JK* de la Figura 3.14 son las siguientes:

después del pulso *a* = se pone 1	después del pulso *e* = conmuta a 1
después del pulso *b* = conmuta a 0	después del pulso *f* = se pone a 0
después del pulso *c* = conmuta a 1	después del pulso *g* = se mantiene en 0
después del pulso *d* = conmuta a 0	después del pulso *h* = conmuta a 1

3.24. Acudir a la Figura 3.10. Este flip-flop *D* es un dispositivo disparado por _____ (flanco, nivel).

Solución:

El flip-flop *D* de la Figura 3.10 es un dispositivo disparado por flanco. Esto se muestra en el diagrama lógico por el símbolo >. También se muestra en la tabla de verdad por el símbolo ↑, indicando que el flip-flop se dispara en la transición H a L del pulso de reloj.

3.25. Acudir a la Figura 3.11. Este cerrojo de 4 bits es un dispositivo disparado por _____ (flanco, nivel).

Solución:

El cerrojo transparente de la Figura 3.11 es un dispositivo disparado por nivel.

3.26. Acudir a la Figura 3.12. Este flip-flop *JK* es un dispositivo disparado por flanco _____ (negativo, positivo).

Solución:

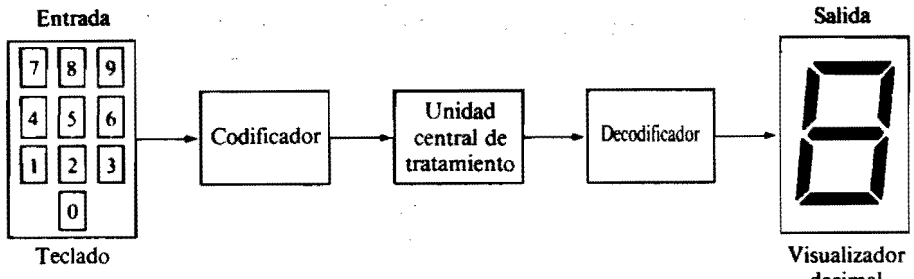
El flip-flop de la Figura 3.12 es un dispositivo disparado por flanco negativo ya que se dispara en la transición H a L (negativa) del pulso de reloj.

3.4. CODIFICADORES, DECODIFICADORES Y VISUALIZADORES DE SIETE SEGMENTOS

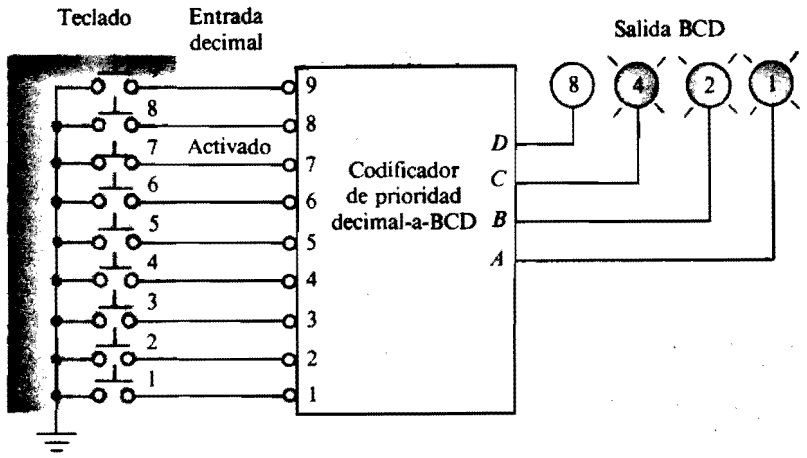
Considerar el sencillo diagrama de bloques de la calculadora de la Figura 3.15*a*. En este sistema digital la entrada decimal del teclado debe traducirse a forma decimal codificada en binario (BCD). Este proceso lo realiza un dispositivo digital llamado *codificador*. La traducción de decimal a BCD se denomina codificación. La salida de la unidad central de tratamiento (CPU) está en forma decimal codificada en binario. El decodificador traduce el BCD al código especial del visualizador de siete segmentos. Para el usuario, el decodificador está traduciendo de BCD a decimal.

Un diagrama lógico de un *codificador de prioridad decimal a BCD* se muestra en la Figura 3.15*b*. El codificador tiene 9 entradas *activas en BAJA* y 4 salidas conectadas a lámparas indicadoras. El cableado del teclado se muestra en la parte izquierda, cada tecla numerada está conectada a su entrada correspondiente del codificador. El ejemplo de la Figura 3.15*b* muestra que al pulsar la tecla 7, se *activa* (pone a tierra) la entrada 7 del codificador. Esta entrada produce la salida BCD 0111, como muestran los visualizadores de la Figura 3.15*b*. La mayoría de los codificadores tienen una característica de *prioridad*. Esto significa que si se pulsan dos teclas al mismo tiempo, la que tiene el valor decimal más alto activará las salidas. Es claro que las conexiones de alimentación son necesarias para completar el cableado del codificador de la Figura 3.15*b*. El codificador puede comprarse en forma de CI, pero también puede ser construido a partir de puertas lógicas individuales. Probablemente se necesitarán entre 10 y 20 puertas lógicas para implementar este circuito.

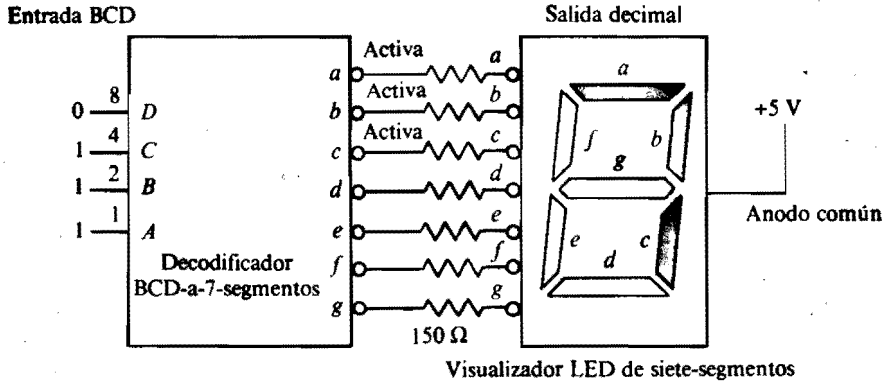
Considerar el circuito decodificador-visualizador detallado en la Figura 3.15*c*. El *decodificador BCD a siete segmentos* traduce el 0111_{BCD} a su equivalente 7 en el visualizador de siete segmentos. El visualizador utilizado en este circuito se denomina de ánodo común porque los siete LEDs (que forman los siete segmentos) tienen sus ánodos conectados a la fuente de alimentación de +5 V. El visualizador LED de siete segmentos de la Figura 3.15*c* tiene las entradas *activas en BAJA*, como muestran los circuitos en las entradas de la *a* a la *g*. Es necesario un nivel BAJO para activar un segmento del visualizador. Observar también que el decodificador tiene las salidas activas compatibles en BAJA. Los resistores de 150 Ω de cada línea, entre el decodificador y el visualizador, son de limitación para mantener las corrientes a niveles seguros. El ejemplo de la Figura 3.15*c* muestra que solamente las salidas del decodificador *a*, *b* y *c* están activadas (en 0 lógico). Las otras salidas (*d*, *e*, *f* y *g*) permanecen a 1 lógico. El visualizador de la Figura 3.15*c*



(a) Diagrama de bloques simplificado de una calculadora



(b) Diagrama lógico de los circuitos codificadores del teclado



(c) Diagrama lógico de los circuitos del decodificador-visualizador

Figura 3.15.

se sólo una conexión de alimentación de +5 V, mientras que el decodificador debe tener dos conexiones de alimentación. Las conexiones de alimentación del decodificador se han omitido en el diagrama. Los decodificadores reales BCD a siete segmentos también contienen entradas para *apagar* (desconectar todos los segmentos) y para *test de lámparas* (conectar todos los segmentos).

PROBLEMAS RESUELTOS

3.27. El codificador de la Figura 3.15*b* traduce un teclado decimal a la forma _____ (ASCII, BCD).

Solución:

Acudir a la Figura 3.15*b*. Este codificador traduce las pulsaciones de las teclas decimales a forma BCD.

3.28. El codificador de la Figura 3.15*b* tiene entradas activas en _____ (ALTA, BAJA).

Solución:

Acudir a la Figura 3.15*b*. Este codificador tiene entradas activas en BAJA. Están representadas en el diagrama lógico por los circulitos en las entradas del codificador.

3.29. El decodificador de la Figura 3.16 traduce del decimal codificado binario a _____ (decimal, hexadecimal).

Solución:

El decodificador en la Figura 3.16 traduce del BCD a la representación decimal en el visualizador de siete segmentos.

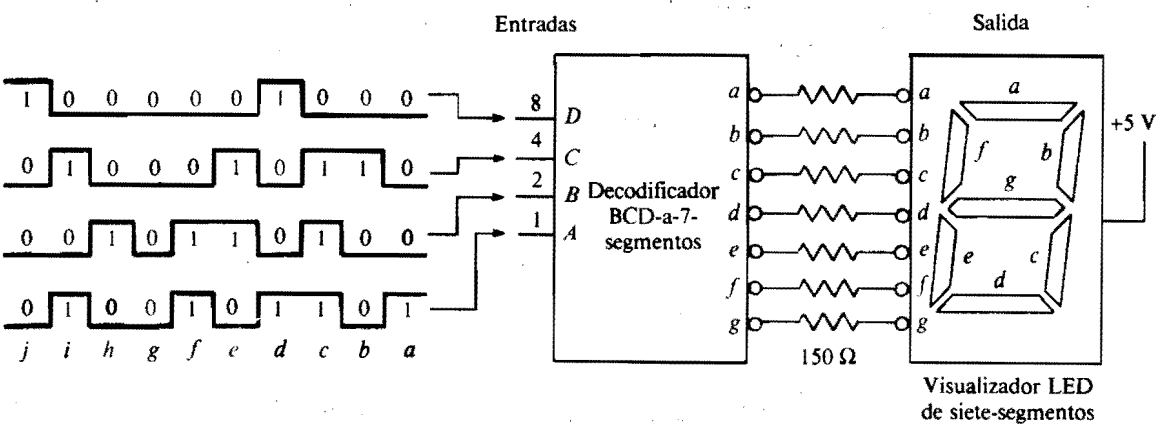


Figura 3.16. Problema del decodificador BCD-a-siete-segmentos.

3.30. El decodificador de la Figura 3.16 tiene las entradas activas en _____ (ALTA, BAJA) y las salidas activas en _____ (ALTA, BAJA).

Solución:

El decodificador de la Figura 3.16 tiene las entradas activas en ALTA y las salidas en BAJA.

3.31. Listar las indicaciones *decimales* del visualizador de siete segmentos para cada pulso de entrada en la Figura 3.16.

Solución:

Las salidas decimales para los diversos pulsos de la Figura 3.16 son las siguientes:

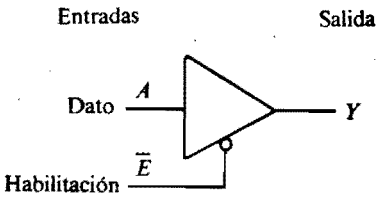
pulso $a = 1$ pulso $f = 3$
pulso $b = 4$ pulso $g = 0$
pulso $c = 7$ pulso $h = 2$
pulso $d = 9$ pulso $i = 5$
pulso $e = 6$ pulso $j = 8$

3.5. BUFFERS Y DISPOSITIVOS DE TRES ESTADOS

La circuitería electrónica que puede traducir señales de una sección de un sistema a otra se denomina *interfaz*. En un sistema basado en microprocesador, las interfaces pueden ser buffers, multiplexores, conversores analógico-digitales (ADC), conversores digitales-analógicos (DAC), excitadores de transistores discretos y aisladores (ópticos o relés).

En esta sección nos concentraremos en un dispositivo de interfaz: el buffer. Los fabricantes han diseñado diversos circuitos integrados especiales denominados *buffers* que se utilizan como interfaz en situaciones especiales. El símbolo lógico para un *buffer no inversor de bus* se ilustra en la Figura 3.17*a*. La operación del buffer del bus se detalla en la tabla de verdad de la Figura 3.17*b*. El buffer tiene una entrada de habilitación activa en BAJA, y el dato no se invierte cuando pasa a través del buffer. Cuando el buffer del bus está inhabilitado, la salida está «flotando» y no tiene efecto en la línea del bus. En la condición de inhabilitación, la salida de la puerta no absorbe ni da ninguna corriente a la línea del bus. El buffer del bus de la Figura 3.17 se dice que tiene *salida de tres estados* (o *tristate*).

Las salidas de los dispositivos lógicos se definen habitualmente como ALTA o BAJA. Por esta razón, las salidas de dos dispositivos lógicos no pueden conectarse entre sí ya que si una está en ALTA y la otra en BAJA, entonces la salida estará indefinida y los CI pueden sufrir daños. Los fabricantes han desarrollado CI especiales con salidas de tres estados para utilizarlas cuando las



(a) Símbolo lógico para el buffer del bus de tres estados

Modo de operación	Entradas		Salida
	$\overline{E}$	A	Y
Habilitado	0	0	0
	0	1	1
Inhabilitado	1	0	Alta impedancia (tensión de salida flota)
	1	1	

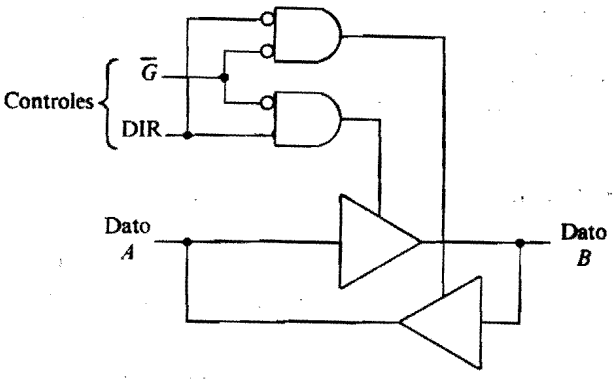
(b) Tabla de verdad para el buffer del bus de tres

Figura 3.17.

salidas de distintos CI deban compartir un camino de salida o bus común. Los dispositivos de tres estados tienen estados de salida de ALTA, BAJA y un *estado especial de alta impedancia (estado de alta-Z)*. Cuando el dispositivo está en su estado de alta-Z, su salida está desconectada o aislada efectivamente del bus.

El diagrama lógico para un transceptor de bus está esbozado en la Figura 3.18*a*. Este *transceptor de bus de tres estados* es un buffer no inversor que puede enviar datos en cualquier dirección. Entre la RAM y el bus de datos y también entre el microprocesador y el bus de datos de un sistema microcomputador es necesario un buffer de dos direcciones. En la práctica, los transceptores son construidos en las RAM y microprocesadores.

La operación del transceptor de un bus de tres estados se detalla en la tabla de verdad de la Figura 3.18*b*. El transceptor tiene dos entradas de control ($\overline{G}$ y DIR). La entrada DIR controla la *dirección* del flujo de datos. Un nivel ALTO en la entrada DIR significa que el terminal *A* es la entrada de datos y el terminal *B* es la salida al bus. Un nivel BAJO en la entrada DIR significa que el terminal *B* es la entrada de datos y el terminal *A* es la salida al bus. Un nivel BAJO en la entrada $\overline{G}$ activa los buffers y permite que pasen los datos. Sin embargo, un nivel ALTO en la entrada $\overline{G}$ inhabilita los buffers quedando los terminales *A* y *B* aislados entre sí.



(a) Diagrama lógico para el transceptor del bus de tres estados

Entradas de control		Operación
Habilitación	Dirección	
$\overline{G}$	DIR	
L	L	Dato *B* a bus *A*
L	H	Dato *A* a bus *B*
H	X	Aislamiento (alta *Z*)

H = nivel lógico ALTO
L = nivel lógico BAJO
X = irrelevante

(b) Tabla de verdad para el transceptor de bus de tres estados

Figura 3.18.

PROBLEMAS RESUELTOS

3.32. La salida de un dispositivo lógico de tres estados puede ser 1 lógico, 0 lógico o el estado de alta _____ (tensión, impedancia).

Solución:

La salida de un dispositivo lógico de tres estados puede ser ALTA, BAJA, o un estado de alta impedancia.

3.33. Acudir a la Figura 3.18. El transceptor es un buffer del bus de tres estados que puede pasar datos en _____ (ambas, una) dirección.

Solución:

El transceptor de la Figura 3.18 puede pasar datos en cualquier dirección (de *A* a *B* o de *B* a *A*) dependiendo del nivel lógico en el control DIR.

3.34. Acudir a la Figura 3.18. Si el control de habilitación del transceptor ($\overline{G}$) está en ALTA, ¿pasarán los datos por los buffers?

Solución:

No. Con el control de habilitación del transceptor en ALTA, los terminales de datos *A* y *B* están aislados y ningún dato puede pasar en ninguna dirección.

3.35. La puerta del buffer del bus de la Figura 3.19 tiene una entrada de habilitación activa en _____ (ALTA, BAJA).

Solución:

El buffer del bus de la Figura 3.19 tiene una entrada de habilitación activa en BAJA.

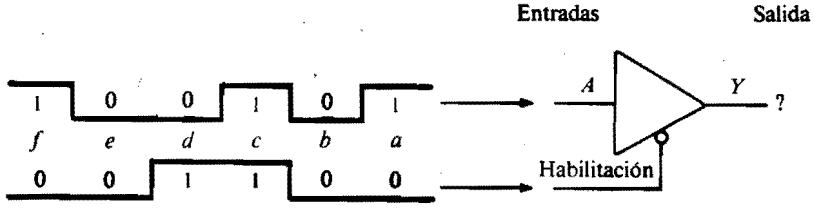


Figura 3.19 Problema del buffer del bus de tres estados.

3.36. Acudir a la Figura 3.19. Listar las salidas de la puerta del buffer del bus durante cada uno de los seis pulsos de entrada.

Solución:

Las salidas de la puerta del buffer del bus de la Figura 3.19 durante cada uno de los seis pulsos de entrada son:

durante el pulso $a = 1$ (puerta habilitada)

durante el pulso $b = 0$ (puerta habilitada)

durante el pulso c = estado de alta Z (puerta inhabilitada)

durante el pulso d = estado de alta Z (puerta inhabilitada)

durante el pulso $e = 0$ (puerta habilitada)

durante el pulso $f = 1$ (puerta habilitada)

3.6. MEMORIAS SEMICONDUCTORAS

El flip-flop, o cerrojo, es la *celda básica de memoria* utilizada en muchas memorias semiconductoras. Las memorias semiconductoras habitualmente se dividen en dos grupos: *memorias de lectura/escritura* y *memorias de sólo lectura* (ROM). Los nombres implican las diferencias entre

los dos tipos de memoria. Una ROM es una memoria no volátil con su patrón binario de ceros y unos programados *permanentemente* por el fabricante. La memoria de lectura/escritura es una memoria que puede ser fácilmente programada, borrada y reprogramada por el usuario. La programación se denomina *escribir* en memoria. *Copiar datos de la memoria,* sin destruir el contenido, se denomina *leer* de memoria. La memoria de lectura/escritura se denomina *RAM (memoria de acceso aleatorio),* con frecuencia. Generalmente, el programa de la RAM es *volátil,* lo cual significa que se perderá si se desconecta la alimentación del CI, incluso durante un instante. Los sistemas microcomputadores normalmente contienen los dos tipos de memorias semiconductoras ROM y RAM. Las posiciones de almacenamiento ROM y RAM están habitualmente en CI separados.

Existen actualmente cuatro posibilidades de memorias de sólo lectura. La ROM estándar es programada por el fabricante. La PROM *(memoria de sólo lectura programable)* puede ser programada permanentemente por el usuario o distribuidor utilizando un equipo especial. Puede ser programada solamente una vez. La EPROM *(memoria de sólo lectura programable borrable)* puede ser programada y borrada por el usuario. Los datos almacenados en la EPROM pueden ser borrados aplicando luz ultravioleta de alta *intensidad,* a través de una ventana especial transparente, en la parte superior del CI. Otra PROM borrable es la EAROM *(memoria de sólo lectura alterable eléctricamente)* que puede ser borrada y programada por el usuario con un equipo especial. La EAROM se borra eléctricamente y no con luz ultravioleta. La ROM, PROM, EPROM y EAROM son consideradas memorias *no volátiles* permanentes y no pierden sus datos cuando se desconecta la alimentación del CI.

Las memorias RAM se subdividen en dos grupos. Si la memoria de lectura/escritura contiene circuitos tipo «flip-flop» como celdas de memoria, se denomina *RAM estática* (SRAM). Una forma más simple de la celda de memoria de lectura/escritura es la base de la *RAM dinámica* (DRAM). Debido a que la celda de memoria de la RAM dinámica está basada en una capacitancia, las celdas deben ser refrescadas cientos de veces por segundo. La RAM estática no necesita refrescarse y mantiene su información binaria, indefinidamente, todo el tiempo que esté alimentado el CI. Las RAM dinámicas tienen mayor capacidad y menos consumo de potencia que las estáticas.

Un dispositivo de memoria semiconductora más moderno es la *RAM no volátil* (NVRAM). La NVRAM combina RAM estáticas y PROM borrables eléctricamente (EEPROM) de idéntica capacidad. Los datos pueden ser transferidos entre la SRAM y la EEPROM y viceversa mediante operaciones de almacenamiento y rellamada. Los datos pueden almacenarse con seguridad en la EEPROM no volátil cuando se desconecte la alimentación. Cuando se conecta la alimentación, los datos de la EEPROM son rellamados automáticamente a la sección SRAM de la RAM no volátil. El concepto de RAM no volátil también está implementado por los fabricantes de microcomputadoras utilizando pastillas comunes de RAM con una batería de seguridad.

En microcomputadoras, las RAM se utilizan para almacenamiento temporal de programas y datos del usuario. Las ROM son utilizadas más frecuentemente para almacenar instrucciones del lenguaje máquina que pueden considerarse el *programa monitor.* El programa monitor puede contener rutinas de inicialización no modificables, rutinas de entrada/salida y algoritmos aritméticos.

PROBLEMAS RESUELTOS

3.37 En la práctica, la memoria de lectura/escritura se denomina _____ (RAM, ROM).

Solución:

En la práctica, la memoria de lectura/escritura se denomina RAM.

3.38. Las letras RAM significan _____ _____ _____ en electrónica digital.

Solución:

Las letras RAM significan memoria de acceso aleatorio. En la práctica las RAM semiconductoras también son memorias de lectura/escritura.

3.39. Las letras PROM significan _____ _____ _____ _____.

Solución:

Las letras PROM significan memoria programable de sólo lectura.

3.40. La ROM y la _____ (RAM, PROM) son dispositivos de almacenamiento permanente no volátiles.

Solución:

La ROM y la PROM son dispositivos de almacenamiento permanente no volátiles. La ROM es programada por el fabricante mientras que la PROM es programada por el usuario.

3.41. La RAM es un dispositivo de almacenamiento _____ (no volátil, volátil).

Solución:

La RAM es un dispositivo de almacenamiento volátil.

3.42. Las letras EPROM significan _____ _____ _____ _____ _____ en electrónica digital.

Solución:

Las letras EPROM significan memoria de sólo lectura programable borrable.

3.43. Las letras EAROM significan _____ _____ _____ _____ _____.

Solución:

Las letras EAROM significan memoria de sólo lectura alterable eléctricamente.

3.44. La EPROM se considera un dispositivo de almacenamiento _____ (temporal, permanente).

Solución:

La EPROM se considera un dispositivo de almacenamiento permanente. Puede ser borrada y reprogramada pero solamente con equipo especial.

3.45. Las memorias de lectura/escritura muchas veces se denominan RAM estáticas o _____.

Solución:

Las RWM muchas veces se clasifican como RAM estáticas o dinámicas.

3.46. La RAM _____ (dinámica, estática) utiliza como celdas de memoria circuitos basados en flip-flops.

Solución:

La RAM estática utiliza como celdas de memoria circuitos basados en flip-flops.

3.47. La RAM _____ (dinámica, estática) necesita que sus posiciones de almacenamiento sean *refrescadas* cientos de veces por segundo.

Solución:

La RAM dinámica necesita que sus posiciones de almacenamiento sean refrescadas cientos de veces por segundo.

3.48. En una microcomputadora, una PROM puede ser utilizada para que contenga un programa _____ (monitor, de usuario temporalmente).

Solución:

En una microcomputadora, una PROM puede utilizarse para que contenga un programa monitor.

3.49. Las letras NVRAM significan _____ .

Solución:

Las letras NVRAM significan RAM no volátil.

3.50. Las más modernas NVRAM contienen RAM estáticas y _____ (DRAM, EEPROM).

Solución:

Las NVRAM contienen SRAM y EEPROM de idéntica capacidad.

3.7. UTILIZACION DE LAS RAM Y ROM

La organización de una RAM o ROM puede considerarse similar a la de una tabla de verdad. La Figura 3.20*a* muestra una posible organización de las posiciones de memoria. Esta tabla representa la organización de una RAM de 16 × 4 bits e indica que los datos de la memoria están organizados en 16 grupos de 4 bits denominados palabras en esta tabla. En la Figura 3.20*a*, la mayoría de las cajas (celdas de memoria) están vacías excepto la palabra 12 que contiene almacenado el dato 0101. En la práctica, cada celda «vacía» contiene algún patrón de ceros y unos.

Un símbolo lógico para una RAM de 16 × 4 bits se da en la Figura 3.20*b*. En el diagrama, la RAM de 64 bits está en el proceso de *escribir* el dato de entrada 0101 en la posición de memoria 12_{10} (1100_2). Las entradas de datos contienen el dato que se va a almacenar (0101_2). La posición de la palabra 12_{10} se localiza inicializando las entradas de dirección con 1100_2 (12_{10}). A continuación, los dos controles ($\overline{CS}$ y $\overline{WE}$) colocan a la RAM en el *modo de escritura*. Observar que las entradas de *habilitación de escritura* ($\overline{WE}$) y de *selección de pastilla* ($\overline{CS}$) deben estar en BAJA durante la operación de escritura. El 0101, se coloca entonces en la palabra de posición 12_{10}, como muestra la Figura 3.20*a*. Algunos fabricantes denominan a la entrada de control $\overline{CS}$ habilitación de memoria.

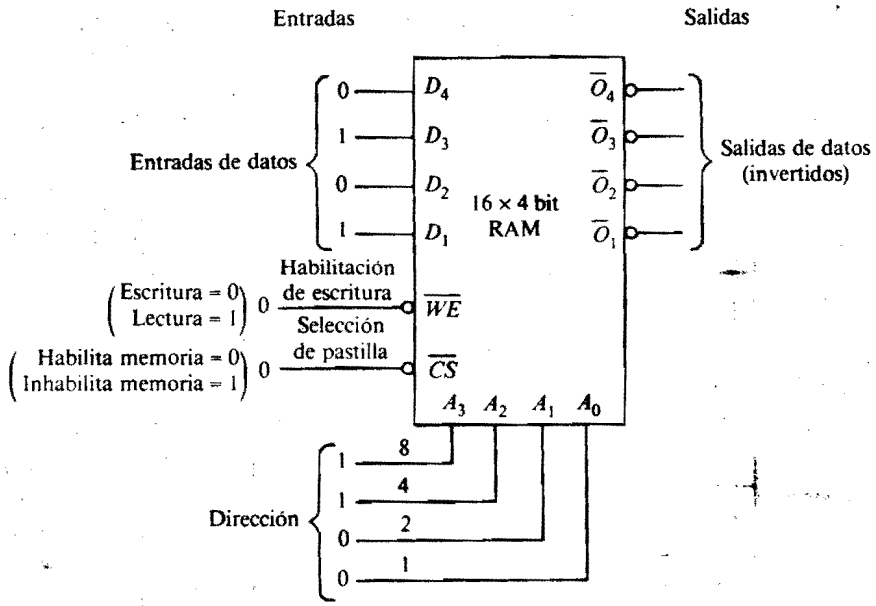
La tabla de verdad para la RAM de 64 bits está detallada en la Figura 3.20*c*. Esta RAM está en el *modo de escritura* cuando ambos controles ($\overline{WE}$ y $\overline{CS}$) están en BAJA. Durante la operación de

escritura, los 4 bits de datos (D_4, D_3, D_2, D_1) son cargados en la posición de memoria señalada por la dirección. Durante la operación de escritura, las salidas ($\overline{O}_4$, $\overline{O}_3$, $\overline{O}_2$, $\overline{O}_1$) flotan en ALTA. Con las entradas de control $\overline{CS} = 0$ y $\overline{WE} = 1$, la RAM está en el *modo de lectura*. Durante el modo de lectura, el complemento de la palabra del dato señalada por las entradas de dirección aparece en

Dirección	Bit *D*	Bit *C*	Bit *B*	Bit *A*
Palabra 0				
Palabra 1				
Palabra 2				
Palabra 3				
Palabra 4				
Palabra 5				
Palabra 6				
Palabra 7				
Palabra 8				
Palabra 9				
Palabra 10				
Palabra 11				
Palabra 12	0	1	0	1
Palabra 13				
Palabra 14				
Palabra 15				

(a) Organización de una RAM de 16 × 4 bits

Figura 3.20.



(b) Símbolo lógico para una RAM de 16 × 4 bits

Modo de operación	Entradas de control		Salidas
	$\overline{CS}$	$\overline{WE}$	$\overline{Q}$
Escritura	0	0	Estado lógico 1
Lectura	0	1	Complemento del dato almacenado en memoria
Mantenimiento	1	X	Estado lógico 1

X = irrelevante

(c) Tabla de verdad para una RAM de 64 bits

Figura 3.20 *(continuación).*

la salida. Los datos almacenados en la RAM no son destruidos por la operación de lectura. En el *modo de mantenimiento* (*hold*) las salidas van todas al nivel ALTO y no entra ningún dato procedente de las cuatro entradas *D*.

La RAM de 64 bits de la Figura 3.20 es una pastilla de memoria de capacidad extremadamente pequeña y se cita por su simplicidad. Sin embargo, esta RAM estática de 16 × 4 bits puede ser comprada todavía como un circuito integrado TTL (lógica transistor-transistor) (CI 7489).

En la pasada década, se realizaron grandes avances para producir memorias semiconductoras de gran capacidad a bajo costo. Las RAM dinámicas comunes pueden ser de 16.384 × 1; 65.536 × 1; 131.072 × 1; 1.048.576 × 1; 16.384 × 4; 65.536 × 4, y 262.144 × 4 bits. También existen en el mercado módulos de RAM dinámicas incluso con capacidades mayores.

Un CI ROM opera de forma similar a la RAM cuando está en los modos de *lectura* y *mantenimiento* (*hold*). Una ROM comparable a la de la Figura 3.20 puede tener todas las entradas y salidas de la RAM *excepto las entradas de datos* y *de habilitación de escritura*. Una palabra es *leída* de la ROM direccionando una posición específica y activando después la entrada de *selección de pastilla* ($\overline{CS}$) con un nivel BAJO.

Algunas organizaciones típicas y capacidades de CI ROM son: 512 × 4; 2048 × 8; 4096 × 8; 8192 × 8; 16.384 × 8; 65.536 × 8, y 131.072 × 8 bits. Algunos módulos EEPROM tienen incluso capacidades mayores.

PROBLEMAS RESUELTOS

3.51. La RAM de 64 bits de la Figura 3.21 está en el modo de _____ (mantenimiento, escritura) durante el pulso *a*, y todas las salidas están flotando en _____ (ALTA, BAJA).

Solución:

Durante el pulso *a* en la Figura 3.21, ambos controles ($\overline{CS}$ y $\overline{WE}$) están en ALTA. La tabla de selección de modo de la Figura 3.20*c* indica que éste es el modo de mantenimiento para la RAM y las salidas están flotando en ALTA.

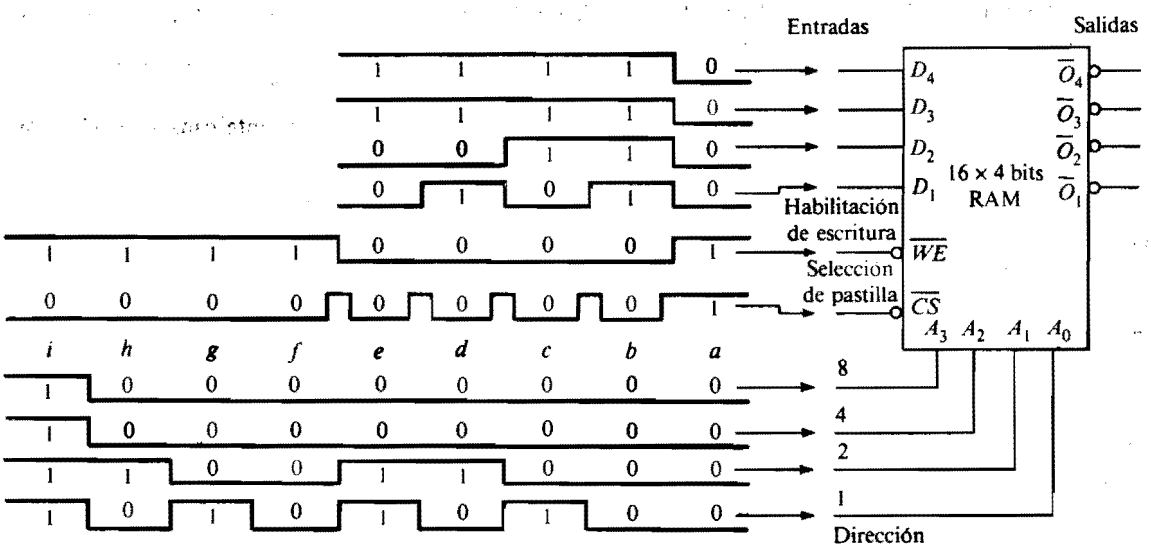


Figura 3.21. Problema de la RAM estática.

3.52. La RAM de la Figura 3.21 está en el modo de _____ (lectura, escritura) durante los pulsos *b, c, d* y *e.*

Solución:

Durante los pulsos *b, c, d* y *e* en la Figura 3.21, ambos controles ($\overline{CS}$ y $\overline{WE}$) están en BAJA. La tabla de selección de modo de la Figura 3.20*c* indica que éste es el modo de escritura para la RAM. Los datos de las entradas *D* se copian en la posición de memoria direccionada por las entradas *A* durante estos cuatro pulsos.

3.53. La RAM de la Figura 3.21 está en el modo de _____ (lectura, escritura) durante los pulsos *f, g, h* e *i.*

Solución:

Durante los pulsos *f, g, h* e *i* en la Figura 3.21, $\overline{CS} = 0$ y $\overline{WE} = 1$. La tabla de selección de modo de la Figura 3.20*c* indica que éste es el modo de lectura para la RAM. Los datos almacenados en la posición de memoria señalada por las entradas de dirección aparecerán en las salidas en forma invertida. La operación de lectura no destruye los contenidos de la RAM.

3.54. Acudir a la Figura 3.21. Listar la dirección y los datos almacenados en la RAM durante la operación de escritura en los pulsos *b, c, d* y *e.*

Solución:

Las posiciones de almacenamiento (direcciones) y los datos almacenados durante cada operación de escritura son los siguientes:

pulso *b* = dirección = 0000; dato almacenado = 1111
pulso *c* = dirección = 0001; dato almacenado = 1110
pulso *d* = dirección = 0010; dato almacenado = 1101
pulso *e* = dirección = 0011; dato almacenado = 1100

3.55. Acudir a la Figura 3.21. Listar la dirección y los datos que aparecen en las salidas de la RAM durante la operación de lectura en los pulsos *f, g, h* e *i.*

Solución:

La posición de almacenamiento (dirección) y los datos de salida durante cada operación de lectura son:

pulso *f* = dirección = 0000
dato almacenado = 1111
dato de salida (invertido) = 0000
pulso *g* = dirección = 0001
dato almacenado = 1110
dato de salida (invertido) = 0001
pulso *h* = dirección = 0010
dato almacenado = 1101
dato de salida (invertido) = 0010
pulso *i* = dirección = 1111
dato almacenado = no había dato almacenado en esta dirección
dato de salida = desconocido (la salida será impredictible)

3.56. Si la RAM de la Figura 3.21 fuese una ROM, ¿qué entrada(s) se omitiría(n) del diagrama lógico?

Solución:

La RAM de la Figura 3.21 puede convertirse en un símbolo lógico de ROM, omitiendo las entradas de datos y de habilitación de escritura. Una ROM la programa permanentemente el fabricante y no tiene modo de escritura.

PROBLEMAS SUPLEMENTARIOS

3.57. Si los flip-flops están interconectados para formar circuitos lógicos secuenciales, entonces las _____ _____ son utilizadas para formar circuitos lógicos combinacionales.

Res. puertas lógicas.

3.58. Una función lógica puede ser descrita de cuatro formas: por su nombre, su símbolo lógico gráfico, su tabla de verdad o su expresión _____.

Res. booleana.

3.59. Acudir a la Figura 3.22. Describir el tren de pulsos en la salida Y si este circuito lógico es una puerta OR de 2 entradas.

Res. pulso $a = 0$, pulso $b = 1$, pulso $c = 1$, pulso $d = 1$.

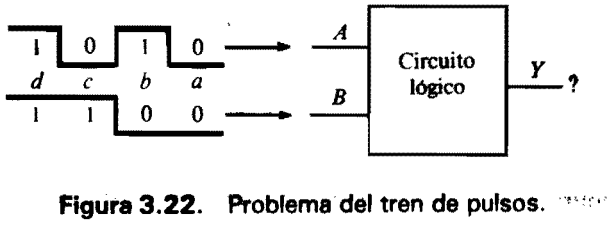


Figura 3.22. Problema del tren de pulsos.

3.60. Acudir a la Figura 3.22. Describir el tren de pulsos en la salida Y si este circuito lógico es una puerta NAND de 2 entradas.

Res. pulso $a = 1$, pulso $b = 1$, pulso $c = 1$, pulso $d = 0$.

3.61. Acudir a la Figura 3.22. Describir el tren de pulsos en la salida Y, si este circuito lógico es una puerta XOR de 2 entradas.

Res. pulso $a = 0$, pulso $b = 1$, pulso $c = 1$, pulso $d = 0$.

3.62. Los microprocesadores ¿qué cuatro instrucciones lógicas tienen en su conjunto de instrucciones?

Res. NOT (complementación o negación), AND, OR y XOR.

3.63. Escribir la expresión booleana para la tabla de verdad de la Figura 3.23.

Res. $\overline{C} \cdot B \cdot \overline{A} + C \cdot B \cdot A = Y$.

Entradas			Salida
C	*B*	*A*	*Y*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Figura 3.23.

3.64. Dibujar el diagrama lógico de un circuito que genere la tabla de verdad de la Figura 3.23.
Res. Ver Figura 3.24.

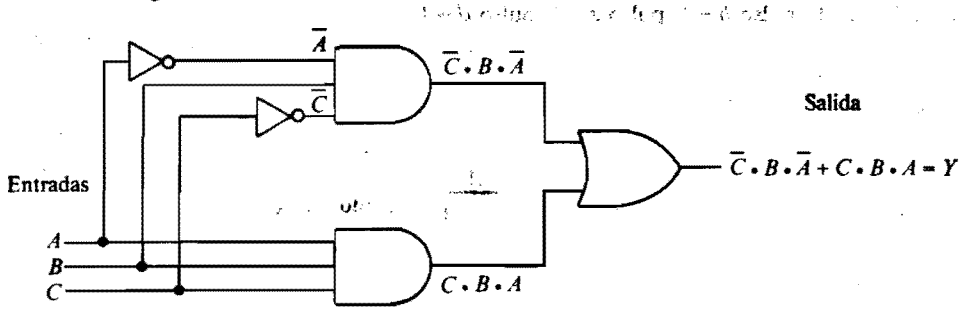


Figura 3.24. Solución del diagrama lógico AND-OR.

3.65. Poner en *set* un flip-flop significa cargar un 1 en la salida normal o _____ $(Q, \overline{Q})$.
Res. Q.

3.66. Poner en *reinicialización* un flip-flop significa cargar un 0 en la salida _____ $(Q, \overline{Q})$.
Res. Q.

3.67. Cuando a un flip-flop se denomina cerrojo, se utiliza como dispositivo de _____ (conteo, memoria).
Res. memoria.

3.68. Listar el modo de operación del flip-flop *JK* durante cada uno de los pulsos de reloj de la Figura 3.25.
Res. pulso a = reset; pulso b = conmuta; pulso c = conmuta; pulso d = set; pulso e = conmuta; pulso f = mantenimiento

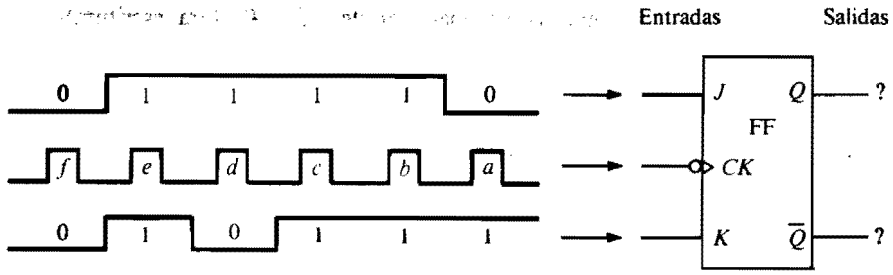


Figura 3.25. Problema del flip-flop *JK*.

3.69. Listar las salidas binarias en la salida normal (Q) del flip-flop *JK* de la Figura 3.25 *después* de cada uno de los pulsos de reloj.

Res. pulso $a = 0$ pulso $c = 0$ pulso $e = 0$
pulso $b = 1$ pulso $d = 1$ pulso $f = 0$

3.70. Los traductores de código electrónico normalmente se denominan _____ o _____.

Res. codificadores, decodificadores.

3.71. Un visualizador LED de siete segmentos puede ser excitado por un dispositivo llamado _____ a-siete-segmentos.

Res. decodificador **BCD.**

3.72. Acudir a la Figura 3.15*c*. Para que luzca el segmento *d* del LED, la entrada *d* del visualizador debe estar excitada en _____ (ALTA, BAJA) por el decodificador.

Res. BAJA.

3.73. Las letras TTL significan _____ _____ - _____ en electrónica digital.

Res. lógica transistor-transistor.

3.74. Cuando los dispositivos TTL o CMOS se conectan a las líneas del bus de la microcomputadora, las salidas deben ser del tipo de _____ _____.

Res. tres estados.

3.75. Cuando la salida de un dispositivo TTL de tres estados no está ni en 0 lógico ni en 1 lógico, pero se le permite que flote al nivel de tensión de la línea del bus, se dice que está en el estado de alta _____.

Res. impedancia (Z).

3.76. Listar los dos tipos de memorias semiconductoras normalmente utilizados en los sistemas microcomputadores.

Res. RAM y ROM.

3.77. Una ROM y una _____ (PROM, RAM) son muy similares, ya que ambas son tipos de memoria permanentes.

Res. PROM.

3.78. Colocar datos en memoria se conoce como operación de _____ (lectura, escritura).
Res. escritura.

3.79. Copiar datos de memoria se conoce como operación de _____ (lectura, escritura).
Res. lectura.

3.80. En la práctica, las letras usadas para denominar una memoria semiconductora de lectura/escritura son _____ (EPROM, RAM).
Res. RAM.

3.81. ¿Qué dos tipos básicos de CI RAM pueden adquirirse?
Res. estáticos y dinámicos.

3.82. Los programas de usuario pueden almacenarse temporalmente en la _____ (EPROM, RAM) de una microcomputadora.
Res. RAM.

3.83. ¿Cuál es la longitud de cada palabra de una RAM de 256×4 bits?
Res. 4 bits.

3.84. ¿Cuántas *palabras* de datos puede contener una ROM de 32×8 bits?
Res. 32.

3.85. ¿Cuál es la capacidad total en bits de una EAROM de 512×8 bits?
Res. 4096 bits.

Capítulo 4

FUNDAMENTOS DE LAS MICROCOMPUTADORAS

4.1. INTRODUCCION

Más de veinte compañías fabrican gran variedad de microprocesadores de propósito general. Muchas más compañías ensamblan microprocesadores, memorias y otros componentes de los sistemas microcomputadores. Debido a esta gran variedad, se ha seleccionado una microcomputadora genérica para estudiarla en este capítulo. Esta microcomputadora universal tiene muchas de las características de las microcomputadoras comunes.

Ante un nuevo microprocesador, el programador debe estudiar lo siguiente:

1. Arquitectura del microprocesador.
2. Repertorio de instrucciones.
3. Mínimo sistema que utiliza el microprocesador.
4. Señales de control.
5. Funciones de las patillas.

La arquitectura está relacionada con la organización de los registros de la CPU, número de bits de los buses de direcciones y datos, etc.

El repertorio de instrucciones es el conjunto de operaciones que puede realizar el microprocesador, donde se incluyen transferencia de datos, operaciones aritméticas y lógicas, instrucciones de comprobación de datos y bifurcación y operaciones de entrada/salida. Estas instrucciones utilizan diversos modos de direccionamiento.

El esquema de un sistema mínimo mostrará cómo se conectan al microprocesador otros dispositivos. Un sistema mínimo puede estar formado por microprocesador, reloj, RAM, ROM, puertos de entrada/salida, decodificador de direcciones y fuente de alimentación. A veces estas funciones son realizadas por CI o componentes separados; sin embargo, algunas unidades microprocesadoras pueden realizar la mayoría de estas funciones.

Las señales de control incluyen las salidas que controlan la operación de otros CI (como por ejemplo RAM, ROM y puertos de E/S). Algunas señales de control típicas pueden controlar lecturas y escrituras en memoria o lecturas y escrituras de entrada y salida.

Un estudio de la función de cada patilla del CI dará más detalles sobre las entradas y salidas especiales del microprocesador. Otras patillas pueden ser las de la fuente de alimentación, reloj, E/S de datos serie, entradas de interrupción y control del bus.

4.2. ARQUITECTURA SIMPLIFICADA DE UNA MICROCOMPUTADORA

La arquitectura de una microcomputadora se muestra en la Figura 4.1. En el centro de todas las operaciones está la MPU (unidad microprocesadora). La MPU necesita conexiones con la fuente

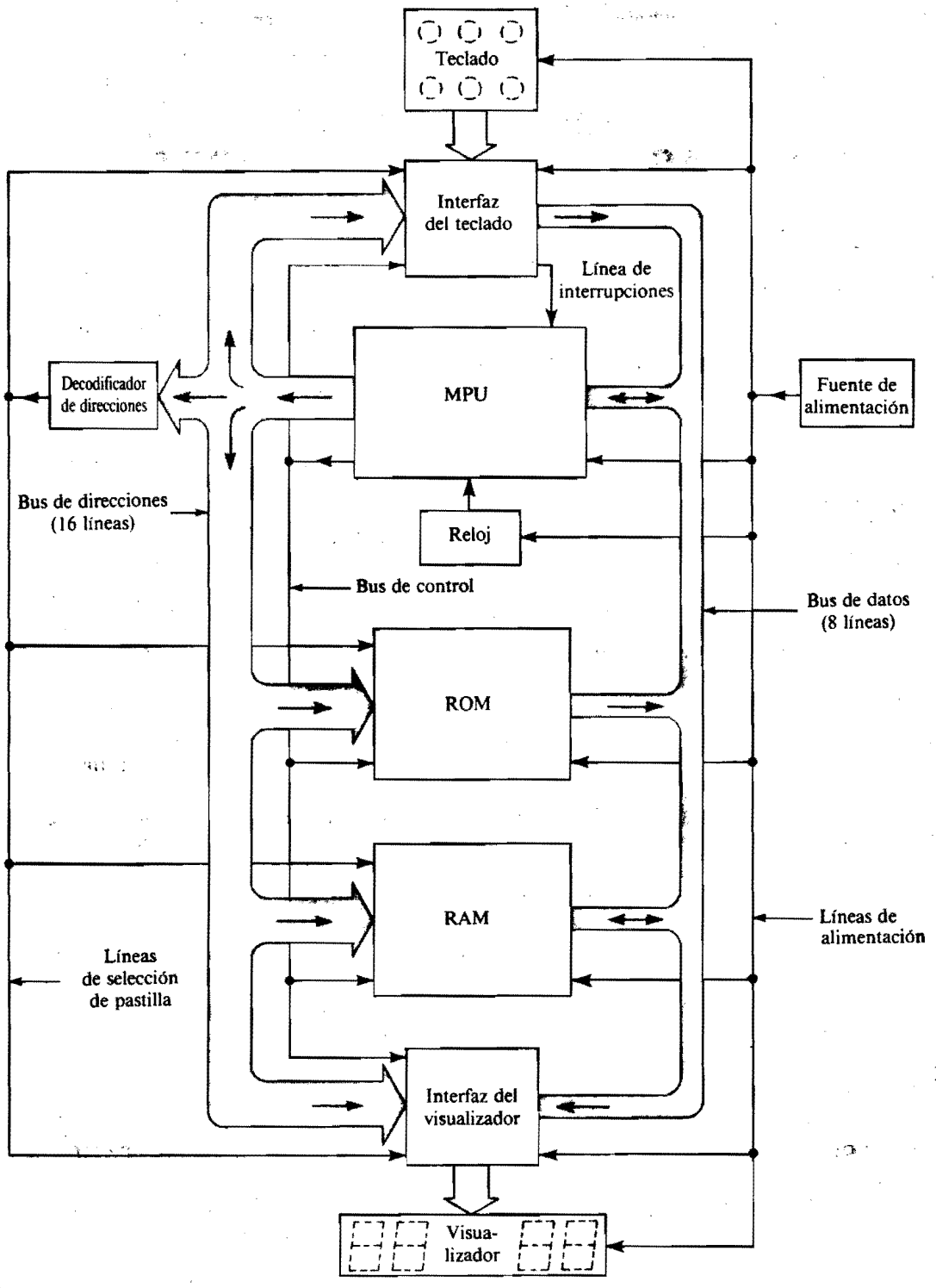


Figura 4.1. Arquitectura de una microcomputadora.

de alimentación y el reloj. El reloj puede ser un circuito separado o puede residir en la pastilla del microprocesador. Una sencilla MPU puede tener 16 líneas de dirección que forman el *bus de direcciones* de un sentido. La MPU también tiene las 8 líneas típicas de datos, con buffers que se conectan al *bus de datos* de dos sentidos.

La arquitectura de la microcomputadora mostrada en la Figura 4.1 tiene dos tipos de memorias semiconductoras. La ROM es la memoria permanente, que probablemente contiene el programa monitor del sistema. La ROM tiene las entradas de dirección junto a las líneas de selección de pastilla y de habilitación de lectura. También tiene la ROM 8 salidas, con buffers de tres estados, conectadas al bus de datos. Cada palabra de memoria tiene por tanto 8 bits. Se sobreentiende que la ROM tiene también conexiones a la fuente de alimentación aunque muchas veces se omitan en los diagramas de bloques.

La arquitectura de la Figura 4.1 muestra también una RAM como dispositivo de almacenamiento temporal de lectura/escritura. La RAM tiene las entradas de dirección junto a las entradas de selección de pastilla y de habilitación de lectura/escritura. La RAM tiene 8 salidas, con buffers de tres estados, conectadas al bus de datos. Esta RAM introduce, saca y almacena los datos como palabras de 8 bits. También se muestran las conexiones de la fuente de alimentación de la RAM.

El sistema microcomputador de la Figura 4.1 tiene un teclado como dispositivo de entrada. Las conexiones de alimentación al teclado se muestran junto con las líneas de datos en un CI especial denominado *interfaz de teclado.* El circuito de interfaz almacena los datos y coordina las entradas del teclado. En el instante adecuado, la interfaz de teclado *interrumpe* la MPU vía la *línea de interrupción* especial. Esta señal de interrupción hace que la MPU (1) termine de ejecutar la instrucción actual, (2) suspenda la operación normal y (3) bifurque a un grupo especial de instrucciones del programa monitor que manipulan la entrada de datos desde el teclado. El circuito de interfaz del teclado tiene entradas de dirección, de selección de pastilla y de control para activar la unidad. Una vez activada, la interfaz de teclado pondrá los datos del teclado en el bus de datos. La MPU acepta los nuevos datos de entrada, vía el bus de datos. Cuando las salidas de tres estados no están activadas, vuelven a su estado de alta impedancia.

La microcomputadora de la Figura 4.1 utiliza como salida un grupo de visualizadores de siete segmentos. El visualizador está conectado a la fuente de alimentación de la derecha. Un circuito especial de *interfaz de visualizador* o CI se utiliza para almacenar datos y excitar los visualizadores de la Figura 4.1. Cuando son activados por las entradas de dirección, de selección de pastilla y de habilitación, la interfaz acepta los datos del bus de datos y los almacena. La interfaz excita entonces, continuamente, los visualizadores, mostrando de forma visual los datos almacenados en la interfaz del visualizador.

Las 16 líneas del bus de direcciones pueden contener 65.536 (2^{16}) diferentes patrones de ceros y unos. Las líneas del bus de direcciones pueden estar conectadas a varios dispositivos como, por ejemplo, RAM, ROM e interfaces. Para conectar o habilitar *solamente* el dispositivo ordenado, un *decodificador de direcciones* muestrea datos en el bus de direcciones. La lógica combinacional del decodificador de direcciones activa la línea adecuada de selección de pastilla, habilitando el dispositivo correcto. Para simplificar la circuitería, no todas las 16 líneas del bus de direcciones van al decodificador de direcciones, memorias o interfaces.

PROBLEMAS RESUELTOS

4.1. Acudir a la Figura 4.1. La _____ (MPU, RAM, ROM) sirve como centro de todas las operaciones y control de la microcomputadora.

Solución:

La MPU sirve como centro de todas las operaciones y control de la microcomputadora. Sin embargo, las instrucciones de la ROM y/o RAM guían las acciones del microprocesador.

4.2. Acudir a la Figura 4.1. El bus de _____ (direcciones, datos) es una ruta de datos de un sentido.

Solución:

El bus de direcciones es una ruta de datos de un sentido, desde la MPU a todos los dispositivos.

4.3. Con 16 líneas, el bus de direcciones de la Figura 4.1 puede acceder hasta _____ (16.384, 65.536) posiciones separadas de memoria y/o de entrada/salida.

Solución:

El bus de direcciones de 16 líneas de la Figura 4.1 puede acceder hasta 65.536 (2^{16}) posiciones de memoria.

4.4. Acudir a la Figura 4.1. Citar como mínimo tres tipos de salida de la unidad microprocesadora.

Solución:

Según la Figura 4.1, la MPU tiene salidas de bus de direcciones (16 líneas), del bus de datos (8 líneas) y del bus de control.

4.5. Acudir a la Figura 4.1. Citar como mínimo cuatro tipos de entradas a la unidad microprocesadora.

Solución:

Según la Figura 4.1, la MPU tiene entradas de la fuente de alimentación, de reloj, de interrupción y del bus de datos (8 líneas).

4.6. Acudir a la Figura 4.1. Citar como mínimo cuatro tipos de entradas a la ROM.

Solución:

Según la Figura 4.1, la ROM tiene entradas de la fuente de alimentación, entradas de dirección provenientes del bus de direcciones, una entrada de selección de pastilla y una entrada de habilitación de lectura del bus de control.

4.7. Acudir a la Figura 4.1. Citar la(s) salida(s) de la ROM.

Solución:

Según la Figura 4.1, la ROM tiene salidas del bus de datos de tres estados (8 líneas).

4.8. Acudir a la Figura 4.1. Citar como mínimo cuatro tipos de entradas a la RAM.

Solución:

Según la Figura 4.1, la RAM tiene entradas de la fuente de alimentación, de dirección provenientes del bus de direcciones, una entrada de selección de pastilla y entradas de lectura/escritura provenientes del bus de control y entradas del bus de datos (8 líneas).

4.9. Acudir a la Figura 4.1. Citar la(s) salida(s) de la RAM.

Solución:

Según la Figura 4.1 la RAM tiene salidas del bus de datos de tres estados (8 líneas).

4.10. Acudir a la Figura 4.1. Citar como mínimo cinco tipos de entrada al circuito especial de interfaz del teclado.

Solución:

Según la Figura 4.1, el circuito de interfaz del teclado tiene entradas desde la fuente de alimentación, de dirección desde el bus de direcciones, una entrada de selección de pastilla y entradas de control desde el bus de control y de datos desde el teclado.

4.11. Acudir a la Figura 4.1. Citar como mínimo dos tipos de salida del circuito de interfaz del teclado.

Solución:

Según la Figura 4.1, el circuito de interfaz del teclado tiene una salida de interrupción y salidas del bus de datos de tres estados (8 líneas).

4.12. ¿Cuál es el propósito de la salida de interrupción del circuito de interfaz de teclado de la Figura 4.1?

Solución:

El circuito de interfaz de teclado de la Figura 4.1 activa la línea de interrupción cuando está lista para enviar datos de entrada desde teclado a la MPU.

4.13. ¿Qué hace la MPU cuando la línea de interrupción de la Figura 4.1 es activada por el circuito de interfaz de teclado?

Solución:

Cuando la línea de interrupción de la Figura 4.1 es activada, la MPU hace lo siguiente:

(a) Completa la instrucción que está ejecutando actualmente.
(b) Suspende la operación normal.
(c) Bifurca a un grupo especial de instrucciones que manipulan la entrada de datos desde el teclado.

4.14. Acudir a la Figura 4.1. Citar como mínimo cinco tipos de entradas al circuito especial de interfaz del visualizador.

Solución:

Según la Figura 4.1, el circuito de interfaz del visualizador tiene entradas de la fuente de alimentación, de dirección desde el bus de direcciones, de datos desde el bus de datos de 8 líneas, una entrada de selección de pastilla y entradas de control desde el bus de control.

4.15. Acudir a la Figura 4.1. Citar las salidas del circuito especial de interfaz del visualizador.

Solución:

Según la Figura 4.1, el circuito de interfaz del visualizador tiene salidas de datos para excitar los visualizadores de siete segmentos.

4.16. ¿Cuál es el propósito del decodificador de direcciones de la Figura 4.1?

Solución:

El decodificador de direcciones de la Figura 4.1 selecciona y ayuda a habilitar un *solo* dispositivo como, por ejemplo, la interfaz de teclado, ROM, RAM o la interfaz del visualizador.

4.3. ORGANIZACION SIMPLIFICADA DE LA MEMORIA

Escribir en o leer en una posición de memoria se denomina *acceder* a la memoria. Generalmente, el acceso a los datos de una memoria puede realizarse de forma secuencial o aleatoria. Los datos en la *memoria de acceso secuencial* se localizan buscándolos en forma serie a través de todas las posiciones de memoria. Por ejemplo, cuando los datos se almacenan en cinta magnética, hay que recorrer la cinta desde el principio al final hasta encontrar el dato apropiado.

En una *memoria de acceso aleatorio,* cualquier posición de memoria puede ser escrita o leída en un tiempo determinado (llamado *tiempo de acceso*). Los dispositivos semiconductores de almacenamiento, RAM y ROM, utilizados en las microcomputadoras son del tipo más rápido de acceso aleatorio.

La microcomputadora genérica que se ha estudiado tiene 16 líneas en el bus de direcciones. Esto es común en muchas pequeñas microcomputadoras utilizadas para entrenamiento. Las 16 líneas de dirección pueden generar un total de 65.536 (2^{16}) combinaciones diferentes de ceros y unos. Algunas combinaciones binarias se muestran bajo el bus de direcciones de la Figura 4.2. Es costumbre representar la dirección binaria en forma hexadecimal; como muestra la Figura 4.2, la dirección 0000 0000 0000 0000_2 es igual a 0000H (0000_{16}). Observar el uso de la H para designar la representación hexadecimal. La dirección más alta generada por el bus de 16 líneas de la Figura 4.2 es 1111 1111 1111 1111_2 que se representa por FFFFH ($FFFF_{16}$).

El diagrama de la Figura 4.3 representa el *mapa de memoria* de la microcomputadora. El bus de direcciones de 16 bits del microprocesador puede generar 65.536 (10000_{16}) direcciones únicas, algunas de las cuales aparecen en notación hexadecimal en la columna izquierda de la Figura 4.3. En esta microcomputadora particular, las 256 primeras posiciones (100_{16}) se encuentran en la ROM. La ROM es, probablemente, de 256 × 8 bits (256 palabras, de 8 bits cada una). Si el bus de direcciones accede a la dirección 0000H, la ROM presentará en la salida su patrón de ceros y unos programado permanentemente (la palabra 11000011).

Es conveniente para los principiantes pensar que los 256 primeros bytes o palabras (100_{16}) de datos en la ROM de la Figura 4.3 están en la página 00H. Observar que el número de página es igual a la representación hexadecimal del byte de orden superior de la dirección (ver Fig. 4.2). Así cuando se trata de la página 00H, solamente es necesario considerar los dos dígitos hexadecimales de orden inferior. Algunos microprocesadores utilizan instrucciones simplificadas para acceder a las posiciones de memoria en la primera página (por ejemplo, el 6502, el 65816).

El mapa simplificado de memoria de la Figura 4.3 muestra que desde la página 01H hasta la 1FH (direcciones 0100H - 1FFFH) no hay memoria en esta microcomputadora. El acceso a esta

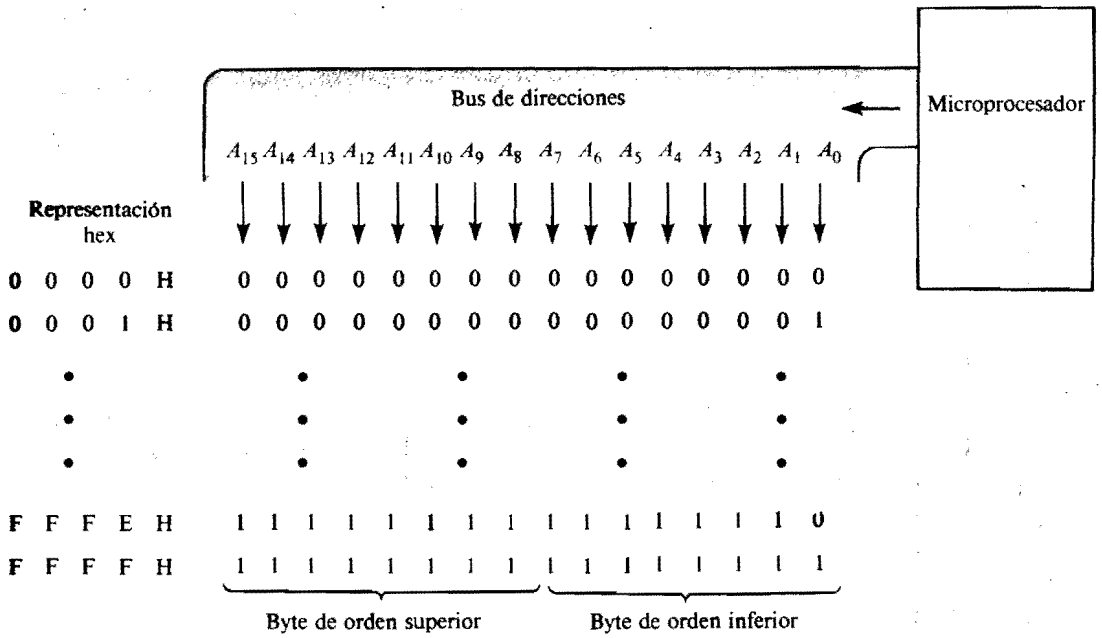


Figura 4.2. Representación de las direcciones del microprocesador.

área abierta producirá resultados impredecibles ya que no está almacenado ningún programa ni ningún dato. La memoria de lectura/escritura de la microcomputadora (RAM) se localiza en la página 20H, de acuerdo con el mapa de memoria de la Figura 4.3. La RAM es un dispositivo de almacenamiento de 256 × 8 bits. Los diseñadores del sistema microcomputador podían haber colocado la RAM en cualquier otra página. El acceso al área comprendida entre las páginas 21H y FFH, en esta microcomputadora, producirá resultados impredecibles porque ningún dispositivo de almacenamiento está ubicado en esta área abierta.

Si un microprocesador fuese a escribir en la posición de memoria 2000H de la Figura 4.3, las ocho celdas de memoria (bits) de la parte superior de la RAM se rellenarían con ceros y unos del bus de datos. Después, si el microprocesador fuese a leer en la dirección 2000H, los datos previamente escritos serían leídos de la RAM.

Físicamente la ROM de la Figura 4.3, probablemente, es un solo CI. Sin embargo, las RAM están organizadas, con frecuencia, de forma diferente. En la Figura 4.3, el área de almacenamiento más oscura de la RAM está dividida en dos mitades. Esto sugiere que el área de memoria de la RAM de la página 20H está formada por dos CI físicamente separados. En este ejemplo se utilizan dos RAM de 256 × 4 bits para formar el área de almacenamiento de 256 × 8 bits en la página 20H de esta microcomputadora.

En la Figura 4.4 se muestran tres organizaciones físicas de CI de RAM semiconductoras. El CI de la Figura 4.4*a* está organizado como una RAM de 256 × 8 bits. En la Figura 4.4*b* se utilizan dos CI para formar el mismo dispositivo de almacenamiento de 256 × 8 bits. Aquí se utilizan dos RAM de 256 × 4 bits para construir la palabra de 8 bits necesaria. Las ocho RAM mostradas en la Figura 4.4*c* también forman un dispositivo de almacenamiento de lectura/escritura de 256 × 8 bits.

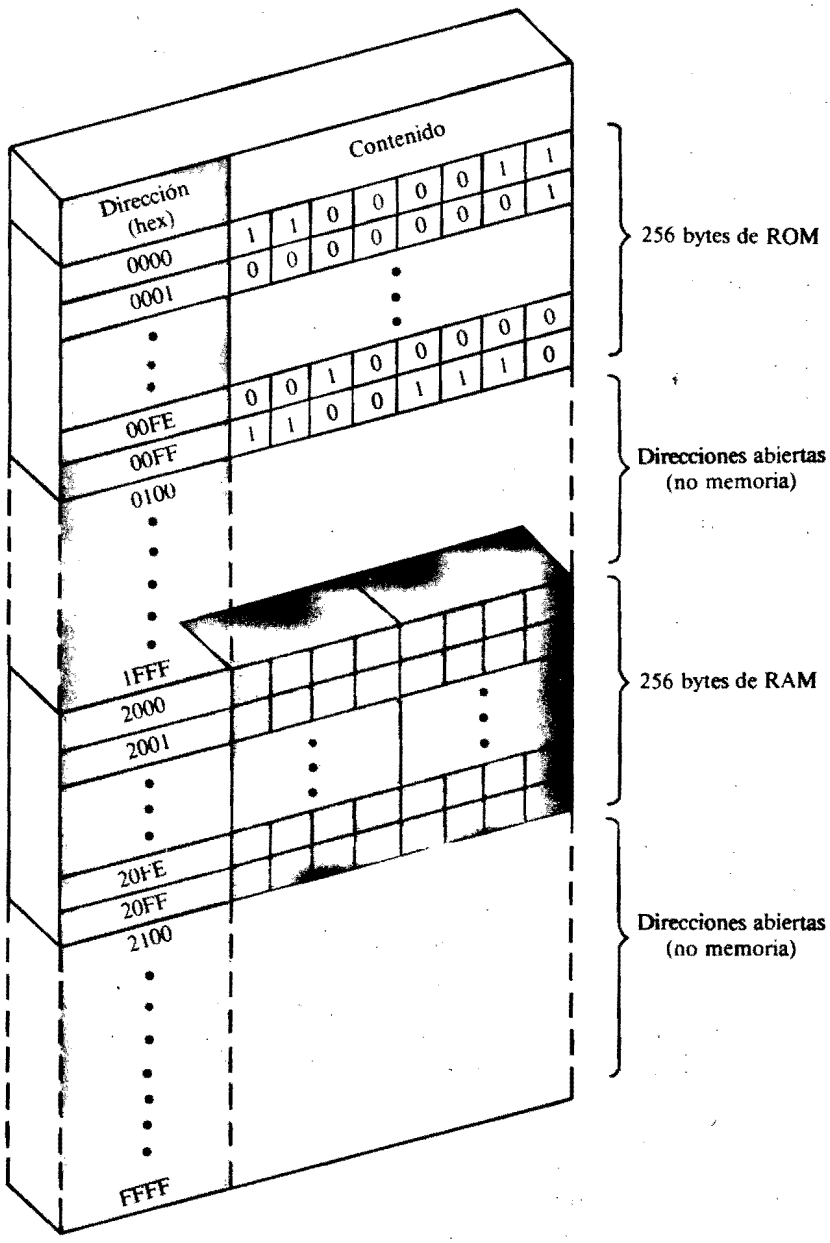


Figura 4.3. Diagrama simplificado de memoria mostrando las posiciones de la ROM y la RAM.

La organización de CI mostrada en la Figura 4.4*c* se utiliza normalmente para construir grandes memorias de lectura/escritura. En general se utilizan CI de RAM de 1024×1; 16.384×1; 65.536×1, y $1.048.576 \times 1$. Utilizando la técnica mostrada en la Figura 4.4*c*, son necesarias ocho RAM de 4096×1 bit son necesarias para formar una memoria de lectura/

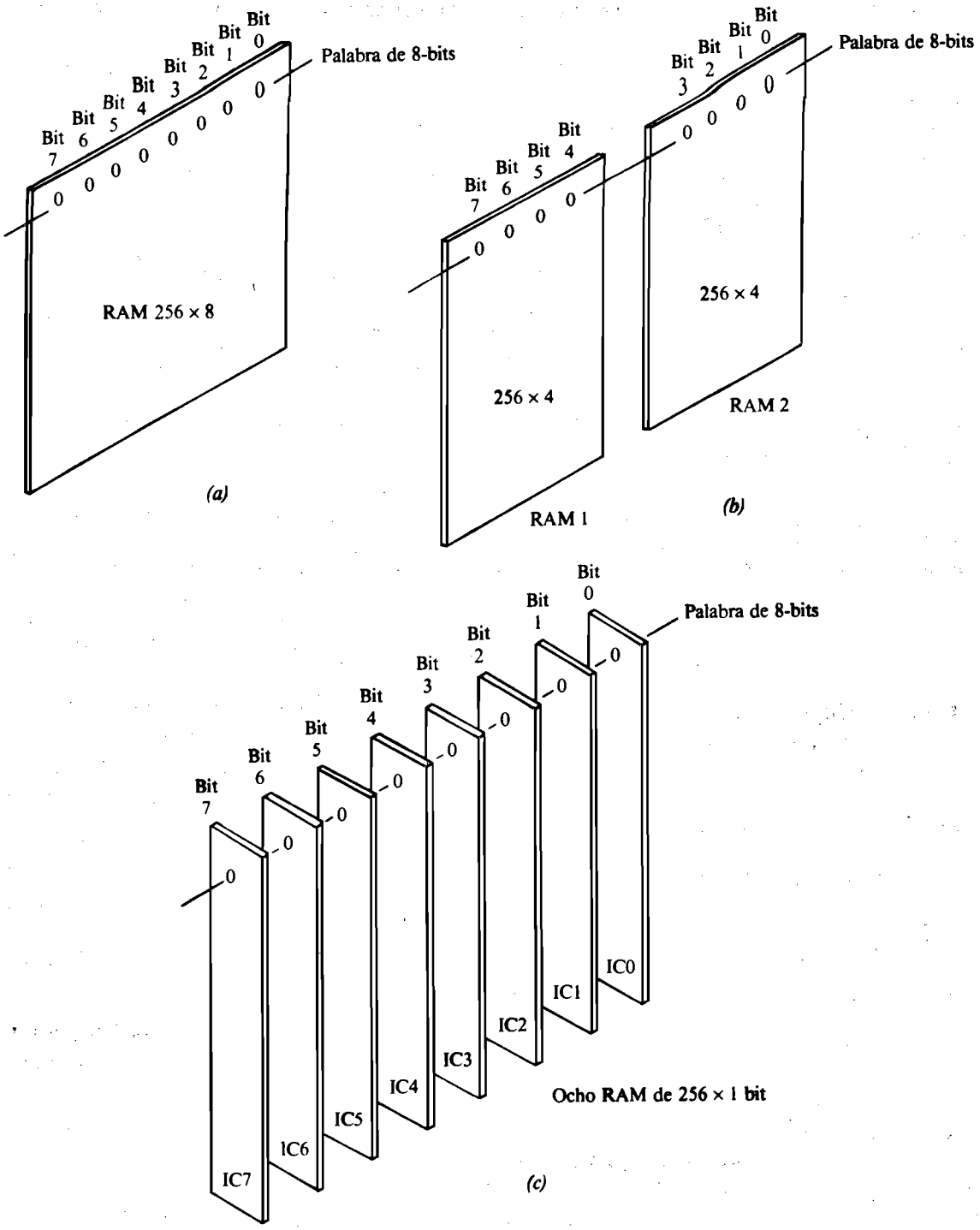


Figura 4.4. Tres métodos de formar una memoria de lectura/escritura de 256 × 8 bits utilizando RAM semiconductoras.

escritura de 4096 × 8 bits. Observar que en dicha configuración, las ocho RAM están activadas por la misma línea de selección de pastilla procedente del decodificador de direcciones.

En un sistema microcomputador, la memoria se divide en bloques de 1K. Una capacidad de 1K no *significa exactamente* 1000, sino 2^8 o 1024 bytes de memoria (1024 × 1). Por tanto, 4K de memoria significan 4096 bytes (1024 × 4 = 4096). Si una microcomputadora comercial tiene 256K de memoria, realmente tendrá 262.144 (1024 × 256 = 262.144) posiciones de memoria de 8 bits cada una. La capacidad de memoria de las microcomputadoras comerciales actuales varía desde unos 64K a más de 1 millón (1M) de bytes. Una capacidad de memoria de 1M realmente significa 1.048.576 (1024 × 1024 = 1.048.576) bytes de memoria.

PROBLEMAS RESUELTOS

4.17. Escribir o leer en una posición de memoria de una microcomputadora se denomina _____ (acceder a, encontrar la) memoria.

Solución:

Escribir o leer en una posición de memoria se denomina acceder a la memoria.

4.18. Las RAM y ROM utilizadas en las microcomputadoras son ejemplos de memorias de acceso _____ (aleatorio, secuencial).

Solución:

Las RAM y ROM son ejemplos de memorias de acceso aleatorio.

4.19. Acudir a la Figura 4.2. Si la dirección en el bus de direcciones fuese 0010 0000 0000 0000, ¿a qué posición de memoria (número hexadecimal) accedería el microprocesador?

Solución:

El binario 0010 0000 0000 0000 es igual a 2000H. Se accedería a la posición de memoria 2000H si la salida del bus de direcciones fuese 0010 0000 0000 0000 de la Figura 4.2.

4.20. Acudir a la Figura 4.3. Si la microcomputadora direcciona la posición de memoria 0001H, la ___ (RAM, ROM) sacará ___ ___ ___ ___ ___ ___ ___ ___ (8 bits).

Solución:

De acuerdo con el mapa de memoria de la Figura 4.3, la posición 0001H de la ROM producirá la salida del dato almacenado permanentemente 00000001.

4.21. Acudir a la Figura 4.3. Si la microcomputadora accede a la posición de memoria 2001H y si el dispositivo de almacenamiento está en el modo de escritura, un _____ (byte, nibble) de datos del bus de datos se colocará en la _____ (RAM, ROM).

Solución:

De acuerdo con la Figura 4.3, la posición de memoria 2001H está en una RAM cuyas palabras son de 8 bits (bytes).

4.22. Acudir a la Figura 4.3. ¿Qué salida aparecerá en el bus de datos si el microprocesador lee la posición de memoria FFFFH?

Solución:

De acuerdo con el mapa de memoria de la Figura 4.3, ningún dispositivo de memoria reside en la dirección FFFFH, y por tanto la salida será impredecible.

4.23. La _____ (RAM, ROM) se ha implementado utilizando solamente un CI.

Solución:

En la Figura 4.3 la ROM se ha implementado utilizando solamente un CI. En este ejemplo la memoria de lectura/escritura se implementó utilizando dos CI RAM de 256 × 4 bits.

4.24. ¿En qué página reside la RAM de la Figura 4.3?

Solución:

La RAM de la Figura 4.3 reside en la página 20H de memoria. Los dos dígitos hexadecimales de orden superior se consideran el número de página en este sistema.

4.25. Ocho CI RAM de 16.384 × 1 bits pueden organizarse de forma similar a los de la Figura 4.4*c* para formar una memoria de lectura/escritura de 16.384 × 8 bits, que se denomina memoria de _____ K.

Solución:

La RAM de 16.384 × 8 bits se referencia como una memoria de 16K. La memoria de 16K tiene 16.384 bytes o palabras de memoria.

4.26. Acudir a la Figura 4.3. ¿Cuál es la memoria máxima que puede ser accedida por 16 líneas de dirección?

Solución:

Un bus de 16 líneas de dirección puede acceder a 64K de memoria. Las direcciones van de 0000H a FFFFH, que es igual a 65.536_{10} posiciones. En este caso las palabras de memoria son de 8 bits; sin embargo, otros sistemas pueden tener 4 ó 16 bits por palabra.

4.27. Si una microcomputadora comercial contiene 512K de memoria, realmente tendrá _____ posiciones de memoria de 8 bits cada una.

Solución:

Una microcomputadora con una memoria de 512K tendrá 524.288 (1024 × 512 = 524.288) bytes de memoria.

4.4. REPERTORIO DE INSTRUCCIONES

El grupo de instrucciones que un microprocesador específico puede ejecutar se denomina su *repertorio de instrucciones*. Los repertorios de instrucciones de los microprocesadores pueden variar desde ocho hasta 200 instrucciones básicas. Los repertorios de instrucciones no están estandarizados, debido al individualismo de cada fabricante y a la diferencia de arquitectura y uso de los microprocesadores.

Las instrucciones de un repertorio pueden clasificarse de diversas formas. En este capítulo, se organizarán en las siguientes categorías:

1. Instrucciones aritméticas.
2. Instrucciones lógicas.
3. Instrucciones de transferencia de datos.
4. Instrucciones de bifurcación.
5. Instrucciones de llamada a subrutina.
6. Instrucciones de vuelta.
7. Instrucciones misceláneas.

Un sencillo repertorio de instrucciones de un microprocesador debería incluir las siguientes *instrucciones aritméticas:*

1. Sumar.
2. Restar.
3. Incrementar.
4. Decrementar.
5. Comparar.
6. Negar.

Otras instrucciones aritméticas utilizadas por algunos microprocesadores pueden incluir operaciones de suma con arrastre, resta con préstamo, multiplicación y división.

El mismo microprocesador simplificado debería tener las siguientes *instrucciones lógicas:*

1. AND.
2. OR.
3. OR exclusiva.
4. Not.
5. Desplazamiento a la derecha.
6. Desplazamiento a la izquierda.

Otras instrucciones lógicas utilizadas por algunos microprocesadores pueden incluir desplazamiento aritmético a la derecha, desplazamiento circular a la derecha, desplazamiento circular a la izquierda, desplazamiento circular a la derecha a través del arrastre, desplazamiento circular a la izquierda a través del arrastre y operación de test (prueba).

El microprocesador básico debería contener variaciones de las siguientes *instrucciones de transferencia de datos:*

1. Cargar.
2. Almacenar.
3. Transferir.
4. Entrar.
5. Salir.

Otras instrucciones de transferencia de datos utilizadas por algunos microprocesadores pueden incluir intercambio y diversas operaciones de borrado e inicialización.

El microprocesador debería contener las siguientes *instrucciones de bifurcación* en su repertorio de instrucciones:

1. Bifurcación incondicional.
2. Bifurcación si cero.
3. Bifurcación si no cero.
4. Bifurcación si igual.

5. Bifurcación si no igual.
6. Bifurcación si positivo.
7. Bifurcación si negativo.

Otras instrucciones de bifurcación condicional utilizadas por algunos microprocesadores pueden depender de condiciones como mayor que o menor que, no arrastre o arrastre, no «overflow» u «overflow». Las operaciones de bifurcación son instrucciones de toma de decisiones.

Un sencillo microprocesador debería tener una *instrucción de llamada a subrutina* (referencia como CALL) para hacer que el programa bifurque o salte a un grupo especial de instrucciones que realizan una tarea específica. Todos los microprocesadores tienen la instrucción de llamada incondicional, y algunos también tienen instrucciones de llamada condicional. Las instrucciones de llamada condicional pueden incluir llamada si cero, llamada si no cero, llamada si positivo, llamada si no positivo, etc.

Al final de la subrutina el programa debe volver donde originalmente abandonó el programa principal. Esta tarea la ejecuta una *instrucción de vuelta.* Las instrucciones de vuelta (retorno) pueden incluir el retorno de una subrutina o de las operaciones de interrupción. Las vueltas son habitualmente incondicionales, pero algunos microprocesadores contienen instrucciones de vuelta condicional.

Un repertorio de instrucciones simplificado de un microprocesador debería incluir las siguientes *instrucciones misceláneas:*

1. No operación.
2. Introducir.
3. Sacar.
4. Esperar.
5. Alto.

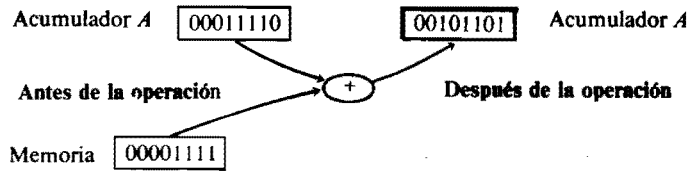
Otras instrucciones misceláneas pueden incluir habilitar interrupción, inhabilitar interrupción, ruptura y ajuste decimal.

Los usuarios del microprocesador pueden describir de varias formas la misma instrucción, como por ejemplo se esboza en la Figura 4.5 para la instrucción de sumar del microprocesador Motorola 6800. El nombre de la instrucción es comúnmente un *verbo de acción* y se denomina *operación.* En la Figura 4.5*a*, la operación es sumar. Los usuarios de la MPU también trabajan frecuentemente con formas abreviadas de la instrucción denominadas *nemotécnicos.* En la Figura 4.5*a*, el nemotécnico para esta operación particular de sumar es ADD *A* (observar que los nemotécnicos son mayúsculos). El registro de instrucción y la circuitería de decodificación de la instrucción de un microprocesador comprenden solamente ceros y unos. El *código de op* es la representación hexadecimal del código binario de 8 bits que hará que la MPU ejecute la instrucción. En la Figura 4.5*a*, el código de op para ADD A es $8B_{16}$ (10001011_2) para el microprocesador 6800. La columna simbólica de la Figura 4.5*a* muestra que el contenido de memoria (*M*) se suma al contenido del acumulador *A* de la MPU, y la flecha indica que la suma se coloca en el acumulador *A*.

Un ejemplo que utiliza la instrucción de sumar (ADD *A*) está esbozado en la Figura 4.5*b*. El contenido del acumulador *A* (00011110_2) se suma al contenido de memoria (00001111_2), la suma (00101101_2) se coloca en el acumulador *A*, a la derecha. Observar que no ha cambiado el contenido de la posición de memoria de la Figura 4.5*b*, mientras ha cambiado el contenido del acumulador *A* y ahora el acumulador *A* contiene la nueva suma.

Operación	Nemotécnico	Código Op (hex)	Simbólico (Todas las etiquetas de los registros referencian contenidos)
Sumar	ADD *A*	8B	$A + M \rightarrow A$

(a) Resumen de la instrucción de suma



(b) Instrucción de suma

Figura 4.5.

PROBLEMAS RESUELTOS

4.28. El grupo de operaciones que puede ejecutar un microprocesador se denomina su _____ _____ .

Solución:

El grupo de operaciones que puede ejecutar un microprocesador se denomina su repertorio de instrucciones.

4.29. Citar siete tipos de las instrucciones que se encuentran en el repertorio de instrucciones de un microprocesador.

Solución:

Una agrupación de operaciones del repertorio contiene las siguientes categorías: aritméticas, lógicas, de transferencia de datos, bifurcaciones, llamadas a subrutina, vueltas de subrutina e instrucciones misceláneas.

4.30. Una operación ADD (de SUMA) es una instrucción aritmética, mientras que una operación OR (O) se denomina una instrucción _____ .

Solución:

Una operación OR se clasifica como una instrucción lógica.

4.31. Las operaciones de desplazamiento se denominan instrucciones _____ .

Solución:

Las operaciones de desplazamiento se denominan instrucciones lógicas.

4.32. Listar cinco operaciones básicas de transferencia de datos encontradas en la mayor parte de los repertorios de instrucciones de los microprocesadores.

Solución:

Las operaciones de transferencia de datos del microprocesador incluyen cargar, almacenar, transferir, entrar y salir.

4.33. Las operaciones de toma de decisión de un microprocesador se denominan instrucciones de _____ .

Solución:

Las operaciones de toma de decisión de un microprocesador se denominan instrucciones de bifurcación.

4.34. Listar varias operaciones de bifurcación condicional que pueden encontrarse en el repertorio de instrucciones de un microprocesador.

Solución:

Las instrucciones de bifurcación condicional de un microprocesador típico incluyen las siguientes: bifurcación si cero, bifurcación si no cero, bifurcación si igual, bifurcación si no igual, bifurcación si positivo, bifurcación si negativo, bifurcación si mayor que, bifurcación si menor que, bifurcación si arrastre, bifurcación si no arrastre, bifurcación si overflow y bifurcación si no overflow.

4.35. Un grupo especial de instrucciones que realizan una tarea específica, dentro de un programa, se denomina un(a) _____ (índice, subrutina).

Solución:

Un grupo especial de instrucciones que realizan una tarea específica en un programa, se denomina subrutina.

4.36. ¿Qué instrucciones hacen que el microprocesador bifurque a una subrutina?

Solución:

La instrucción CALL o de llamada a subrutina hace que la MPU bifurque a una subrutina.

4.37. ¿Qué instrucción se coloca al final de una subrutina para indicar al microprocesador que vuelva a la posición correcta en el programa principal?

Solución:

La instrucción vuelta de subrutina (RET) se coloca al final de una subrutina para hacer que el microprocesador bifurque a la posición correcta del programa principal. Esta instrucción también puede restaurar algunos registros a la condición que tenían antes de la operación de llamada a subrutina.

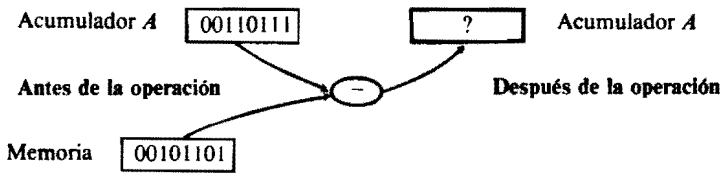
4.38. Listar cinco operaciones misceláneas encontradas en los repertorios de instrucciones de muchos microprocesadores.

Solución:

Las instrucciones misceláneas de la MPU incluyen no operación, introducir (*push*), sacar (*pop*), esperar (*wait*), alto (*halt*), habilitación de interrupción (*enable interrupt*), inhabilitación de interrupción (*disable interrupt*), bifurcar y ajuste decimal.

Operación	?	? (hex)	Simbólico (Todas las etiquetas de los registros referencian contenidos)
Restar	SUB *A*	80	$A - M \rightarrow A$

(a) Resumen de la instrucción de resta



(b) Problema de resta

Figura 4.6.

4.39. Acudir a la Figura 4.6*a*. ¿Qué deberá aparecer encima de SUB *A* en el diagrama que describe la instrucción de resta?

Solución:

La abreviatura SUB *A* es el nemotécnico para restar del acumulador *A*. En la Figura 4.6*a*, encima de SUB *A* debe aparecer «Nemotécnico».

4.40. El 80 hexadecimal de la Figura 4.6*a* se denomina _____ _____ .

Solución:

El 80_{16} de la Figura 4.6*a* se denomina código de op de la instrucción de restar.

4.41. Acudir a la Figura 4.6*b*. La diferencia binaria que deberá aparecer en el acumulador *A*, después de la operación SUB *A*, será _____ .

Solución:

El binario 00110111 - 00101101 = 00001010 (decimal 55 − 45 = 10). El acumulador *A* contendrá la diferencia de 00001010_2 después de la operación SUB *A* en la Figura 4.6*b*.

4.5. ORGANIZACION SIMPLIFICADA DE LA CPU

La unidad principal de cualquier sistema computador se denomina *unidad central de tratamiento* o *CPU*. Muchos CI microprocesadores *son* la CPU del sistema. Es común en tecnología de computadoras implementar la memoria de programa, memoria de datos, interfaces de entrada/salida, decodificador de direcciones y CPU en CI separados, como sugiere el sistema de la Figura 4.1. La MPU forma la unidad central de tratamiento en este sistema.

Generalmente la CPU contiene elementos de memoria denominados registros y circuitería computacional denominada unidad aritmética y lógica (ALU). También contiene circuitería de decodificación de instrucciones y una sección de temporización y control, además de las conexiones necesarias de entrada y salida.

Las funciones principales de la CPU de una microcomputadora son:

1. Buscar, decodificar y ejecutar instrucciones del programa en el orden adecuado.
2. Transferir datos entre memoria y las secciones de entrada/salida.
3. Responder a las interrupciones externas.
4. Proporcionar señales de temporización y control al sistema completo.

La mayoría de las CPU de los microprocesadores contienen como mínimo los elementos esbozados en la Figura 4.7. Las secciones principales incluyen diversos registros, la unidad aritmética y lógica, el decodificador de instrucciones y la importante sección de temporización y control, junto a las entradas y salidas. La mayoría de las CPU realmente contienen algunos registros más especiales, así como muchas entradas y salidas especializadas no detalladas en la Figura 4.7.

La *unidad aritmética y lógica (ALU)* de la CPU realiza operaciones como sumar, desplazar/desplazar circularmente, comparar, incrementar, decrementar, negar, AND, OR, XOR, complementar, borrar y preinicializar (*preset*). Si la ALU ejecutase la instrucción de SUMAR (ADD), el procedimiento podría ser parecido a lo esbozado en la Figura 4.8*a*. Aquí el contenido del acumulador ($0A_{16}$) se suma al contenido del registro temporal (05_{16}). La suma ($0F_{16}$) entonces se vuelve a colocar en el acumulador.

Algunas de las secciones funcionales de una ALU típica se encuentran en la Figura 4.8*b*. La ALU contiene un sumador y desplazador, que está conectado al acumulador, vía el bus interno

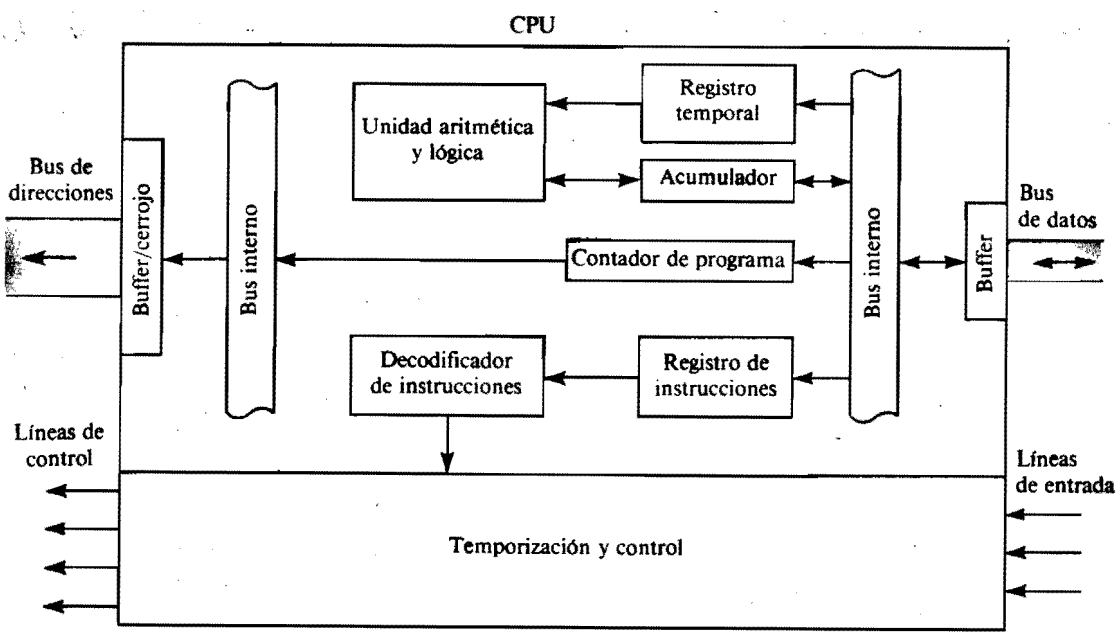
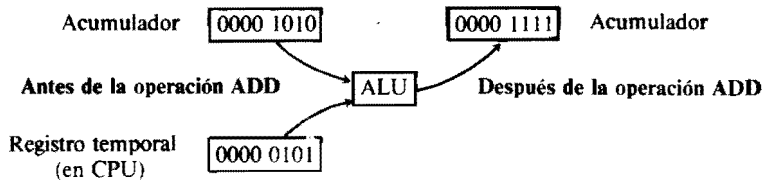
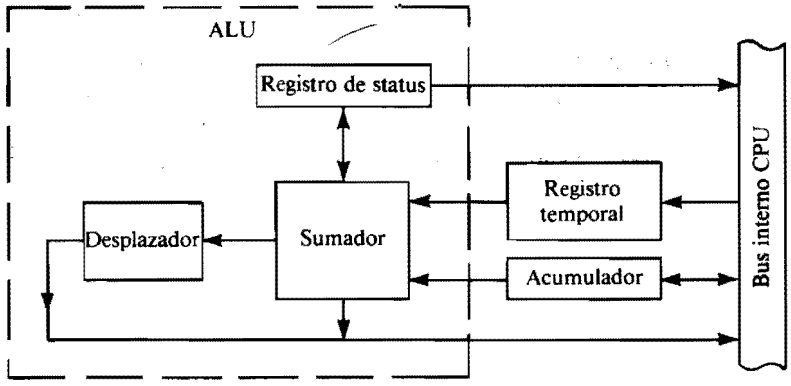


Figura 4.7. Arquitectura simplificada de una CPU.



(a) Ejecución de la instrucción de sumar en la ALU



(b) Organización de la ALU

Figura 4.8.

de datos. De importancia crítica para el programador es el *registro de status* localizado en la ALU, también denominado *registro de códigos de condición* o de *señalizadores*. El registro es realmente un grupo de flip-flops individuales que pueden ser puestos a 1 ó 0 según las condiciones creadas por la última operación de la ALU. Los flip-flops individuales, o señalizadores, incluyen indicadores para cero, resultados negativos, arrastres desde el MSB, etc. Los señalizadores son utilizados para tomar decisiones cuando posteriormente se emplean instrucciones de bifurcación. Los registros temporales y acumulador mostrados en la Figura 4.8*b* se consideran muchas veces parte de la ALU. El acumulador es utilizado en la mayoría de las operaciones realizadas por la CPU, como por ejemplo en las transferencias de datos.

La *sección de temporización y control* de la Figura 4.7 es probablemente la sección más compleja de la CPU. Afecta y secuencia todos los eventos de la CPU y de la microcomputadora. En un capítulo anterior se mencionó que cada instrucción del programa puede dividirse en etapas de búsqueda, decodificación y ejecución. Cada una de estas etapas puede ser además subdividida en una serie de pequeños pasos, denominados *microprograma*. El microprograma para cada instrucción reside en la sección de decodificación de instrucciones y es ejecutado por la sección de temporización y control de la CPU.

Un registro de 16 bits llamado *contador de programa* se muestra como parte de la CPU de la Figura 4.7. El contador de programa es responsable de seguir la pista de la dirección de la siguiente instrucción de memoria que se va a buscar. Debido a que las instrucciones del programa, normalmente, son ejecutadas en secuencia, el contador de programa cuenta normalmente de forma ascendente hasta que no se le diga que haga otra cosa. La mayoría de los microprocesado-

tienen un contador de programa de 16 bits que accederá a 64K palabras de memoria vía el bus de direcciones. La ejecución secuencial normal de las instrucciones del programa puede ser alterada por instrucciones especiales de bifurcación, llamada, o vuelta, o por interrupciones que hacen que el contador de programa bifurque a un número distinto de la dirección siguiente. Para ejecutar un programa después de conectar la computadora, el operador debe inicializar el contador de programa con el número de la posición de la primera instrucción del programa.

La *secuencia de búsqueda-decodifica-ejecuta la instrucción* es fundamental para la operación de la computadora. La primera instrucción buscada, de la memoria de programa, se *supone* que es el código de op de la primera instrucción y es colocada en el *registro de instrucciones* por la sección de control de la CPU. El código de op es entonces interpretado por el *decodificador de instrucciones*. El decodificador de instrucciones indica a la sección de temporización y control que ejecute la instrucción específica.

La organización de la CPU mostrada en la Figura 4.7 es una versión simplificada. La mayoría de las CPU de los microprocesadores contienen como mínimo algunos registros más de 8 y 16 bits. Existen grandes variaciones en los microprocesadores según el número y tipos de registros de la CPU.

PROBLEMAS RESUELTOS

4.42. Las letras CPU significan ¿qué parte de un sistema microcomputador?

Solución:

Las letras CPU significan unidad central de tratamiento.

4.43. Acudir a la Figura 4.1. La CPU reside ¿en qué bloque de este sistema microcomputador?

Solución:

La unidad central de tratamiento reside en el bloque MPU (unidad microprocesadora) del sistema mostrado en la Figura 4.1.

4.44. Una CPU generalmente contiene *(a)* dispositivos de memoria denominados _____, *(b)* circuitería computacional denominada _____, *(c)* circuitería _____ _____ y *(d)* una sección de temporización y _____.

Solución:

Una CPU generalmente contiene *(a)* dispositivos de almacenamiento llamados registros, *(b)* circuitería computacional llamada ALU, *(c)* circuitería de decodificación de instrucciones y *(d)* una sección de temporización y control.

4.45. Citar cuatro tareas principales de la CPU de una microcomputadora.

Solución:

Cuatro tareas de la CPU de una microcomputadora son:

(a) Buscar, decodificar y ejecutar instrucciones de programa en el orden adecuado.
(b) Transferir datos entre memoria y las secciones de entrada/salida.
(c) Responder a interrupciones externas.
(d) Proporcionar señales de temporización y control a todo el sistema.

4.46. ¿Qué significan las letras ALU?

Solución:

Las letras ALU significan unidad aritmética y lógica.

4.47. Acudir a la Figura 4.9. ¿Cuál es el contenido binario del acumulador después de la operación AND?

Solución:

El contenido binario del acumulador de la Figura 4.9 después de realizar la operación AND, bit a bit, es 00000000. La tabla de verdad AND se muestra en la Figura 3.1.

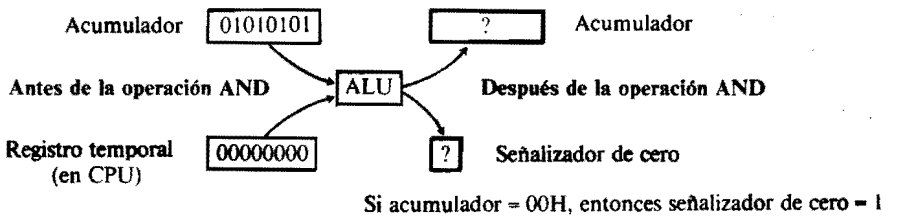


Figura 4.9.

4.48. Acudir a la Figura 4.9. El señalizador de cero estará en _____ (set, reinicialización) después de la operación AND.

Solución:

La condición del señalizador de cero se basa en el contenido del acumulador después de la operación AND. Debido a que el acumulador es igual a cero, el señalizador de cero del registro de status se pondrá a 1 después de la operación AND de la Figura 4.9.

4.49. El registro de status de la ALU también se denomina registro de códigos _____, y los flip-flops individuales se denominan _____ (señalizadores, signos).

Solución:

El registro de status de la ALU se denomina registro de códigos de condición, y los flip-flops individuales muchas veces se denominan señalizadores (*flags*).

4.50. ¿Qué sección importante de la CPU afecta y secuencia todos los eventos en la CPU y en la microcomputadora?

Solución:

La sección de control y temporización de la CPU afecta y secuencia todos los eventos en la CPU y en la microcomputadora.

4.51. El registro de la CPU que sigue la pista de la dirección de la siguiente instrucción que se va a buscar en la memoria de programa se denomina _____ _____.

Solución:

El contador de programa sigue la pista de la dirección de la siguiente instrucción que se va a buscar en la memoria de programa.

4.52. El contador de programa normalmente incrementa su contenido accediendo a las instrucciones de la memoria de programa en orden secuencial excepto cuando se altera por ¿qué tipos de instrucciones?

Solución:

Las instrucciones de bifurcación, llamada o vuelta, o interrupciones, pueden alterar el normal incremento del contador de programa.

4.53. Al comienzo de la ejecución de un programa el código de op de la primera instrucción se coloca en el _____ (acumulador, registro de instrucciones) de la CPU.

Solución:

Al comienzo de la ejecución de un programa, el código de operación de la primera instrucción se coloca en el registro de instrucciones de la CPU.

4.54. La sección de la CPU que interpreta el código de op colocado en el registro de instrucciones e indica la sección de control y temporización sobre cómo ejecutar la instrucción se denomina _____ .

Solución:

El decodificador de instrucciones de la CPU interpreta el código de op en el registro de instrucciones e indica a la sección de control y temporización cómo ejecutar la instrucción.

4.6. OPERACION DE LA MICROCOMPUTADORA

Considerar la sencilla tarea de sumar tres números como $10 + 5 + 18 = 33_{10}$. Puede escribirse un pequeño programa de microcomputadora para que realice esta tarea elemental siguiendo los pasos que a continuación se indican:

Instrucción 1 CARGAR (LOAD) el primer número (10_{10}) en la CPU.
Instrucción 2 SUMAR (ADD) el segundo número (5_{10}) al primero.
Instrucción 3 SUMAR (ADD) el tercer número (18_{10}) a la suma de los dos primeros números.
Instrucción 4 ALMACENAR (STORE) la suma de los tres números (33_{10}) en la posición 2000_{16} de la memoria de datos.

Después de cargarlas en la memoria de programa, las instrucciones anteriores deben aparecer como las del diagrama de memoria de la Figura 4.10. Observar que la primera instrucción del programa comienza en la dirección 0000H. La primera instrucción (CARGAR el número 0AH) tiene 2 bytes de memoria. El primer byte de memoria contiene la parte de *operación* de la instrucción, mientras que el segundo byte contiene el *operando*. El código de op para la operación LOAD (CARGAR) es 86H (10000110_2) para el microprocesador que se está utilizando en este ejemplo. El operando 0AH (00001010_2) es el primer número que se va a cargar en el acumulador del microprocesador. Observar en la Figura 4.10 que es costumbre representar las cantidades binarias en notación hexadecimal. En la máquina real éstos son niveles ALTOS y BAJOS de tensión.

Suponer que el programa se coloca en el bloque de RAM de la microcomputadora ilustrada en la Figura 4.1. Suponer también que la CPU de la microcomputadora tiene las características

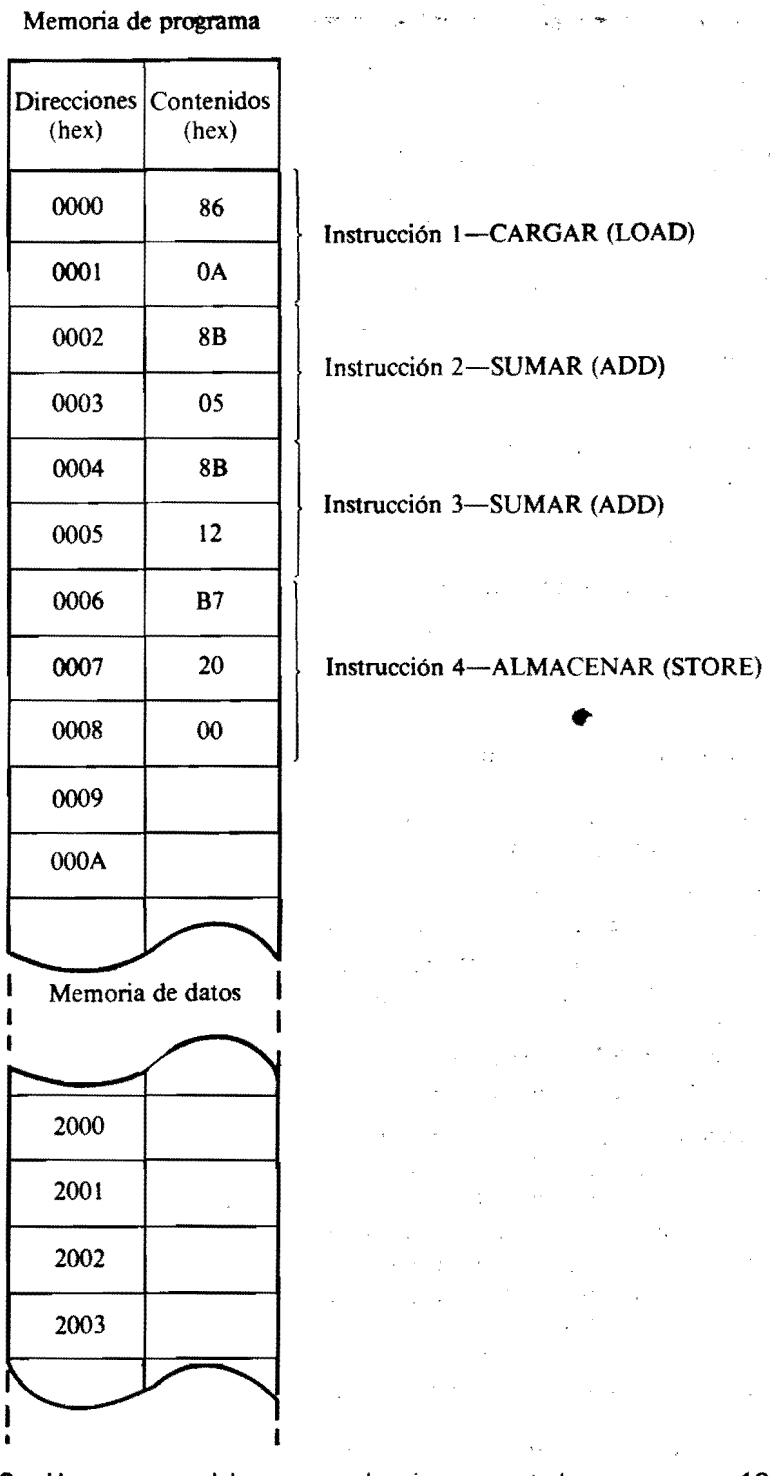
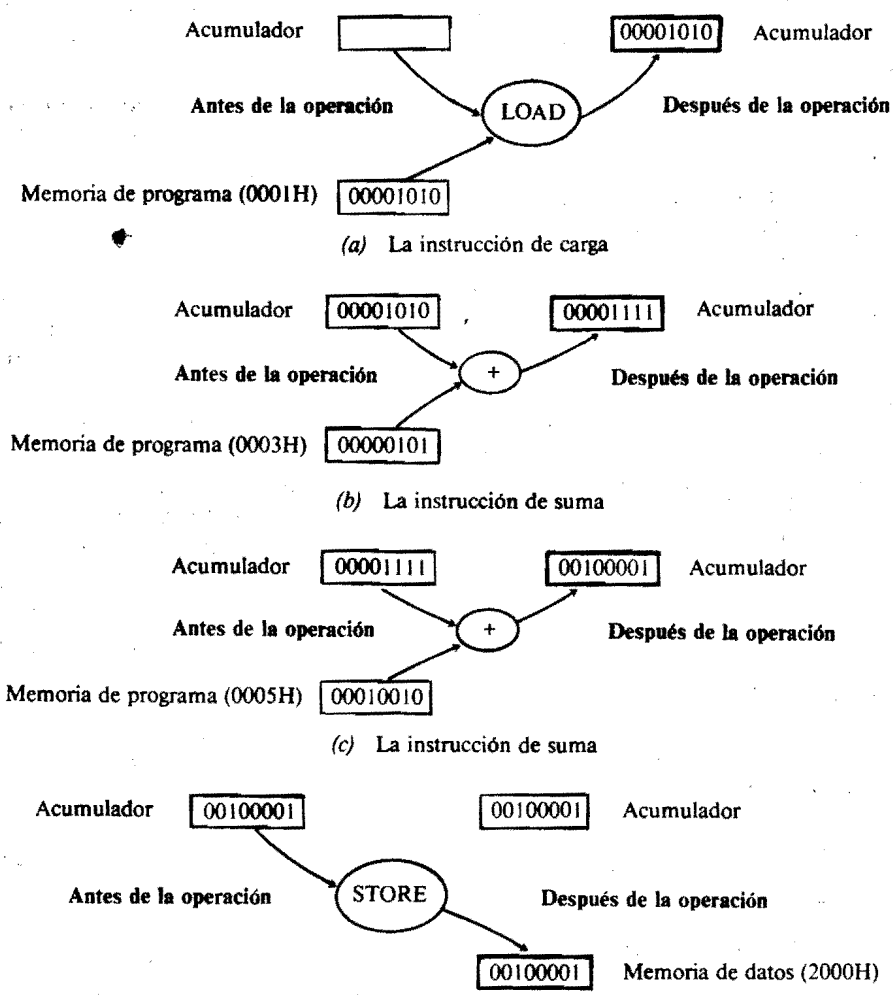


Figura 4.10. Un segmento del programa de microcomputadora para sumar 10 + 5 + 18.

detalladas en la Figura 4.7. Entonces la Figura 4.11 detalla los resultados de cada una de las cuatro operaciones del programa CARGAR-SUMAR-SUMAR-ALMACENAR.

La operación CARGAR (instrucción 1) detallada en la Figura 4.11*a* muestra el contenido de la posición de memoria 0001H cargado en el acumulador de la CPU. Después de la operación, el acumulador contiene 00001010_2, que es el primer número a sumar. La operación de CARGAR sobreescribe en el contenido del acumulador y lo borra. La operación de SUMAR (instrucción 1) se muestra en la Figura 4.11*b*. El contenido de la posición de memoria 003H (00000101_2) se suma al contenido del acumulador (00001010_2), dando la suma de 00001111_2. La suma se coloca en el acumulador. Observar que el número del acumulador cambia durante la operación de suma (ADD).



(a) La instrucción de carga

(b) La instrucción de suma

(c) La instrucción de suma

(d) La instrucción de almacenar directo

Figura 4.11.

La segunda operación de SUMA (instrucción 3) se detalla en la Figura 4.11*c*. La suma acumulada, 00001111_2, se suma al contenido de la posición de memoria 0005H. La suma binaria es 00001111 + 00010010 = 00100001. La suma final de 00100001_2 aparece en el acumulador después de la segunda operación de SUMA.

La operación de ALMACENAR (instrucción 4) se muestra en la Figura 4.11*d*. El contenido del acumulador (00100001_2) es transferido y almacenado en la posición 2000H de la memoria de datos. Observar que la posición de la memoria de datos estaba identificada en la memoria de programa por 2 bytes separados (posiciones 007 y 0008H). La posición de memoria de programa 0006H en la Figura 4.10 contiene B7H que es el código de op para la instrucción ALMACENAR directo.

Considerar la búsqueda, decodificación y ejecución de la instrucción CARGAR en las direcciones 0000H y 0001H de la memoria de programa de la Figura 4.10. Este tipo de instrucción de CARGAR probablemente puede ser ejecutada entre 2 y 6 microsegundos *(μs)* por muchas microcomputadoras. Algunas de las acciones que ocurren en la CPU, buses, líneas de control y memoria se detallan para la instrucción de CARGAR en la Figura 4.12.

Comenzando en la parte superior izquierda de la Figura 4.12, el contador de programa en la CPU se inicializa con la dirección del primer paso del programa. A continuación, la dirección de 16 bits es transferida al registro de direcciones y después al bus de direcciones y memoria de programa. Para activar la memoria de programa, la CPU envía una señal de lectura (línea $R/\overline{W}$ a 1) mientras que el decodificador de direcciones (no parte de la CPU) habilita la selección de pastilla de memoria ($\overline{CS}$) con un 0. A continuación, el contador de programa en la CPU es incrementado a 0001H. La posición 0000H de la memoria de programa es accedida y leída en el bus de datos. El código de op (86H) para la instrucción de CARGAR es transferido al registro de instrucciones de la CPU. Esto completa la etapa de búsqueda del código de op de la instrucción CARGAR.

A continuación el código de op (86H) que está contenido en el registro de instrucciones de la CPU es interpretado por el decodificador de instrucciones. En este caso la CPU determina que se trata de la *instrucción CARGAR inmediato,* que significa que cargará el contenido de la posición de memoria que sigue inmediatamente al código de op en el acumulador. El contenido del contador de programa (0001H) es transferido al registro de direcciones, bus de direcciones y memoria. La CPU genera un pulso de lectura ALTO, que va a la entrada $R/\overline{W}$ de la memoria. Un nivel BAJO se aplica a la entrada de memoria $\overline{CS}$, que la habilita. En la tercera línea de la parte inferior de la Figura 4.12, el contador de programa es incrementado a 0002H, quedando listo para la búsqueda de la siguiente instrucción. Se accede a la posición de memoria 0001H, y su contenido (0AH) es colocado en el bus de datos y después en el acumulador de la CPU. Observar que las etapas de búsqueda y ejecución de esta instrucción son similares a las operaciones de lectura, excepto que el dato leído de memoria era transferido a un registro diferente de la CPU. El código de op de la etapa de búsqueda se coloca siempre en el registro de instrucciones.

PROBLEMAS RESUELTOS

4.55. Las dos partes de una instrucción de un microprocesador se denominan *operación* y _____.

Solución:

Las dos partes de una instrucción de un microprocesador se denominan operación y operando.

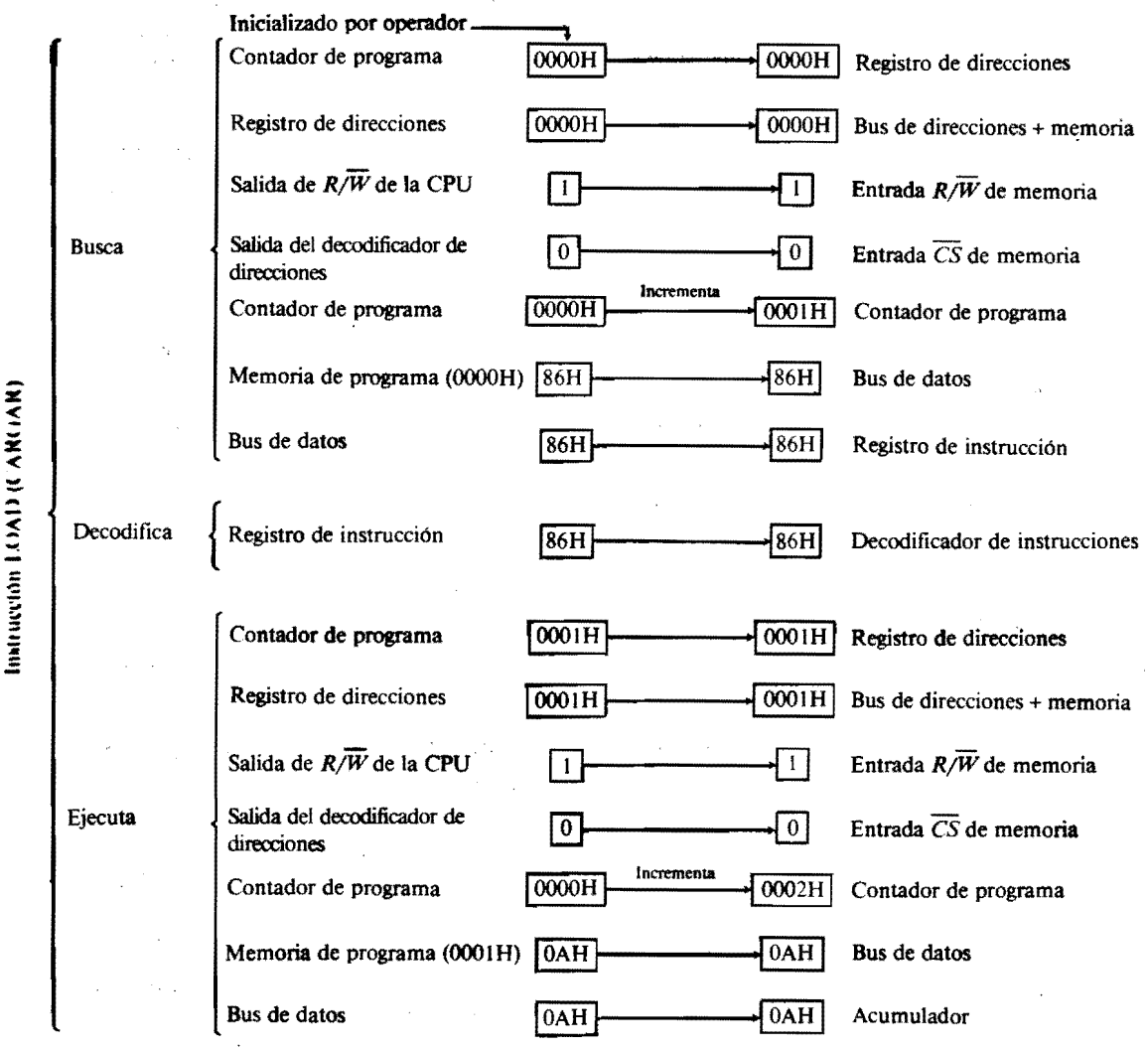


Figura 4.12. Un análisis detallado de la instrucción de cargar inmediato.

4.56. Acudir a la Figura 4.10. Cuando es accedida la posición 0002H de la memoria de programa, el contenido (8B) es transferido, vía bus de _____ (direcciones, datos), al registro _____ de la CPU.

Solución:

Cuando es accedida la posición 0002H de la memoria de programa de la Figura 4.10, el contenido (8B) es transferido, vía bus de datos, al registro de instrucción de la CPU.

4.57. Acudir a la Figura 4.10. La posición de la memoria del programa 0003H contiene 05H, que se considera la parte _____ (operando, operación) de la instrucción.

Solución:

En la Figura 4.10, la posición de memoria 0003H contiene 05H, que se considera la parte operando de la instrucción.

4.58. Un número binario de 16 bits está representado normalmente por _____ (un número) dígitos hexadecimales.

Solución:

Un número binario de 16 bits es costumbre representarlo por cuatro dígitos hexadecimales. Por tanto $0010\ 0000\ 0000\ 1111_2$ puede ser representado por 200F en notación hexadecimal.

4.59. Listar las tres etapas que sigue la CPU cuando ejecuta cualquier instrucción.

Solución:

Los ciclos de la CPU son las etapas de búsqueda, decodificación y ejecución cuando corre cualquier instrucción.

4.60. Acudir a la Figura 4.12. La etapa de *búsqueda* de esta instrucción es básicamente una operación de _____ (lectura, escritura), donde el contenido de memoria acaba en el _____ (acumulador, registro de instrucciones) de la CPU.

Solución:

La etapa de búsqueda de esta o cualquier instrucción es básicamente una operación de lectura, donde el contenido de memoria acaba en el registro de instrucciones.

4.61. Durante la etapa de búsqueda de una instrucción, el _____ (decodificador de instrucciones, contador de programa) apunta a la dirección del código de op que se va a acceder en memoria.

Solución:

Durante la etapa de búsqueda de una instrucción, el contador de programa apunta la dirección del código de op que se va a acceder en memoria.

4.62. Acudir a la Figura 4.12. Además de las entradas de dirección, ¿qué otras dos entradas de memoria necesitan ser activadas para leer un byte de memoria?

Solución:

Según la Figura 4.12, la entrada $R/\overline{W}$ (lectura/escritura) de la MPU y la $\overline{CS}$ (selección de pastilla) de memoria del decodificador de direcciones necesitan ser habilitadas junto con las entradas de dirección para leer un byte de memoria.

4.63. Acudir a la Figura 4.10. Una vez que la CPU ejecuta la instrucción 4, ¿qué número decimal se almacenará en la posición 2000H de la memoria de datos?

Solución:

El programa de la Figura 4.10 suma 0AH + 05H + 12H, dando un total de 21H. La suma de 21H será almacenada entonces en la posición 2000H de la memoria de datos. Esto se ilustra también en la Figura 4.11*d*.

4.64. Acudir a la Figura 4.12. Una vez conectado a la red, la inicialización del contador del programa es realizada por el _____ (decodificador de direcciones, operador).

Solución:

La inicialización del contador de programa es realizada por el operador. El contador de programa se inicializa con la dirección de la memoria de programa de la primera instrucción del programa que se va a ejecutar. Esta primera inicialización del contador de programa puede, o no, ser 0000H como se indica en la Figura 4.12.

PROBLEMAS SUPLEMENTARIOS

4.65. Listar cinco cosas que sea necesario conocer cuando se aprende algo sobre un nuevo microprocesador.

Res. arquitectura, repertorio de instrucciones, sistema mínimo que utiliza el microprocesador, señales de control y funciones de las patillas.

4.66. El bus de una dirección de 16 bits de la Figura 4.1 se denomina bus de _____.

Res. direcciones.

4.67. El bus de 8 bits de dos direcciones de la Figura 4.1 se denomina bus de _____.

Res. datos.

4.68. Acudir a la Figura 4.1. Las conexiones de la MPU al bus de datos se consideran _____ (entradas, salidas, entradas y salidas).

Res. entradas y salidas.

4.69. Acudir a la Figura 4.2. Si la dirección en el bus de direcciones fuese 0010 0000 1000 1111, ¿a qué posición de memoria (número hex) accedería el microprocesador?

Res. $208F_{16}$.

4.70. Acudir a la Figura 4.3. Si la microcomputadora accede a la posición de memoria 00FEH, la ROM _____ (sacará, aceptará) _ _ _ _ _ _ _ _ (8 bits).

Res. sacará, 00100000.

4.71. La ROM de la Figura 4.3 se dice que reside ¿en qué página de memoria?

Res. 00_{16} (página cero).

4.72. Ocho CIs RAM de 4096 × 1 bit pueden organizarse de forma similar a los de la Figura 4.4*c* para formar una memoria de lectura/escritura de 4096 × _____ bits, que se denominaría memoria de _____ K.

Res. 8, 4.

4.73. La lista de operaciones que pueden ser realizadas por un microprocesador se denomina su repertorio de _____.

Res. instrucciones.

4.74. Una operación de SUMAR (ADD) es una instrucción aritmética, mientras que una operación de ALMACENAR (STORE) es una instrucción de _____.

Res. transferencia de datos.

4.75. Las operaciones de bifurcación son las instrucciones de toma de _____ del microprocesador.

Res. decisiones.

4.76. Las instrucciones de llamada a subrutina y las _____ (aritméticas, de vuelta) están siempre asociadas con las subrutinas.

Res. de vuelta.

4.77. La circuitería computacional de una CPU se denomina _____ (tres letras).

Res. ALU (unidad aritmética y lógica).

4.78. La CPU de una microcomputadora contiene una serie de dispositivos de almacenamiento temporal denominados _____ (registros, ROM).

Res. registros.

4.79. La CPU de una microcomputadora contiene circuitería utilizada para interpretar instrucciones a la que se denomina _____.

Res. decodificador de instrucciones.

4.80. La CPU de una microcomputadora contiene un bloque de circuitería para secuenciar eventos, que se denomina sección de _____ y _____.

Res. control, temporización.

4.81. La CPU de una microcomputadora reside en un dispositivo denominado _____ (interfaz, microprocesador).

Res. microprocesador.

4.82. Acudir a la Figura 4.13. ¿Cuáles son los contenidos binarios del acumulador después de la operación OR?

Res. 00111111.

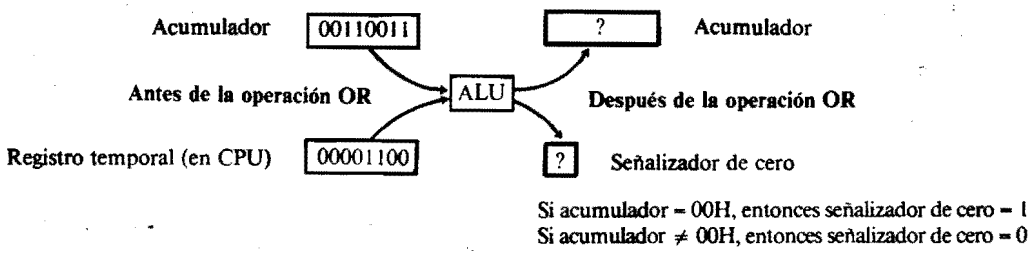


Figura 4.13. La instrucción OR.

4.83. Acudir a la Figura 4.13. El señalizador de cero estará a _____ (1, 0) después de la operación OR.

Res. a 1.

4.84. Los señalizadores son parte de un código de condición especial o registro _____ que se encuentra en la ALU de la unidad central de tratamiento de una microcomputadora.

Res. de status.

4.85. El contador de programa de la CPU sigue la pista _____ (de la dirección, del tiempo) de la siguiente instrucción del programa que se va a ejecutar.

Res. de la dirección.

4.86. Durante la etapa de búsqueda, un _____ (código de op, operando) es leído de la memoria de programa y transferido al registro de instrucción de la CPU.

Res. código de op.

4.87. Durante la etapa de decodificación, el decodificador de la instrucción en la _____ (CPU, memoria de programa) interpreta el código de op que actualmente reside en el registro _____ (de instrucciones, temporal).

Res. CPU de instrucciones.

4.88. Acudir a la Figura 4.14*a* y *b*. ¿Cuál es el contenido binario del acumulador de la CPU en la Figura 4.14*b* después de la operación AND?

Res. 00000000_2.

Memoria de programa

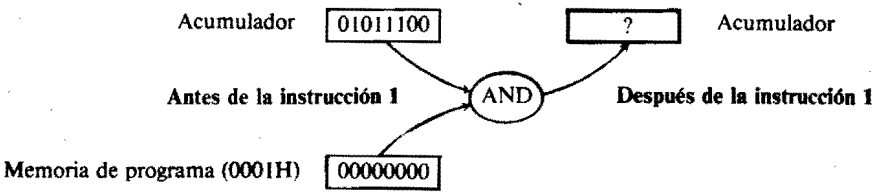
Dirección (hex)	Contenidos (hex)	
0000	84	Instrucción 1—AND lógica del operando 00H con el contenido desconocido del acumulador de la CPU
0001	00	
0002	4C	Instrucción 2—INCREMENTA el acumulador
0003	B7	Instrucción 3—ALMACENA (STORE) el contenido del acumulador en la posición 2001H de la memoria de datos
0004	20	
0005	01	
0006		

Memoria de datos

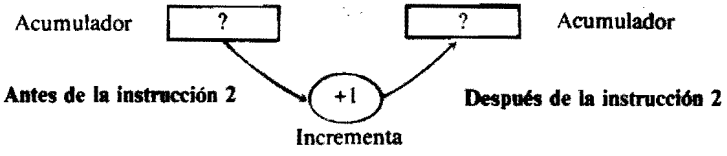
2000	
2001	
2002	

(a) Segmento de programa

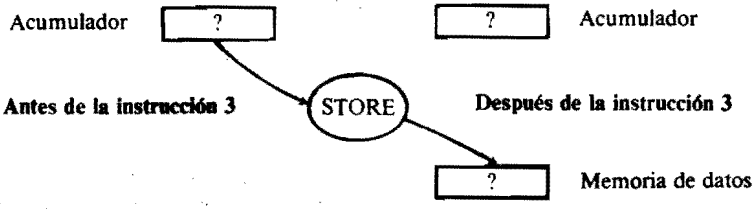
Figura 4.14.



(b) La instrucción AND



(c) La instrucción incrementar



(d) La instrucción de almacenar directo

Figura 4.14. *(continuación).*

4.89. Acudir a la Figura 4.14*a* y *c*. El contenido binario del acumulador de la CPU en la Figura 4.14*c* después de la operación de incrementación es _____ (8 bits).

Res. 00000001_2.

4.90. Acudir a la Figura 4.11*a* y *d*. ¿Cuál es la dirección hexadecimal de la posición de la memoria de datos mostrada en la Figura 4.14*d*?

Res. 2001H.

4.91. Acudir a la Figura 4.14*a* y *d*. El contenido binario de la posición de la memoria de datos mostrada en la Figura 4.14*d* después de la operación de almacenar es _____ (8 bits).

Res. 00000001_2.

4.92. Acudir a la Figura 4.14*a*. ¿Qué dos posiciones de la memoria de programa contienen las direcciones de la operación de almacenar?

Res. 0004H y 0005H.

4.93. Acudir a la Figura 4.14*a*. El código de op para la instrucción 1 es _____ (hex), mientras que el operando es _____ (hex).

Res. 84H, 00H.

4.94. Acudir a la Figura 4.14*a*. El código de op para la instrucción 2 es _____, mientras que el operando es _____.

Res. 4CH, no existe operando en la instrucción de incrementar.

4.95. Acudir a la Figura 4.14*a*. El código de op para la instrucción 3 es _____, mientras que el operando es _____.

Res. B7H, combina un operando de 2 bytes para formar la dirección 2001H de la memoria de datos.

Capítulo 5

EL MICROPROCESADOR

5.1. CARACTERISTICAS COMUNES DE LOS MICROPROCESADORES

Un *microprocesador* puede definirse brevemente como una pastilla de muy alta escala de integración (VLSI), que realiza las tareas de la unidad central de tratamiento de una microcomputadora u otro sistema de control automático. Lo que sigue es un esbozo, abreviado, de las características principales compartidas por casi todos los microprocesadores de 8 y 16 bits. Este listado sirve para dar al estudiante una visión general de las características más importantes de los diversos microprocesadores.

Conexiones de alimentación

Los microprocesadores (excepto el 8080) requieren una fuente de alimentación regulada de 5 V dc.

Tamaño en bits

Los microprocesadores se clasifican normalmente en unidades de 4, 8, 16 ó 32 bits. El tamaño en bits de un microprocesador a veces se denomina tamaño de palabra. La longitud del registro acumulador es una buena pista para conocer el tamaño de palabra de un microprocesador. Los microprocesadores 8080/8085, 6800, 6502 y Z80 son de 8 bits. Los 8086, 8088, 68000, 65816 y Z8000 son MPU típicas de 16 bits. El 80386, 68020, 32000 y Z-80000 son ejemplos de microprocesadores avanzados de 32 bits.

Líneas de datos

Los microprocesadores transfieren datos e instrucciones entre la MPU y memoria (o E/S) vía un bus de datos bidireccional. El 6800, 6502, Z80 y 8088 son procesadores que utilizan buses externos de datos de 8 bits. Muchos miembros de la familia 8080 multiplexan direcciones o información de control en las líneas de datos parte del tiempo.

Líneas de dirección

Los microprocesadores más antiguos (8080/8085, 6800, 6502) utilizan buses de dirección de 16 bits que pueden direccionar solamente 2^{16} ó 64K de memoria. Las MPU más modernas de 16 bits tienen buses de direcciones de 16, 20 ó 23 bits. Un bus de direcciones más ancho permite direccionar memorias mayores.

Líneas de control

La mayoría de los microprocesadores se caracterizan porque tienen todas o algunas de las siguientes líneas de control:

1. Líneas de reloj.
2. Líneas de lectura/escritura.
3. Líneas de entrada/salida.
4. Líneas de interrupción.
5. Líneas de reinicialización.
6. Líneas de control del bus.
7. Líneas de status del ciclo.

Registros internos

Contador de programa. El contador de programa (PC) es el registro que contiene la dirección de la siguiente instrucción del programa. La longitud del contador de programa es igual que la anchura del bus de direcciones. El contador de programa normalmente contiene 16 bits en los microprocesadores de 8 bits pero es mayor en las MPU de 16 y 32 bits.

Acumulador. El acumulador es el registro o registros asociados a las operaciones de la ALU y a veces a las operaciones de E/S. Puede ser de 8, 16 ó 32 bits. Las MPU del 8080/8085, 6800 y 6502 tienen todas acumuladores de 8 bits. Algunos procesadores (68000 y Z8000) tienen sólo registros de propósito general que pueden ser utilizados como acumuladores.

Registro de status o señalizadores. El registro de status está en todos los microprocesadores. Los bits individuales del registro se denominan *señalizadores.* Las condiciones de los señalizadores se asocian, generalmente, a las operaciones de la ALU y son utilizados por instrucciones de bifurcación posteriores para tomar decisiones.

Registros de propósito general. Los registros de propósito general pueden utilizarse para almacenar datos temporalmente o para que contengan una dirección. No tienen asignada ninguna tarea específica. En los microprocesadores de 8 bits, los registros de propósito general no pueden funcionar como un acumulador en la ALU y en operaciones de E/S. Sin embargo, las MPU de 16 bits habitualmente permiten que los registros de propósito general se utilicen como acumuladores.

Registro índice. El registro índice se utiliza para que contenga la dirección de un operando cuando se utiliza el modo de direccionamiento indexado (8080/8085, 6800, 6502, Z80, 8086). Los registros de propósito general son utilizados como registros índice en los microprocesadores Z8000 y 68000.

Registro del puntero de pila. El puntero de pila (SP) es un registro especializado que sigue la pista de la siguiente posición de memoria disponible en la pila. La *pila* es un área reservada de la RAM utilizada para almacenamiento temporal de datos, direcciones de vuelta y contenido de registros. La pila se utiliza durante las llamadas a subrutina y durante las interrupciones.

Modos de direccionamiento

Un *modo de direccionamiento* es la técnica utilizada para buscar el operando deseado durante la ejecución de una instrucción. Los microprocesadores individuales pueden no utilizar todos los modos de direccionamiento listados a continuación:

1. Modo de direccionamiento inherente (implícito).
2. Modo de direccionamiento inmediato.

3. Modo de direccionamiento extendido o absoluto.
4. Modo de direccionamiento de registro.
5. Modo de direccionamiento indirecto de registro.
6. Modo de direccionamiento índice.
7. Modo de direccionamiento de página cero (directo).
8. Modo de direccionamiento relativo.
9. Modo de direccionamiento basado.
10. Modo de direccionamiento basado en índice.
11. Modo de direccionamiento de cadena.
12. Modo de direccionamiento de E/S.
13. Modo de direccionamiento de status de registro.

Generalmente, los microprocesadores más modernos de 16 bits tienen modos de direccionamiento más potentes y flexibles.

PROBLEMAS RESUELTOS

5.1. Casi todos los microprocesadores requieren una fuente de alimentación regulada de _____ V dc.

Solución:

Todos los microprocesadores (excepto el 8080) requieren una fuente de alimentación regulada de 5 V dc.

5.2. El 8080/8085, 6800, Z80 y 6502 son todos ejemplos de microprocesadores de _____ bits.

Solución:

El 8080/8085, 6800, Z80 y 6502 son todos ejemplos de microprocesadores de 8 bits. Todos tienen acumuladores de 8 bits.

5.3. La mayoría de los microprocesadores de 8 bits utilizan buses de datos de _____ (8, 16) bits para transferir datos e instrucciones a y desde memoria (o E/S).

Solución:

La mayoría de las MPU de 8 bits utilizan líneas de datos de 8 bits.

5.4. La mayoría de los microprocesadores de 8 bits utilizan buses de direcciones de _____ (8, 16, 20) bits.

Solución:

La mayoría de los microprocesadores de 8 bits utilizan buses de direcciones de 16 bits. El bus de 16 bits direccionará 2^{16} ó 64K de memoria.

5.5. La línea de control _____ envía un mensaje desde la MPU a la RAM, indicando a la RAM si el acceso a memoria es una operación de lectura o escritura.

Solución:

La línea de control de lectura/escritura de la MPU señala la RAM si el acceso a memoria es una operación de lectura o escritura.

5.6. El registro interno de la MPU que contiene la dirección de la siguiente instrucción del programa es el _____ .

Solución:

El contador de programa contiene la dirección de la siguiente instrucción de memoria.

5.7. Los bits individuales del registro de status de un microprocesador se denominan comúnmente _____ .

Solución:

Los bits individuales del registro de status del microprocesador se denominan comúnmente señalizadores.

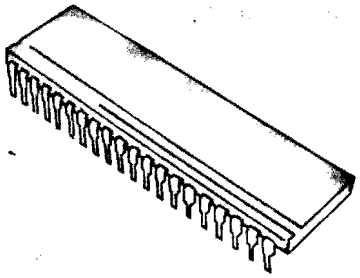
5.2. DESCRIPCIONES DE LA HOJA DE DATOS

Los folletos sobre microprocesadores específicos contienen varias características comunes. Las hojas de datos contienen información sobre el empaquetamiento del CI, diagramas de patillas y función de cada patilla del CI. El esquema de la arquitectura de la CPU aparece junto a una descripción de las características más importantes. Los diagramas de temporización aparecen junto al repertorio de instrucciones del procesador. La hoja de datos también esboza los sistemas que utilizan el microprocesador.

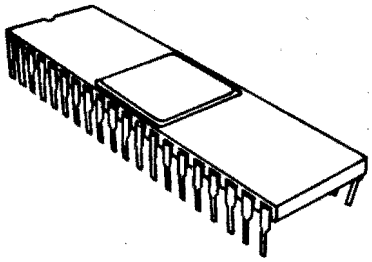
El microprocesador está ubicado normalmente en un circuito integrado *encapsulado en doble línea con 40 patillas (40-pin dual-in-line package)* (CI DIP 40 patillas). En la Figura 5.1 se muestran dos tipos de encapsulado. El DIP de 40 patillas de la Figura 5.1*a* está encapsulado en plástico, mientras que el de la Figura 5.1*b* utiliza una base cerámica. El empaquetamiento cerámico del microprocesador se aconseja para operaciones a altas temperaturas. Los microprocesadores vienen también empaquetados con 28, 42, 50 y 68 patillas. Las unidades mayores pueden ser empaquetadas en portadores de pastillas planos más modernos montados en superficie.

En las Figuras 5.1*c* y *d* se detallan dos métodos para determinar la patilla 1 del CI DIP de 40 patillas. Observar la muesca central y la ranura longitudinal que sirven como marcas de índice en el CI DIP plástico de la Figura 5.1*c*. Inmediatamente en *sentido contrario a las agujas del reloj*, de estas marcas de índice, está la patilla 1 del CI. En la Figura 5.1*d*, el punto en la parte inferior izquierda es la marca índice para mostrar qué terminal del CI es la patilla 1. Las patillas se numeran entonces en modo creciente, en sentido contrario a las agujas del reloj, en torno del CI cuando éste se mira desde la parte superior.

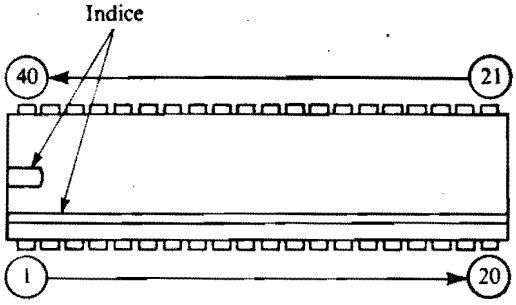
Un *diagrama de patillas,* como el de la Figura 5.2*a*, se incluye en las hojas de datos del microprocesador. El fabricante detalla además el nombre y uso de cada patilla del microprocesador. El diagrama de patillas y las breves descripciones de la Figura 5.2 corresponden al microprocesador Intel 8080. Observar que las patillas 2, 11, 20 y 28 son patillas de la fuente de alimentación. Las patillas 15 y 22 ($\varnothing 1$, $\varnothing 2$) son las entradas de reloj de un circuito de reloj externo de dos fases. Las patillas 3 a 10 del CI 8080 son bidireccionales (significa que unas veces son entradas y otras salidas), según indican las direcciones de las flechas de la Figura 5.2*a*. Estas patillas de datos ($D_0 - D_7$) son las ocho conexiones al bus de datos del sistema. El bus de direcciones del sistema de 16 bits se debe conectar a las salidas de dirección A_0 a A_{15}. Otras seis salidas *(SYNC, DBIN, WAIT,* $\overline{WR}$*, HLDA* e *INTE)* llevan las señales de control y temporización a las demás partes del sistema. Otras cuatro entradas *(READY, HOLD, INT* y *RESET)* son entradas de control que



(a) Microprocesador DIP plástico de 40 patillas

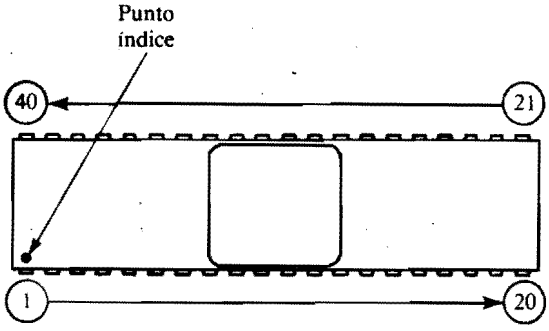


(b) Microprocesador DIP plástico de 40 patillas



(Vista superior)

(c) Numeración de las patillas del CI DIP plástico de 40 patillas



(Vista superior)

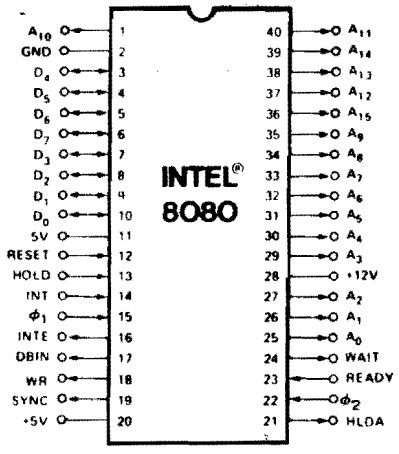
(d) Numeración de las patillas en el CI DIP cerámico de 40 patillas

Figura 5.1.

comparten la información con otras partes del sistema. La tabla de la Figura 5.2*b* da detalles adicionales sobre cada patilla del microprocesador Intel 8080.

Una hoja de datos típica también debe incluir un diagrama de la estructura del microprocesador. La Figura 5.3*a* muestra un diagrama de bloques del microprocesador Intel 8080. El diagrama de la CPU de la Figura 5.3*a* del microprocesador 8080 muestra los registros internos, incluyendo el acumulador, *B* y *C*, *D* y *E*, *H* y *L*, puntero de pila, registro de status (señalizadores) y algunos registros temporales. El diagrama de bloques también muestra el registro de instrucción y el decodificador de instrucciones, así como la sección de control y temporización. El diagrama de la CPU 8080 también muestra la ALU y los señalizadores asociados y el bloque de ajuste decimal en la Figura 5.3*a*. Las ocho entradas/salidas de datos así como las salidas de dirección de 16 bits tienen buffers. La CPU del 8080 también contiene muchas líneas de control internas, caminos de datos y buses.

Algunas hojas de datos contienen también un resumen de los registros de la CPU que son de interés para el programador. La Figura 5.3*b* muestra los registros del 8080 que utiliza el programador. Observar que el registro principal es el registro *A*, o acumulador. Los registros *B* y *C*, *D* y *E*, y *H* y *L* son unidades de propósito general. El puntero de pila, contador de programa y señalizadores son registros especializados. El registro par *HL* también puede utilizarse como registro de direcciones.



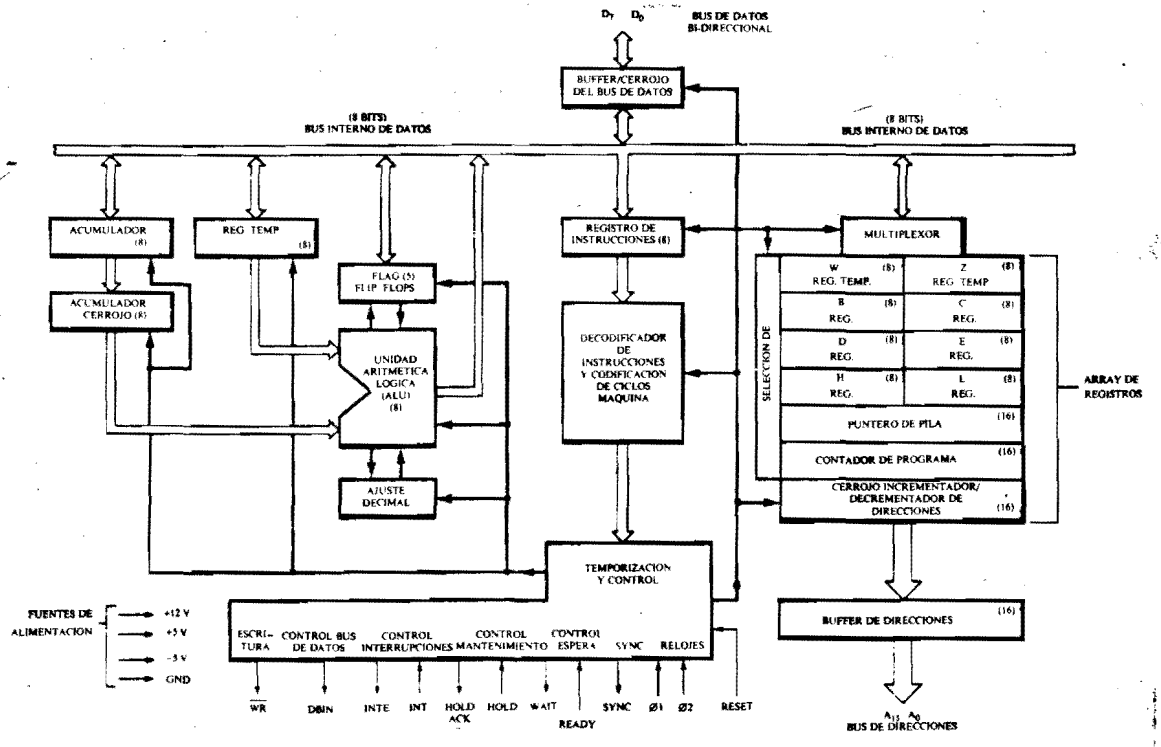
(a) **Diagrama** de patillas para el microprocesador Intel 8080
(Cortesía de Intel Corporation)

Nombre de patilla	Propósito	Entrada o salida
GND, +5 V, −5 V, +12 V	Conexiones de la fuente de alimentación	Entradas
Ø1, *Ø2*	Señales de reloj	Entradas
D_0–D_7	Líneas de datos	Bidireccional
A_0–A_{15}	Líneas de dirección	Salidas
SYNC	Sincronizador	Salida
DBIN	«Strobe» de dato de entrada	Salida
WAIT	MPU en estado de espera	Salida
$\overline{WR}$	«Strobe» de escritura	Salida
HLDA	Conocimiento de mantenimiento	Salida
INTE	Conocimiento de interrupción	Salida
READY	Entrada estable de datos	Entrada
HOLD	Petición de mantenimiento	Entrada
INT	Petición de interrupción	Entrada
REINICIALIZACION (RESET)	Reinicialización de la MPU	Entrada

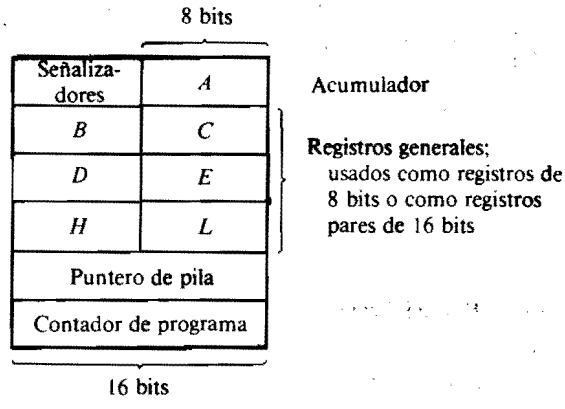
(b) Funciones de patilla para el microprocesador Intel 8080

Figura 5.2.

Las hojas de datos contienen *diagramas de temporización* elaborados que muestran las relaciones entre las entradas de reloj y otras señales externas (sync, escritura, salidas de dirección, entradas/salidas de datos, etc.) así como operaciones internas. El fabricante también incluye esquemas que ilustran cómo se utiliza el procesador en un *sistema mínimo*. Un sistema mínimo basado en el 8080 puede incluir el microprocesador, reloj, controlador del sistema, RAM, ROM e interfaces de los puertos de entrada/salida.



(a) Diagrama funcional de bloques del microprocesador Intel 8080 *(Cortesía de Intel Corporation)*



(b) Registros de programación de la CPU del Intel 8080

Figura 5.3.

Las hojas de datos del fabricante también contienen información detallada sobre el *repertorio de instrucciones* del microprocesador. El resumen del repertorio de instrucciones para los microprocesadores Intel 8080/8085 se muestra en la Figura 5.4. El Intel 8085 es una versión actualizada del microprocesador 8080 y utiliza casi el mismo repertorio de instrucciones. El procesador 8085 se estudiará con más detalle en los Capítulos 10 y 11.

Instrucción		Código	Bytes	Estados T		Ciclos máquina
				8085A	8080A	
ACI	DATO	CE dato	2	7	7	FR
ADC	REG	1000 1SSS	1	4	4	F
ADC	M	8E	1	7	7	FR
ADD	REG	1000 0SSS	1	4	4	F
ADD	M	86	1	7	7	FR
ADI	DATO	C6 dato	2	7	7	FR
ANA	REG	1010 0SSS	1	4	4	F
ANA	M	A6	1	7	7	FR
ANI	DATO	E6 dato	2	7	7	FR
CALL	ETIQ.	CD direc.	3	18	17	SRRWW*
CC	ETIQ.	DC direc.	3	9/18	11/17	SR*/SRRWW*
CM	ETIQ.	FC direc.	3	9/18	11/17	SR*/SRRWW*
CMA		2F	1	4	4	F
CMC		3F	1	4	4	F
CMP	REG	1011 1SSS	1	4	4	F
CMP	M	BE	1	7	7	FR
CNC	ETIQ.	D4 direc.	3	9/18	11/17	SR*SRRWW*
CNZ	ETIQ.	C4 direc.	3	9/18	11/17	SR*SRRWW*
CP	ETIQ.	F4 direc.	3	9/18	11/17	SR*SRRWW*
CPE	ETIQ.	EC direc.	3	9/18	11/17	SR*SRRWW*
CPI	DATO	FE dato	2	7	7	FR
CPO	ETIQ.	E4 direc.	3	9/18	11/17	SR*SRRWW*
CZ	ETIQ.	CC direc.	3	9/18	11/17	SR*SRRWW*
DAA		27	1	4	4	F
DAD	RP	00RP 1001	1	10	10	FBB
DCR	REG	00SS S101	1	4	5	F*
DCR	M	35	1	10	10	FRW
DCX	RP	00RP 1011	1	6	5	S*
DI		F3	1	4	4	F
EI		FB	1	4	4	F
HLT		76	1	5	7	FB
IN	PORT	DB dato	2	10	10	FRI
INR	REG	00SS S100	1	4	5	F*
INR	M	34	1	10	10	FRW
INX	RP	00RP 0011	1	6	5	S*
JC	ETIQ.	DA direc.	3	7/10	10	FR/FRR†
JM	ETIQ.	FA direc.	3	7/10	10	FR/FRR†
JMP	ETIQ.	C3 direc.	3	10	10	FRR
JNC	ETIQ.	D2 direc.	3	7/10	10	FR/FRR†
JNZ	ETIQ.	C2 direc.	3	7/10	10	FR/FRR†
JP	ETIQ.	F2 direc.	3	7/10	10	FR/FRR†
JPE	ETIQ.	EA direc.	3	7/10	10	FR/FRR†
JPO	ETIQ.	E2 direc.	3	7/10	10	FR/FRR†
JZ	ETIQ.	CA direc.	3	7/10	10	FR/FRR†
LDA	DIREC.	3A direc.	3	13	13	FRRR
LDAX	RP	000X 1010	1	7	7	FR
LHLD	DIREC.	2A direc.	3	16	16	FRRRR

Instrucción		Código	Bytes	Estados T		Ciclos máquina
				8085A	8080A	
LXI	RP, DATO16	00RP 0001 dato16	3	10	10	FRR
MOV	REG,REG	0100 DSSS	1	4	5	F*
MOV	M,REG	0111 0SSS	1	7	7	FW
MOV	REG,M	01DD D110	1	7	7	FR
MVI	REG,DATO	00DD D110 dato	2	7	7	FR
MVI	M,DATO	36 dato	2	10	10	FRW
NOP		00	1	4	4	F
ORA	REG	1011 0SSS	1	4	4	F
ORA	M	B6	1	7	7	FR
ORI	DATO	F6 dato	2	7	7	FR
OUT	PORT	D3 dato	2	10	10	FRO
PCHL		E9	1	6	5	S*
POP	RP	11RP 0001	1	10	10	FRR
PUSH	RP	11RP 0101	1	12	11	SWW*
RAL		17	1	4	4	F
RAR		1F	1	4	4	F
RC		D8	1	6/12	5/11	S/SRR*
RET		C9	1	10	10	FRR
RIM (8085A sólo)		20	1	4	–	F
RLC		07	1	4	4	F
RM		F8	1	6/12	5/11	S/SRR*
RNC		D0	1	6/12	5/11	S/SRR*
RNZ		C0	1	6/12	5/11	S/SRR*
RP		F0	1	6/12	5/11	S/SRR*
RPE		E8	1	6/12	5/11	S/SRR*
RPO		E0	1	6/12	5/11	S/SRR*
RRC		0F	1	4	4	F
RST	N	11XX X111	1	12	11	SWW*
RZ		C8	1	6/12	5/11	S/SRR*
SBB	REG	1001 1SSS	1	4	4	F
SBB	M	9E	1	7	7	FR
SBI	DATO	DE dato	2	7	7	FR
SHLD	DIREC.	22 direc.	3	16	16	FRRWW
SIM (8085A sólo)		30	1	4	–	F
SPHL		F9	1	6	5	S*
STA	DIREC.	32 direc.	3	13	13	FRRW
STAX	RP	000X 0010	1	7	7	FW
STC		37	1	4	4	F
SUB	REG	1001 0SSS	1	4	4	F
SUB	M	96	1	7	7	FR
SUI	DATO	D6 dato	2	7	7	FR
XCHG		EB	1	4	4	F
XRA	REG	1010 1SSS	1	4	4	F
XRA	M	AE	1	7	7	FR
XRI	DATO	EE dato	2	7	7	FR
XTHL		E3	1	16	18	FRRWW

Tipos de ciclo máquina

F Búsqueda de instr. de cuatro períodos de reloj
I Búsqueda de instr. de seis períodos de reloj
R Lectura de memoria
Lectura de E/S
W Escritura en memoria
Escritura de E/S
S Bus desocupado
X Variable o dígito binario opcional

DDD Dígitos binarios que identifican un registro destino B = 000, C = 001, D = 010 Memoria = 110
SSS Dígitos binarios que identifican un registro fuente E = 011, H = 100, L = 101 A = 111
RP Registro par BC = 00, HL = 10
DE = 01, SP = 11

* Búsqueda de instrucción de cinco períodos de reloj en 8080A.
† La secuencia más larga del ciclo máquina se aplica de la evaluación de condición en el 8080A.
● Se presentará un ciclo extra de lectura (READ)–R– para esta condición en el 8080A.

Repertorio de instrucciones para los microprocesadores Intel 8080/8085 (*Cortesía de Intel Corporation*)

Figura 5.4.

Considerar las primeras instrucciones del 8080/8085 de la Figura 5.4. La instrucción «suma con arrastre inmediato» se simboliza con el nemotécnico especial de Intel: ACI. El código de op para la instrucción es CEH, como muestra la segunda columna («Código»). La tercera columna («Bytes») indica el número de bytes de memoria del programa que necesita la instrucción. La tabla muestra que la instrucción ACI requiere 2 bytes del espacio de memoria de programa; la primera posición contendrá el código de op CEH, mientras que la segunda posición de la memoria del programa contendrá el número que se va a sumar: el dato o el operando. La columna de «Estados *T*» muestra la duración relativa del tiempo que tarda en ejecutarse la instrucción. La columna derecha de la tabla se refiere a las operaciones externas básicas que la CPU está realizando. En este caso la F significa *búsqueda normal de la instrucción* desde la memoria de programa, mientras que la R significa *lectura* de memoria. Resúmenes del repertorio de instrucciones como el de la Figura 5.4 son extremadamente valiosos para la persona que deba programar un sistema microcomputador basado en el 8080. También se incluyen en el manual de datos detalles adicionales de cada instrucción.

PROBLEMAS RESUELTOS

5.8. Citar al menos cinco cosas que contenga la hoja de datos de un microprocesador típico.

Solución:

La hoja de datos de un microprocesador típico puede contener información sobre empaquetamiento del CI, diagrama de patillas, nombre y función de cada patilla del CI, organización o arquitectura del microprocesador, diagramas de tiempo, repertorio de instrucciones y sistemas típicos que utilizan el procesador.

5.9. Cuando se mira desde arriba, la patilla 1 está inmediatamente a continuación _____ (en el sentido de las agujas del reloj, en sentido contrario de las agujas del reloj) de la marca índice en un CI DIP.

Solución:

Acudir a las Figuras 5.1*c* y *d*. Cuando se miran desde la parte superior, la patilla 1 está inmediatamente a continuación, en sentido contrario a las agujas del reloj, la marca índice.

5.10. Acudir a la Figura 5.2. Las patillas D_0 a D_7 del CI 8080 son líneas de datos _____ (de entrada, de salida, bidireccionales) que están conectadas al bus de datos del sistema.

Solución:

De acuerdo con la Figura 5.2, las patillas D_0 y D_7 en el CI 8080 son líneas de datos bidireccionales.

5.11. Acudir a la Figura 5.2. ¿Qué salida del microprocesador 8080 estará activada por un nivel BAJO durante la operación de escritura?

Solución:

Según la Figura 5.2*b*, la salida $\overline{WR}$ será activada por un nivel BAJO para indicar a un dispositivo de memoria del sistema que el dato se va a escribir en memoria.

5.12. Acudir a la Figura 5.2. ¿Qué tensión suministra la unidad de alimentación que utiliza el microprocesador 8080?

Solución:

Según la Figura 5.2, el 8080 requiere tensiones de alimentación de −5, +5 y +12 V.

5.13. Acudir a la Figura 5.3*b*. Citar como mínimo seis registros de uso general localizados en la CPU del 8080.

Solución:

Según la Figura 5.3*b*, los registros de propósito general de 8 bits de la CPU 8080 son los registros *B, C, D, E, H* y *L*. Aunque no esté etiquetado como tal, el acumulador también se considera un registro de propósito general.

5.14. Acudir a la Figura 5.3*a*. El registro contador de programa contiene _____ bits.

Solución:

Según la Figura 5.3*a*, el contador de programa contiene 16 bits (observar el pequeño 16 a la derecha del bloque contador de programa).

5.15. Los señalizadores (registro de status) están asociados con _____ (la ALU, el decodificador de instrucciones).

Solución:

Los señalizadores están más asociados con la ALU.

5.16. Acudir a la Figura 5.4. El nemotécnico de Intel para la instrucción de «suma inmediata» es ADI. El código de op del 8080 para la instrucción ADI es _____ (hex).

Solución:

Según la Figura 5.4, el nemotécnico para la instrucción «suma inmediata» es ADI y el código de op del 8080 es $C6_{16}$.

5.17. Acudir a la Figura 5.4. La instrucción ADI necesitará _____ (un número) bytes de memoria del programa.

Solución:

Según la información de la columna «Bytes» de la Figura 5.4, la instrucción ADI (instrucción de suma inmediata) necesitará 2 bytes de memoria de programa.

5.18. Acudir a la Figura 5.4. Durante la ejecución de la instrucción ADI, la CPU del 8080 va a través de un ciclo máquina de búsqueda de instrucción y un ciclo de _____ (lectura, escritura).

Solución:

Según la información de la columna «Ciclos máquina» de la Figura 5.4, la instrucción ADI hace que la CPU vaya a través de un ciclo máquina de búsqueda de instrucción y de un ciclo de lectura. La F en la columna «Ciclos máquina» significa un ciclo de búsqueda de la instrucción, mientras que la R significa un ciclo de lectura de memoria.

5.3. FUNCIONES Y DIAGRAMA DE LAS PATILLAS

En las siguientes secciones se examinará un microprocesador genérico algo más complejo. Este microprocesador tendrá muchas de las características de las unidades reales. Por simplicidad se han omitido muchas señales específicas de la MPU.

El diagrama de patillas del microprocesador genérico se muestra en la Figura 5.5*a*. La MPU está ubicada en un DIP de 40 patillas. Este CI utiliza una fuente de alimentación de 5 V conectada a las patillas 1 y 2. La utilización de fuentes de alimentación de +5 V parece lo más común en los diseños más modernos de microprocesadores.

Las patillas X_1 y X_2 en la parte superior derecha de la Figura 5.5*a* son para conectarlas a un cristal que regule la frecuencia de reloj de la MPU. Como es habitual en las unidades más modernas, la circuitería de reloj se construye dentro de la pastilla microprocesadora, mientras que los diseños más antiguos necesitaban un circuito generador de reloj separado. La salida *CLK* (patilla 38) es una señal de reloj que se utiliza como reloj del sistema. La frecuencia de la salida de *CLK* es probablemente menor que la del reloj interno.

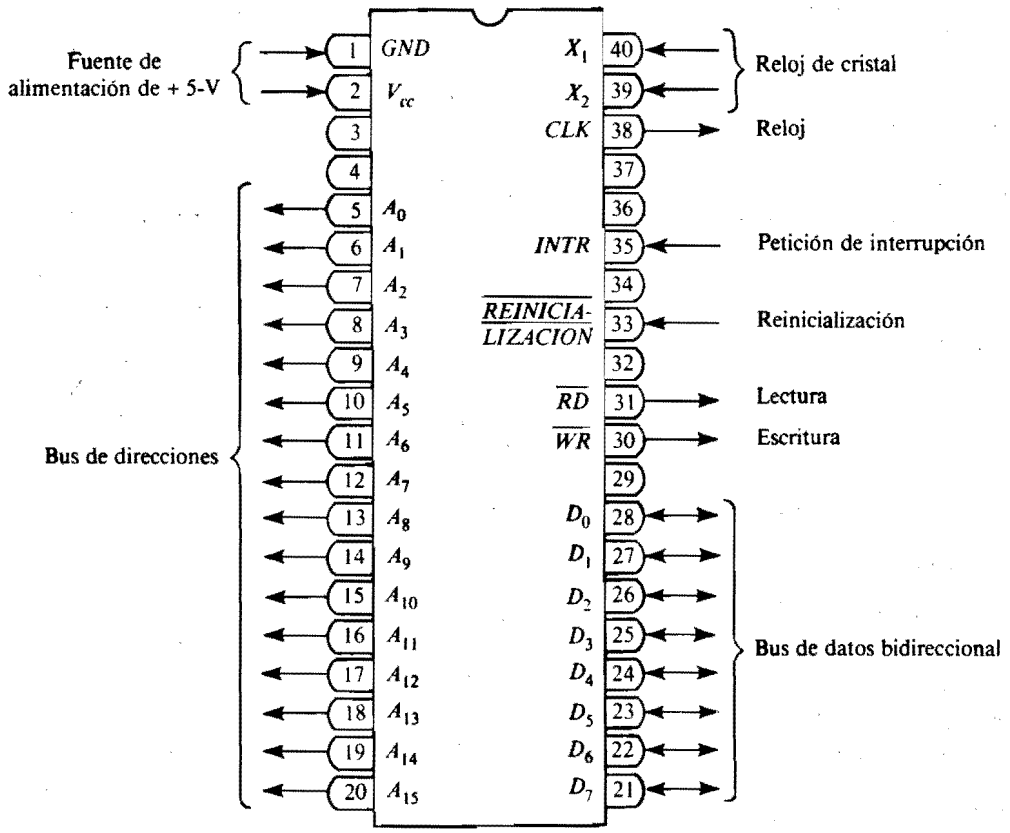
El bus de direcciones del sistema se conectará a las patillas del CI etiquetadas A_0 a A_{15} en la Figura 5.5*a*. Estas 16 salidas de dirección pueden acceder hasta 65.536 (2^{16}) posiciones de memoria y/o entradas/salidas.

El flujo de instrucciones y datos dentro y fuera del microprocesador pasa a través de las patillas etiquetadas D_0 a D_7 en el CI de la Figura 5.5*a*. Estas patillas (21-28) son *bidireccionales*, ya que unas veces sirven como entradas y otras como salidas. En general, estas patillas de datos también pueden ser colocadas en la condición de tres estados (estado de alta impedancia).

La patilla 30 del CI de la Figura 5.5*a* es la salida de *control de escritura*. Un nivel BAJO en la patilla $\overline{WR}$ indica que el dato del bus de datos se va a grabar en la posición de memoria o E/S, seleccionada. La salida de *control de lectura* está etiquetada $\overline{RD}$ (patilla 31) en el CI de la Figura 5.5*a* Un nivel BAJO en la patilla $\overline{RD}$ indica que se va a leer la memoria o el dispositivo de E/S, seleccionado y que el bus de datos está disponible para transferir el dato leído. La lectura se realiza cuando la señal $\overline{RD}$ está en el nivel BAJO.

La habilitación de la entrada de *reinicialización* en la Figura 5.5a hace que la MPU suspenda su trabajo en el programa actual y bifurque a una rutina de inicialización. Un nivel BAJO en la entrada $\overline{REINICIALIZACION}$ de la MPU pondrá en reinicialización el contador de programa con algún número predeterminado, como por ejemplo 0000_{16}. Los demás registros internos de la CPU también se pueden poner a cero o cambiar sus contenidos durante la operación de reinicialización. Cuando la entrada de $\overline{REINICIALIZACION}$ alcanza de nuevo el nivel ALTO (inhabilitado), la CPU comenzará a ejecutar instrucciones en la nueva posición de memoria 0000_{16} (u otra posición predeterminada de memoria). En esta posición de memoria comienza una *rutina de inicialización* habitualmente almacenada en la ROM. La mayoría de las acciones del microprocesador se realizan en conjunción con el reloj, y por tanto se dice que son síncronas. La entrada de $\overline{REINICIALIZACION}$ a la MPU es asíncrona y puede interrumpir y detener una instrucción a medio ejecutar.

La entrada de *petición de interrupción* al microprocesador es la patilla 35 en la Figura 5.5*a* La entrada *INTR* responde a un nivel ALTO de algún dispositivo *externo*. La Figura 5.6 ayudará a demostrar lo que ocurre cuando la MPU responde a la señal de petición de interrupción. Suponer que el circuito de interfaz de entrada de la Figura 5.6 tiene 8 bits en paralelo listos para transferirlos a la MPU. El procedimiento puede seguirse a través de los números inscritos en los círculos de la Figura 5.6 como se indica:



(a) Diagrama de patillas para un microprocesador genérico

Nombre de patilla	Propósito	Entrada o salida
GND, V_{cc}	Conexiones de la fuente de alimentación	Entrada
X_1, X_2	Conexiones de cristal para el reloj interno	Entradas
CLK	Señal del reloj	Salida
A_0–A_{15}	Bus de direcciones	Salidas
D_0–D_1	Bus de datos	Bidireccional
$\overline{RD}$	Control de salida	Salida
$\overline{WR}$	Control de escritura	Salida
INTR	Petición de interrupción	Entrada
$\overline{REINICIALIZACION}$ $(\overline{RESET})$	Reinicializa el contador de programa	Entrada

(b) Funciones de las patillas para un microprocesador genérico

Figura 5.5.

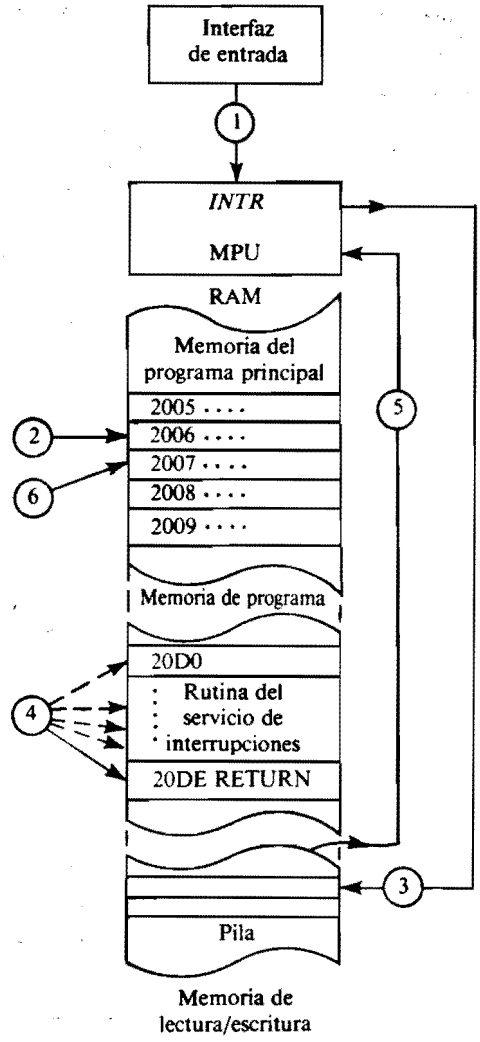


Figura 5.6. Pasos en el tratamiento de una entrada de petición de interrupción al microprocesador genérico.

1. La interfaz de entrada envía una señal de petición de interrupción a la MPU. Ver el número 1 inscrito en el círculo de la Figura 5.6.
2. La MPU completa la instrucción que está ejecutando (instrucción de la posición de memoria 2006H).
3. Debido a que el control debe volver posteriormente a la instrucción 2007H, los contenidos del contador de programa (ahora 2007H) y de la mayoría de los registros internos de la CPU son introducidos en un área especial de la RAM denominada *pila*. Estos serán devueltos posteriormente en el orden adecuado a los registros de la CPU y contador de programa.

4. La MPU ahora bifurca a una dirección predeterminada, de la memoria de programa, que contiene un programa especial denominado rutina de servicio de interrupción (en la dirección 20D0 de este ejemplo). La MPU ejecuta entonces las instrucciones de la rutina de servicio de interrupción, que en este caso es una operación de entrada. En la dirección 20DE la MPU encuentra el fin de la rutina de servicio de interrupción y se le indica que vuelva al programa principal.
5. Antes de volver al programa principal, los datos del contador de programa y registro son sacados de la pila y devueltos a la CPU.
6. El contador de programa ahora devuelve la CPU a la posición de memoria 2007H del programa principal, donde continúa.

La interrupción es un método muy útil de permitir que un dispositivo periférico obtenga la MPU para hacer algo de manera casi inmediata. La mayoría de los microprocesadores tienen una o más interrupciones. Las entradas de interrupción también pueden denominarse de reinicialización, reiniciadoras, interrupciones enmascarables o trampas.

PROBLEMAS RESUELTOS

5.19. Acudir a la Figura 5.5. Este microprocesador utiliza una fuente de alimentación de _____ V y circuitería de reloj _____ (externa, interna) con las patillas 39 y 40 conectadas a un(a) _____ (tensión ac, cristal).

Solución:

Según la Figura 5.5, este microprocesador utiliza una fuente de alimentación de +5 V y circuitería de reloj interna con las patillas 39 y 40 conectadas a un cristal.

5.20. Acudir a la Figura 5.5*a* La flecha apunta hacia fuera desde la patilla 5 porque este terminal es _____ (bidireccional, de entrada, de salida).

Solución:

La flecha apuntando hacia afuera en la Figura 5.5*a* simboliza que este terminal de dirección (patilla 5) es una salida.

5.21. Acudir a la Figura 5.5*a* La línea que deja la patilla 30 del microprocesador se considera parte del bus de _____ (direcciones, control, datos) de una microcomputadora.

Solución:

La patilla 30 de la Figura 5.5*a* es la salida de escritura, que se considera parte del bus de control de una microcomputadora.

5.22. Acudir a la Figura 5.5*a* Una señal BAJA en la patilla 33 del microprocesador _____ (inhabilitará, habilitará) la entrada de reinicialización, haciendo que la MPU ponga _____ (el contador, la memoria) de programa a 0000_{16}.

Solución:

Según la Figura 5.5, una señal BAJA en la patilla 33 habilitará la entrada de reinicialización, haciendo que la MPU ponga el contador de programa a 0000_{16}. La barra sobre *REINICIALIZACION* indica que se necesita un nivel BAJO para habilitar esta entrada.

5.23. Acudir a la Figura 5.5. La patilla de reinicialización es una entrada _____ (asíncrona, síncrona), ya que no actúa en conjunción con el reloj.

Solución:

La patilla de reinicialización en la MPU de la Figura 5.5 es una entrada asíncrona ya que no actúa en conjunción con el reloj.

5.24. Acudir a la Figura 5.5. Las transferencias de datos son realizadas con palabras en paralelo de _____ bits en este microprocesador.

Solución:

De acuerdo con la Figura 5.5*a*, el bus de datos debe ser de 8 bits, y por tanto las transferencias de datos son realizadas con palabras en paralelo de 8 bits.

5.25. Acudir a la Figura 5.5. Una señal de petición de interrupción es una _____ (entrada, salida) del microprocesador.

Solución:

Según la dirección de las flechas de la Figura 5.5, una señal de petición de interrupción es una entrada al microprocesador.

5.26. Una petición de interrupción hace que la MPU bifurque y ejecute una _____ _____ en la memoria de programa y después vuelva al programa principal.

Solución:

Según la Figura 5.6, una petición de interrupción hace que la MPU bifurque y ejecute una rutina de servicio de interrupción en la memoria de programa y después vuelva donde dejó el programa principal.

5.4. ARQUITECTURA DEL MICROPROCESADOR

Casi todos los microprocesadores contienen como mínimo lo siguiente:

1. Unidad aritmética y lógica.
2. Varios registros.
3. Contador de programa.
4. Circuitería de decodificación de instrucciones.
5. Sección de control y temporización.
6. Cerrojos y buffers de datos.
7. Líneas de control y buses internos.
8. Varias entradas y salidas de control.

Además de estos elementos, una pastilla microprocesadora puede contener también algunas de las unidades funcionales siguientes:

1. Memoria ROM.
2. Memoria RAM.
3. Puertos de entrada/salida serie.
4. Circuitería de reloj interna.
5. Temporizadores programables.

6. Circuitería de arbitración de prioridad de interrupciones.
7. Lógica de interfaz de comunicación de E/S serie a paralelo.
8. Lógica de control de acceso directo a memoria.

En la última sección se estudió el diagrama y las funciones de las patillas de un microprocesador genérico. La arquitectura de esa misma MPU se examinará en esta sección. La organización interna del microprocesador, o arquitectura, se ilustra en la Figura 5.7.

Comenzando con las conexiones externas, la MPU de la Figura 5.7 tiene ocho conexiones del bus de datos bidireccional que conducen al bus de datos interno. En la parte izquierda, la MPU tiene 16 salidas del bus de direcciones que memorizan las direcciones del bus interno. Las salidas de control se muestran en la parte inferior izquierda; y son las líneas de escritura, lectura y reloj. Dos señales de entrada son aceptadas por esta MPU en la parte inferior derecha de la Figura 5.7; son las líneas de petición de interrupción y de reinicialización. Esta MPU tiene circuitería interna de reloj que solamente necesita para que opere un cristal externo (o un capacitador en algunos

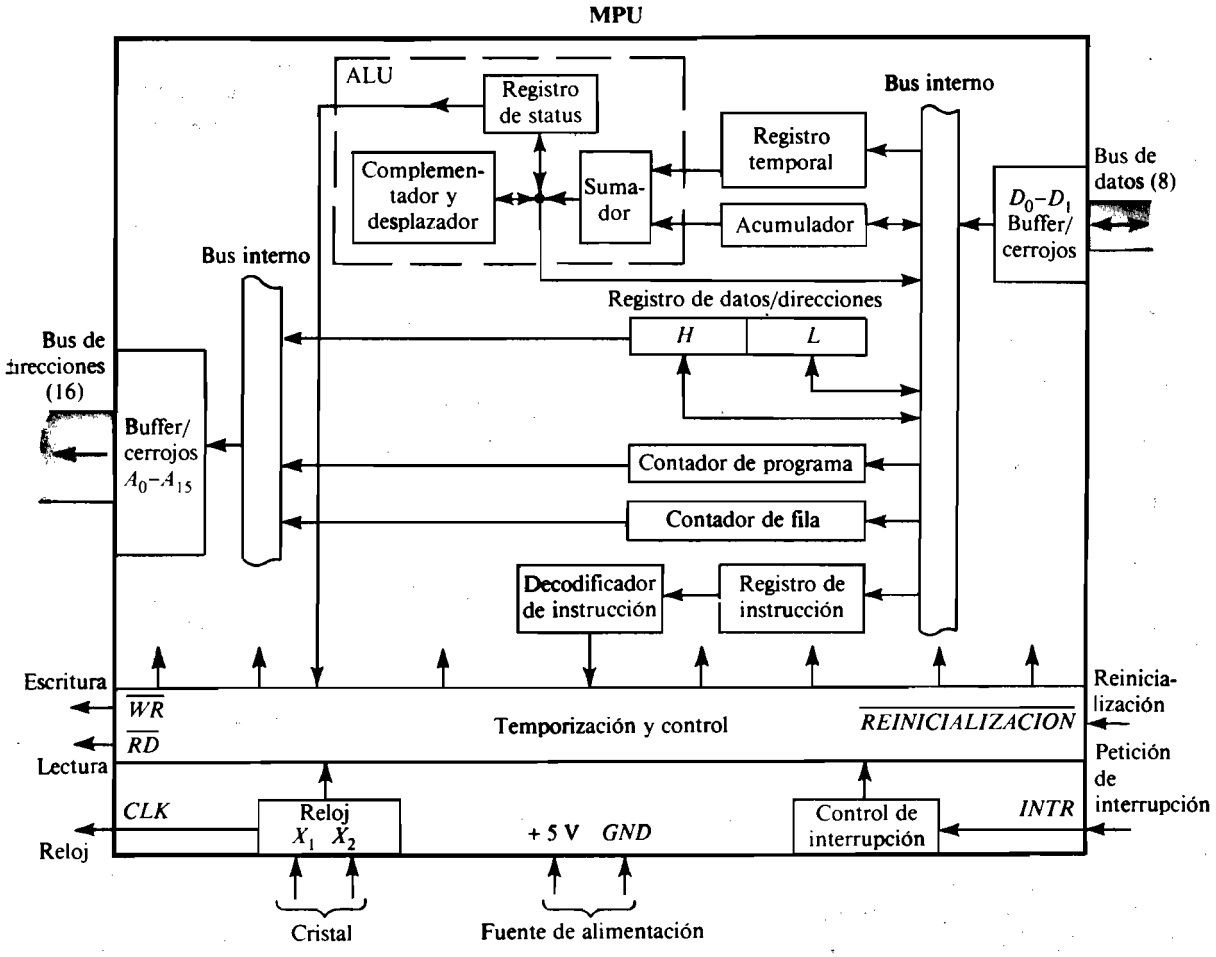


Figura 5.7. Diagrama de bloques funcionales del microprocesador genérico.

casos). El microprocesador de la Figura 5.7 está alimentado por una sola fuente de alimentación de 5 V.

Las funciones de la mayoría de las unidades de la MPU de la Figura 5.7 han sido examinadas en explicaciones anteriores. Estas funciones se definen a continuación de forma breve:

1. Registro de instrucción: Esta unidad es un registro de 8 bits que contiene el primer byte de una instrucción (el código de op).
2. Decodificador de instrucciones: Esta unidad interpreta el contenido del registro de instrucción, determina el microprograma exacto que se debe seguir para ejecutar la instrucción completa y dirige adecuadamente la sección de control.
3. Unidad aritmética y lógica: Esta unidad realiza las operaciones aritméticas, lógicas y de desplazamiento circular que afectan al registro de status (señalizadores). Los resultados de la sección de la ALU se colocan de nuevo en el acumulador vía el bus interno. El registro temporal y acumulador muchas veces se consideran parte de la ALU. Las condiciones de los señalizadores son realimentadas a la unidad de control y temporización.
4. Acumulador: Esta unidad es un registro de propósito general de 8 bits que es el foco de la mayor parte de las instrucciones aritméticas, lógicas, de carga, almacenamiento y E/S.
5. Contador de programa: Esta unidad es un área de almacenamiento de 16 bits que siempre apunta a la siguiente instrucción que se va a ejecutar. Siempre contiene una dirección de 16 bits. Puede ser incrementada o puesta a cero por la sección de control o modificada por instrucciones de transferencia.
6. Unidad de control y temporización: Esta sección recibe señales del decodificador de instrucciones para determinar la naturaleza de la instrucción que se va a ejecutar. La información del registro de status también está disponible para las bifurcaciones condicionales. Las señales de temporización y control son enviadas a todo el microprocesador para coordinar la ejecución de las instrucciones. También se generan las señales externas de control.
7. Registro de status: La MPU genérica, muy sencilla, de la Figura 5.7 contiene solamente los *señalizadores de cero* y *arrastre* en su registro de status.

Las unidades recientes añadidas a este microprocesador incluyen el reloj interno, control de interrupción, puntero de pila y registro multipropósito de direcciones/datos.

La circuitería de reloj interna, junto con el cristal externo, genera señales similares a las de la Figura 5.8. El *circuito de reloj* genera una señal de reloj de dos fases sin solapamiento para utilizarla en el interior la MPU. La salida *CLK* del MPU es similar a la señal de reloj Ø2 y sirve para sincronizar acciones en el sistema completo. En la Figura 5.8, las señales de reloj se dividen en *estados T* (T_1, T_2, etc.) y una división de tiempo mayor denominada *ciclo máquina*. Los estados *T* son siempre de una duración determinada, mientras que la longitud de un ciclo máquina puede variar. La Figura 5.8 muestra un ciclo máquina que consta de cuatro estados *T*.

La relación entre los estados *T* (T_1, T_2, etc.) y los ciclos máquina (M_1, M_2, etc.) se muestra en la Figura 5.9*a*. Los ciclos máquina se asocian a *acciones* de la MPU, como lectura, escritura, búsqueda o ejecución. En este microprocesador los *tipos de ciclo máquina* son:

1. Lectura (busca un código de op).
2. Lectura de memoria o E/S.
3. Escritura en memoria o E/S.
4. Ejecución de una operación interna.

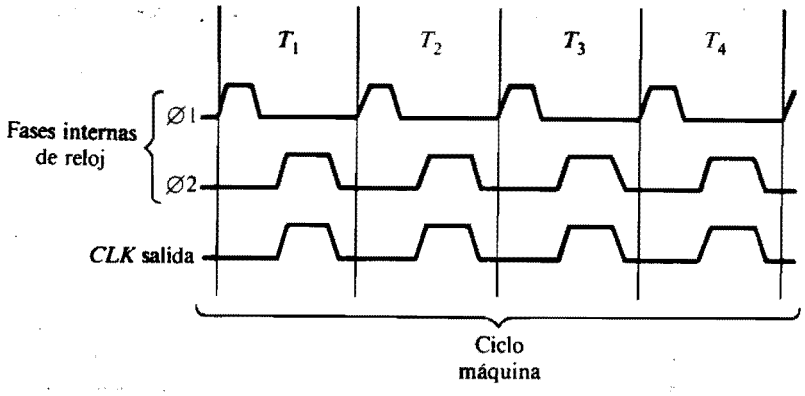


Figura 5.8. Formas de onda del reloj.

La instrucción de almacenamiento está dividida en cuatro ciclos máquina (M_1 - M_4) en la Figura 5.9*a*. Estos son: primero, la búsqueda del código de operación (lectura); segundo, la lectura de memoria de programa; tercero, otra lectura de la memoria del programa, y cuarto, la operación de escritura en memoria. La combinación completa de cuatro acciones (lectura, lectura, lectura y escritura) se denomina *ciclo de instrucción,* como muestra la Figura 5.9*a*. Observar que no todos los ciclos máquina tienen la misma duración. En la Figura 5.9*a*, el primer ciclo máquina (M_1) tiene cuatro estados *T* mientras que los otros ciclos máquina tienen solamente tres estados *T* cada uno. El ciclo completo de instrucción para la instrucción de almacenamiento tiene 13 estados *T*.

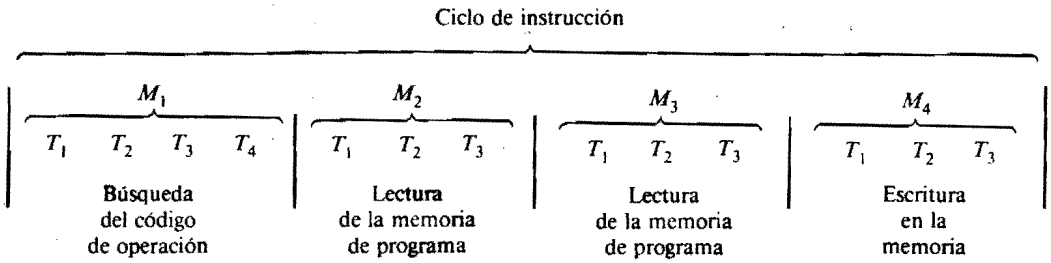
Un ejemplo de la temporización de una instrucción de suma inmediata (ADD) se muestra en la Figura 5.9*a*. El primer ciclo máquina (M_1) es la operación de búsqueda del código de op (lectura). Durante T_4 de M_1, la MPU decodifica la instrucción ADD y decide que necesita dos ciclos máquina más para completar la instrucción. El segundo ciclo máquina (M_2) lee el siguiente byte de la memoria del programa como operando, mientras M_3 ejecuta la operación de suma en la ALU del microprocesador. Observar en la Figura 5.9 que los ciclos máquina no tienen todos la misma longitud ni son ciclos de instrucción.

La *unidad de control* de la Figura 5.7 aceptará una señal de interrupción de un dispositivo externo a través de la entrada *INTR*. Entonces, guía las acciones de la MPU vía la unidad de control a través de los pasos 2 a 6 esbozados anteriormente en la Figura 5.6. De forma breve, bifurca a una rutina de servicio de interrupción que responde a la petición de interrupción y cuando se completa vuelve al programa principal.

El *puntero de pila* de la Figura 5.7 es como un contador de programa, ya que contiene una dirección, decrementa o incrementa su contenido y puede ser cargado con una nueva dirección. El puntero de pila es de 16 bits en este microprocesador, así que puede acceder a 16 líneas de dirección. El uso del puntero de pila se detallará más tarde en la Sección 5.6.

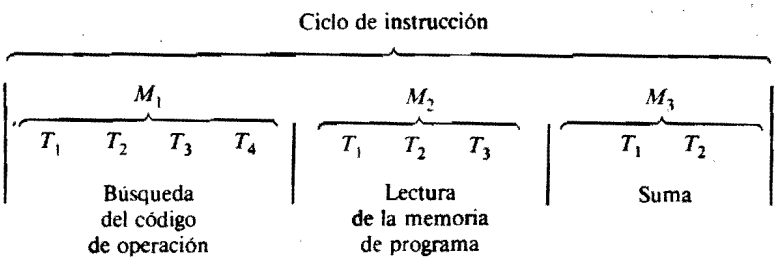
El *registro de direcciones/datos* de la Figura 5.7 realmente está formado por dos registros de 8 bits que pueden ser utilizados separadamente o como un par de registros combinados. Los dos registros de direcciones/datos de 8 bits están etiquetados con *H* y *L* (para el byte de orden superior y el de orden inferior). Habitualmente son referenciados como el registro *H* y el registro *L* a menos que se utilicen juntos, y entonces se denomina el *par de registros HL*. Los registros *HL* son registros de propósito general como el acumulador, ya que pueden ser incrementados, decremen-

Instrucción STORE (ALMACENAR)
Almacena el contenido del acumulador en la posición de memoria dada por los dos siguientes bytes de la memoria de programa



(a) Temporización y ciclos máquina para una instrucción de almacenamiento inmediato

Instrucción ADD (SUMAR)
Suma el contenido del acumulador a los del siguiente byte en la memoria del programa y deja la suma en el acumulador



(b) Temporización y ciclos máquina para una instrucción de suma inmediata

Figura 5.9.

tados y cargados con datos y pueden ser la fuente de los datos almacenados. El registro *HL* también sirve como *registro de direcciones* ya que puede contener la dirección de destino de almacenamiento o la dirección fuente cuando se cargan datos. En resumen, los registros *HL* pueden ser utilizados para *señalar direcciones.* La Sección 5.5 tratará del uso del registro de direcciones/datos. Algunos microprocesadores tienen un registro especial denominado *contador de datos* que apunta a posiciones de memoria (utilizado como el registro par *HL* de esta unidad).

PROBLEMAS RESUELTOS

5.27. Citar como mínimo seis unidades funcionales contenidas en la mayoría de las pastillas microprocesadoras.

Solución:

La mayoría de las pastillas MPU contienen como mínimo una ALU, varios registros, un contador de programa, circuitería de decodificación de direcciones, una sección de control y temporización, bus con buffers y cerrojos, buffers internos y líneas de control y varias entradas y salidas de control.

5.28. Acudir a la Figura 5.7. ¿Qué unidad funcional se encuentra entre el bus de datos y el bus interno de datos de este microprocesador?

Solución:

Un buffer y los cerrojos del bus están localizados entre el bus de datos y el bus interno de datos de la MPU genérica de la Figura 5.7.

5.29. Acudir a la Figura 5.7. ¿Qué tres líneas de control de salida ayudan a sincronizar las acciones de otras partes de un sistema basado en microprocesador?

Solución:

En la MPU de la Figura 5.7, las salidas de lectura, escritura y reloj ayudan a sincronizar las acciones de las demás partes del sistema. La mayoría de los microprocesadores tienen más señales de control.

5.30. El contador de programa es un área de memoria de 16 bits reservada para _____ (direcciones, instrucciones).

Solución:

El contador de programa es un área de memoria de 16 bits reservada para direcciones. Siempre apunta a la dirección de la siguiente instrucción que se va a ejecutar.

5.31. Acudir a la Figura 5.7. Las señales de control de lectura y escritura son generadas por la sección de _____ y _____ de la MPU.

Solución:

Las señales de control de lectura y escritura son generadas por la sección de control y temporización de esta MPU. La sección de control y temporización sigue las direcciones del decodificador de instrucciones o a veces la unidad de control de interrupciones.

5.32. Citar cuatro tipos de ciclos máquina utilizados por el microprocesador genérico que se está estudiando en esta sección.

Solución:

El microprocesador genérico utiliza los siguientes tipos de ciclos máquina: lectura o búsqueda del código de op, lectura de memoria o entrada/salida, escritura en memoria o entrada/salida y ejecución de una operación interna.

5.33. En la Figura 5.9 ¿qué instrucción se ejecuta en menor tiempo la de suma o la de almacenamiento?

Solución:

Los estados T son siempre de la misma longitud. La instrucción de la Figura 5.9 necesita solamente 9 estados T, mientras que la instrucción de almacenamiento requiere 13 estados T. Por tanto, la instrucción de suma necesita menos tiempo para ejecutarse.

5.34. Si cada estado T es igual a 500 nanosegundos (ns), ¿cuánto tiempo tardará en ejecutarse la instrucción de suma de la Figura 5.9?

Solución:

La instrucción de suma tarda nueve estados T en ejecutarse de acuerdo con el diagrama de la Figura 5.9. Por tanto, 9×500 ns = 4,5 μs. La instrucción de suma necesita 4,5 μs para ejecutarse.

5.35. Acudir a la Figura 5.7. Activar la entrada *INTR* hace que la unidad _____ _____ indique a la sección de control y temporización que bifurque temporalmente a una rutina de servicio de interrupción en la memoria del programa.

Solución:

Al activar la entrada *INTR* de la MPU de la Figura 5.7 se logra que la unidad de control de interrupciones indique a la sección de control y temporización que bifurque temporalmente a una rutina de servicio de interrupción en memoria del programa.

5.36. El puntero de pila es como el contador de programa ya que contiene una _____ (dirección, instrucción) de _____ bits.

Solución:

El puntero de pila es como el contador de programa, ya que contiene una dirección de 16 bits.

5.37. Acudir a la Figura 5.7. Las dos unidades de almacenamiento etiquetadas con *H* y *L* se denominan registro de _____/_____ .

Solución:

Las dos unidades de almacenamiento de la Figura 5.7 etiquetadas con *H* y *L* se denominan registro de datos/direcciones o registro de propósito general.

5.38. Acudir a la Figura 5.7. Cuando los dos registros de datos/direcciones se conectan entre sí y se utilizan como un área de almacenamiento de 16 bits, se denominan el registro par _____ .

Solución:

Cuando los dos registros de direcciones/datos de la Figura 5.7 se conectan entre sí y se usan como un área de almacenamiento de 16 bits, se denominan el registro par *HL*.

5.39. Acudir a la Figura 5.7. Los registros de datos/direcciones pueden ser utilizados para almacenamiento y manipulación de datos, o combinados como un par pueden ser usados para _____ _____ (señalar direcciones, seleccionar datos).

Solución:

Los registros de datos/direcciones de la Figura 5.7 pueden ser utilizados para almacenar y manipular datos, o combinados como un par pueden ser utilizados para señalar direcciones.

5.5. UTILIZACION DEL REGISTRO DE DIRECCIONES/DATOS

Una característica interesante del microprocesador genérico de la Figura 5.7 es el uso del registro par *HL* para direccionar posiciones de memoria. Se dice que el registro par *HL* se utiliza como un *puntero* o *puntero de dirección* cuando temporalmente se encarga de la tarea del contador del programa principal apuntando a las direcciones de memoria o E/S. Algunos microprocesadores muy comunes, como el 8080/8085 y Z80, tienen registros de este tipo para apuntar a las direcciones. El registro par *HL* también se denomina registro de direcciones, contador de datos, puntero, o registro de propósito general en otros microprocesadores.

Considerar la sencilla tarea de sumar los contenidos de tres posiciones consecutivas de memoria y almacenar la suma en la posición de memoria siguiente. Los contenidos de memoria de este

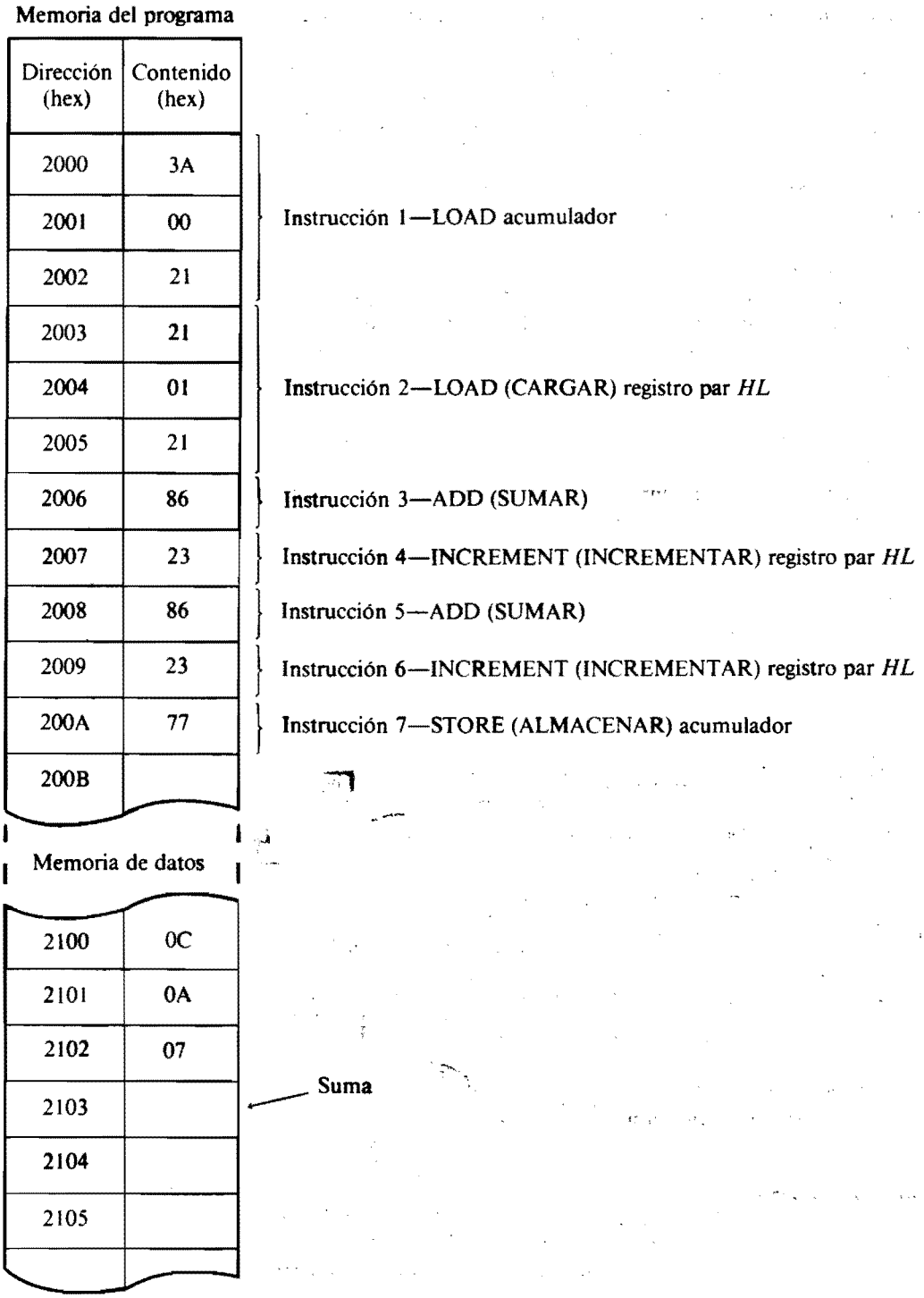


Figura 5.10. Contenidos e instrucciones de memoria para un problema ejemplo de suma.

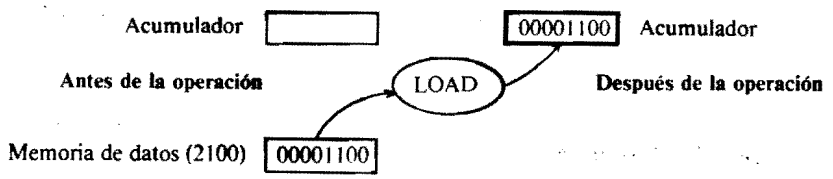
problema se detallan en la Figura 5.10. El programa se carga en las posiciones de memoria 2000H a 200AH, mientras que los tres números que se van a sumar (0CH + 0AH + 07H) se cargan en las posiciones de la memoria de datos 2100H a 2102H. El programa contiene siete instrucciones, como se lista a la derecha de la Figura 5.10. Recordar que en este programa el contenido del acumulador siempre será la suma actual y el del registro par *HL* una dirección.

La primera instrucción (código de op 3A) de la Figura 5.10 indica a la CPU que cargue el acumulador el contenido de la posición de memoria 2100H. La ejecución de esta instrucción de carga directa del acumulador se detalla en la Figura 5.11*a*, donde el contenido de la posición 2100H de la memoria de datos se carga en el acumulador de la CPU. El acumulador contiene ahora el primer número que se va a sumar.

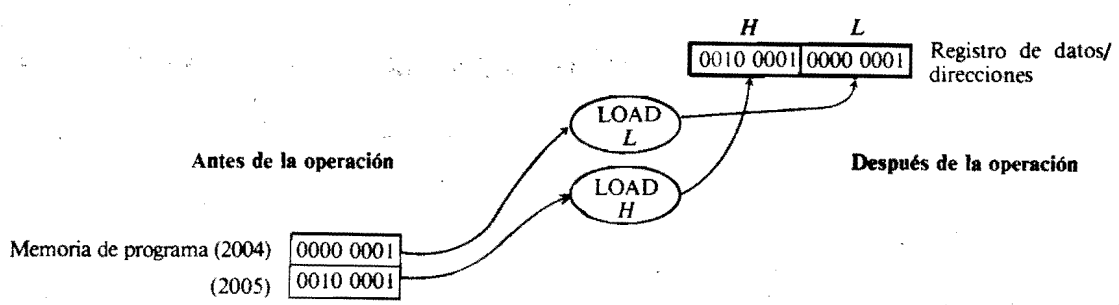
La instrucción 2 de la Figura 5.10 indica a la CPU que cargue 2101H en el registro par de 16 bits *HL*. Esta posición (2101H) es una dirección en la memoria de datos. Esta es la instrucción de «cargar inmediato el registro par *HL*». La acción de carga exacta realizada por la instrucción 2 se detalla en la Figura 5.11*b*. Observar que el contenido de la primera posición de la memoria del programa (2004H) es cargado en el byte de orden inferior (*L*) del registro *HL*. El contenido de la siguiente posición de la memoria del programa es cargado en el byte de orden superior *H* del registro par *HL*.

La instrucción 3 de la Figura 5.10 indica a la CPU que sume el contenido del acumulador con el contenido de la posición de memoria señalada por el registro par *HL*. La Figura 5.11*c* detalla el procedimiento de suma. El registro *HL* apunta a la posición 2101H de la memoria de datos, y la ALU suma su contenido (00001010_2) al contenido del acumulador (00001100_2), dando una suma de 00010110_2, que se vuelve a transferir al acumulador.

La instrucción 4 de la Figura 5.10 indica a la CPU que incremente (sume +1 a) el contenido del registro par *HL*. La acción de incrementar de la CPU se muestra gráficamente en la Figura 5.11*d*. Observar que solamente ha cambiado el byte de orden inferior en el registro par *HL*.

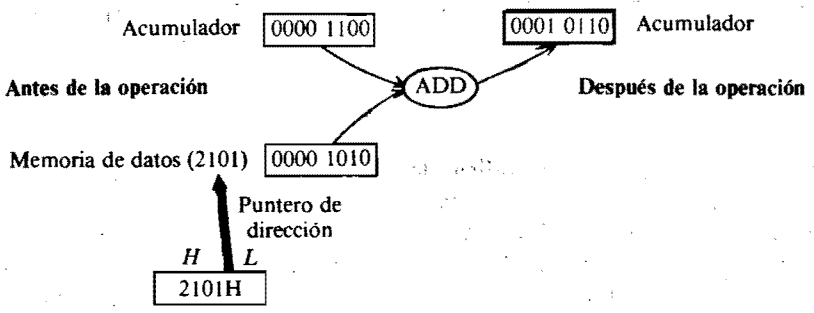


(a) Instrucción 1—cargar acumulador

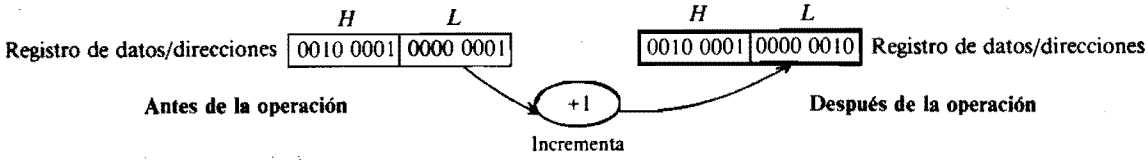


(b) Instrucción 2—cargar registro par *HL*

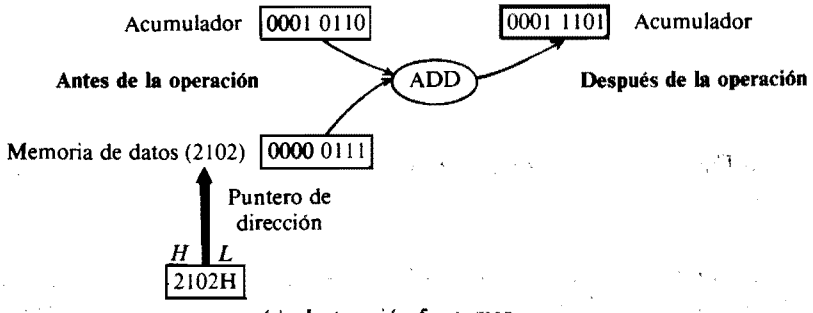
Figura 5.11.



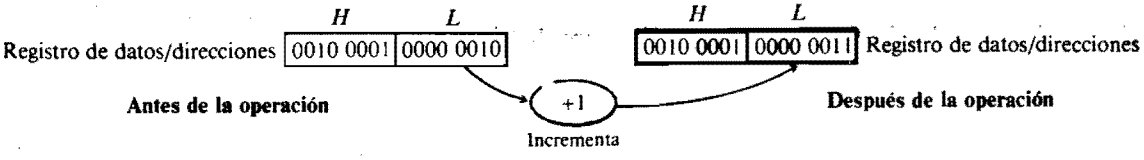
(c) Instrucción 3—**sumar**



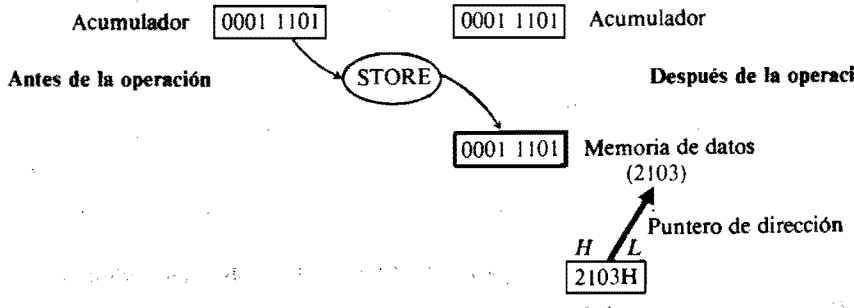
(d) Instrucción 4—incrementar registro par *HL*



(e) Instrucción 5—sumar



(f) Instrucción 6—incrementar registro par *HL*



(g) Instrucción 7—almacenar acumulador

Figura 5.11. *(continuación).*

La instrucción 5 de la Figura 5.10 indica a la CPU que sume, de nuevo, el contenido del acumulador con el de la posición de memoria señalada por el registro par *HL*. La Figura 5.11*e* muestra el registro *HL* señalando a la posición 2102H de la memoria de datos. Los contenidos de ambas posiciones de memoria 2102H y del acumulador se suman, dando la suma de 00011101_2, que se coloca en el acumulador.

La instrucción 6 de la Figura 5.10 indica, de nuevo, a la CPU que incremente el registro par *HL*. La Figura 5.11*f* detalla gráficamente esta acción.

La instrucción 7 de la Figura 5.10 indica a la CPU que almacene el contenido del acumulador (la suma 00011101_2) en la posición de memoria señalada por el registro par *HL*. La Figura 5.11*g* muestra la suma del acumulador almacenada en la posición 2103H de la memoria de datos.

Las instrucciones que usaron el registro par *HL* como puntero (instrucciones 3, 5 y 7) utilizan un modo de direccionamiento llamado *direccionamiento indirecto de registro*. Los modos de direccionamiento se tratarán con más detalle en el capítulo siguiente.

PROBLEMAS RESUELTOS

5.40. Otro nombre utilizado para el registro de direcciones es contador de _____ (datos, programa).

Solución:

Otro nombre utilizado para el registro de direcciones es contador de datos.

5.41. Acudir a la Figura 5.10. ¿Qué suma hexadecimal se transferirá a la posición 2103H de la memoria de datos una vez que se haya ejecutado el programa?

Solución:

Sumar $0C + 0A + 07 = 1D_{16}$ ($12 + 10 + 7 = 29_{10}$). La suma de $1D_{16}$ se transferirá a la posición 2103H de la memoria de datos después de que se ejecute el programa de la Figura 5.10.

5.42. Acudir a las Figuras 5.10 y 5.11. El acumulador siempre contiene la _____ (instrucción, suma), mientras que el registro par *HL* contiene una _____ (dirección, cuenta del programa).

Solución:

El acumulador de la Figura 5.10 siempre contiene la suma actual, mientras que el registro par *HL* contiene una dirección. La dirección del registro par *HL* se utiliza entonces para señalar a las posiciones de memoria.

5.43. Acudir a la Figura 5.10. Cuando se utiliza el registro *HL* como puntero de dirección igual que en la Figura 5.11*c*, ¿qué modo de direccionamiento se está utilizando?

Solución:

En la instrucción 3 se está utilizando el modo de direccionamiento de registro indirecto (instrucción de suma) de la Figura 5.10.

5.44. Acudir a la Figura 5.12. ¿Cuáles son los contenidos hexadecimales del acumulador y del registro par *HL* al principio de este programa?

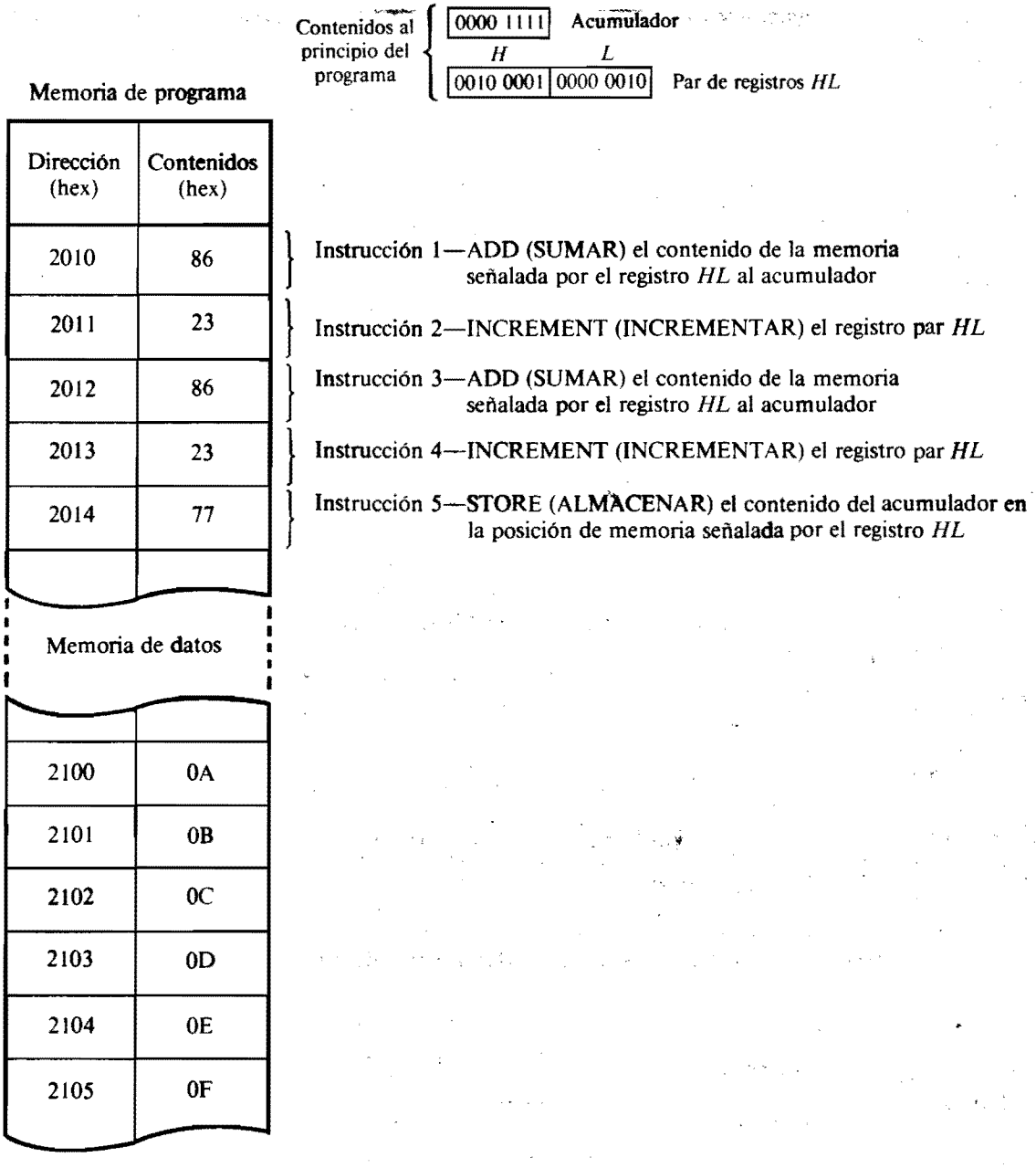


Figura 5.12. Contenidos de memoria e instrucciones para un problema de suma.

Solución:

Según la información de la Figura 5.12, el contenido del acumulador al comienzo del programa es 00001111_2, ó $0F_{16}$. El contenido del registro par *HL* al comienzo del programa es 0010 0001 0000 0010_2 ó 2102_{16}.

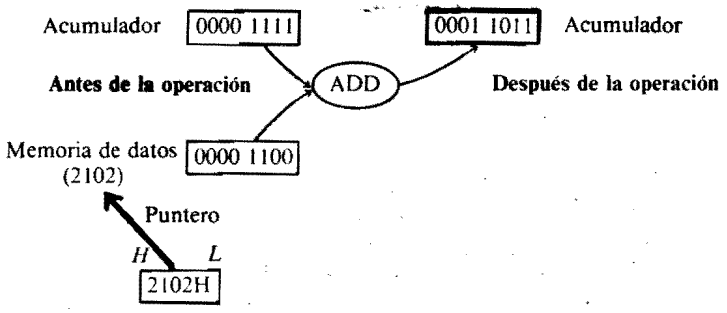


Figura 5.13.

5.45. ¿Cuál es el contenido del acumulador después de la ejecución de la instrucción 1 (instrucción de suma) de la Figura 5.12?

Solución:

Ver Figura 5.13.

5.46. ¿Cuál es el contenido del registro par *HL* después de la ejecución de la instrucción 2 (instrucción de incrementar) de la Figura 5.12?

Solución:

El registro par *HL* contiene 2102 + 1 = 2103H. Después de la instrucción 2 de la Figura 5.12, el registro par *HL* contendrá 2103H.

5.47. ¿Cuál es el contenido del acumulador después de la ejecución de la instrucción 3 (instrucción de suma) de la Figura 5.12.

Solución:

Ver Figura 5.14.

5.48. ¿Cuál es el contenido del registro par *HL* después de la ejecución de la instrucción 4 (instrucción de incrementar) en la Figura 5.12?

Solución:

El registro par *HL* contiene 2103 + 1 = 2104H. Después de la instrucción 4 de la Figura 5.12, el registro par *HL* contiene 2104H.

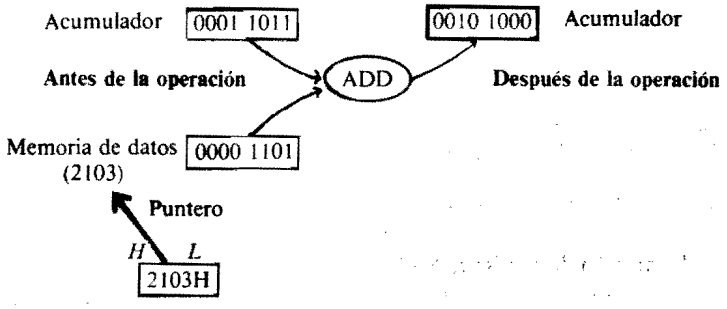


Figura 5.14.

5.49. Acudir a la instrucción 5 de la Figura 5.12. ¿Dónde estará almacenada la suma en el acumulador después de ejecutar esta instrucción de almacenamiento?

Solución:

La suma (00101000_2) estará almacenada en la posición de memoria de datos 2104H, como señala el registro par *HL*.

5.6. UTILIZACION DEL PUNTERO DE PILA

El microprocesador genérico de la Figura 5.7 contiene un puntero de pila. El *puntero de pila* es un registro contador especializado, de 16 bits, que siempre contiene una dirección. La dirección del puntero de pila es la posición de un grupo especial de posiciones de memoria de la memoria de datos conocida como *pila*. En algunas MPU la pila puede ser un grupo de áreas de almacenamiento localizadas en la pastilla microprocesadora. En la Sección 5.3 se vio que cuando una rutina de servicio de interrupción era ejecutada por la CPU, los datos actuales de los registros de la CPU debían ser almacenados temporalmente. El contenido de estos registros es almacenado en la pila. De igual forma, cuando se utiliza una subrutina regular, el contenido del contador de programa debe ser almacenado, para que la CPU pueda volver a la posición adecuada de la memoria de programa una vez que secomplete la rutina. El área de almacenamiento temporal que se utiliza es la pila. Una *subrutina* es un programa especializado, más corto (como, por ejemplo, multiplicar), que se utiliza muchas veces en el programa principal.

La pila del microprocesador genérico estará localizada en una sección de la memoria de lectura/escritura. El programador selecciona un grupo de posiciones de la RAM para que sirvan como pila. El puntero de pila es cargado con una dirección una unidad mayor que la dirección del tope de la pila, como se indica en la Figura 5.15. El puntero de pila se pone a 220AH, que es una dirección una unidad mayor que la de la primera posición de memoria de la pila que es 2209H.

Los datos pueden escribirse en la pila utilizando una instrucción de PUSH (INTRODUCIR) o de CALL (LLAMAR). Los datos pueden ser leídos de la pila utilizando una instrucción de POP (SACAR) o RETURN (VOLVER). Escribir en la pila a veces se denomina *introducir en la pila,* mientras que leer de la pila se denomina *sacar de la pila.* La pila funciona como una memoria de acceso secuencial, ya que el último dato en entrar es el primer dato en salir. Esto se denomina una memoria *último en entrar primero en salir,* o LIFO. También puede denominarse memoria de primero en entrar último en salir, o memoria FILO.

La instrucción PUSH puede producir resultados como muestra la Figura 5.16*a*, donde el contenido del registro par *HL* se introduce en la pila. Observar que el registro par *HL* de 2 bytes debe ser introducido en dos posiciones de memoria. La secuencia de acciones puede ser seguida a través de los números inscritos en los círculos.

1. El puntero de pila de la CPU se decrementa de 220AH a 2209H.
2. El puntero de pila señala al área de memoria 2209H sobre el bus de direcciones del sistema y el byte de orden superior (00000000_2) es introducido en la pila.
3. El puntero de pila es decrementado, de nuevo, de 2209H a 2208H.
4. El puntero de pila señala al área de memoria 2208H sobre el bus de direcciones del sistema, y el byte de orden inferior (00001111_2) del registro de datos/direcciones es almacenado en la pila.

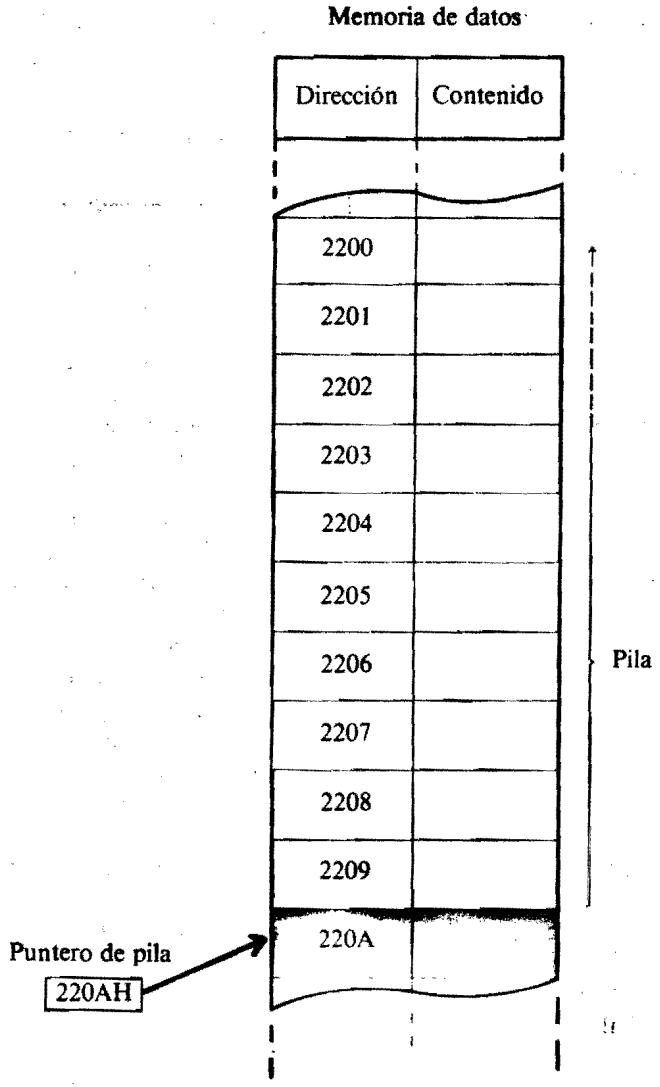
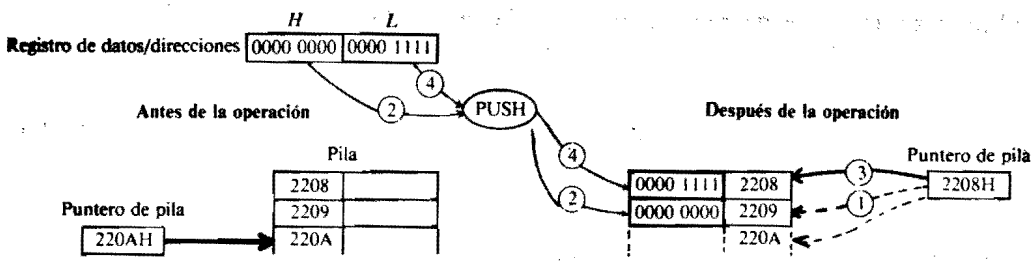


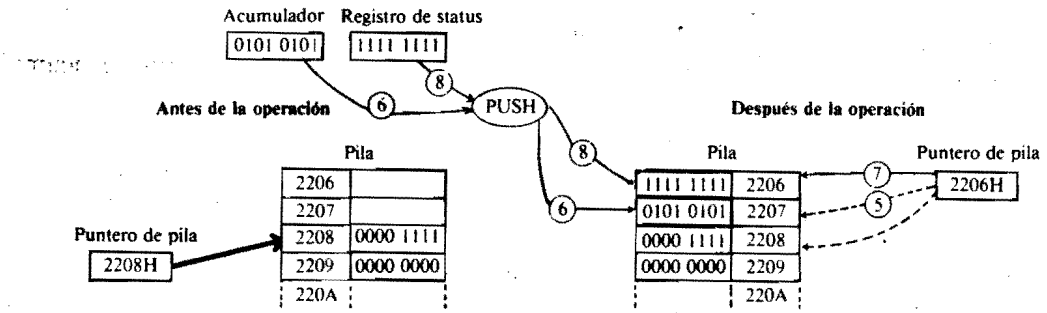
Figura 5.15. Inicialización de la pila de la RAM.

En la Figura 5.16*b* se ilustra otra operación de INTRODUCIR (PUSH). Sin embargo, esta vez los contenidos del acumulador y del registro de status se introducen en la pila. De nuevo, seguir la acción con los números inscritos en los círculos de la Figura 5.16*b*.

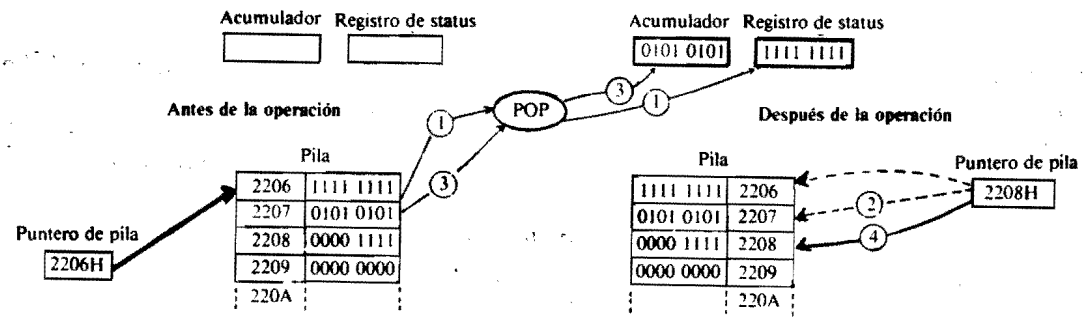
5. Antes de esta operación, el puntero de pila está señalando a la última entrada de la pila. Es decir, que está apuntando a la *parte superior de la pila.* El puntero de pila ahora se decrementa a 2207H.
6. El puntero de pila direcciona la posición 2207H de la pila, y el contenido del acumulador (01010101_2) es introducido en la pila.



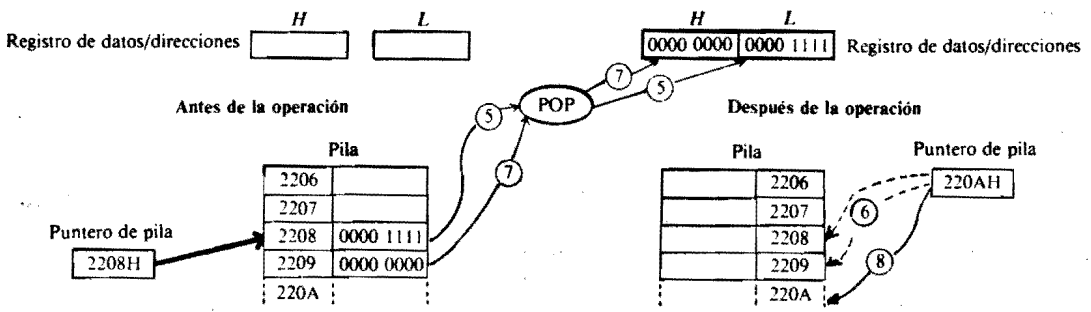
(a) Introducción del contenido del registro de datos/direcciones en la pila



(b) Introducción del contenido del acumulador y del registro de status en la pila



(c) Restauración del contenido del acumulador y del registro de status al sacarlos de la pila



(d) Restauración del contenido del registro de datos/direcciones al sacarlo de la pila

Figura 5.16.

7. El puntero de pila es decrementado de 2207H a 2206H.
8. La posición de dirección del puntero de pila es 2206H. El contenido del registro de status (11111111_2) es introducido en la parte superior de la pila.

La pila puede continuar creciendo a medida que se introduzcan más datos en la pila, lo cual se simboliza en la Figura 5.15 con la línea sombreada situada en la parte superior de la pila. La pila no tiene límites definidos excepto los impuestos por la posición de otros programas y el tamaño de la RAM.

Generalmente, por cada instrucción PUSH (INTRODUCCION) habrá más tarde una instrucción POP (SACAR) para quitar el dato de la parte superior de la pila. Como la pila es una memoria LIFO (last-in-first-out [último en entrar primero en salir]), los datos deben llegar en orden inverso. La descarga de la pila se detalla en las Figuras 5.16*c* y *d*.

Considerar la instrucción POP (SACAR) de la Figura 5.16*c*. El registro de status y el acumulador son recargados con los datos que almacenaban antes. La secuencia de acciones para sacar de la pila puede ser seguida por los números inscritos en círculos de la Figura 5.16*c*.

1. El puntero de pila direcciona la parte superior de la pila, o posición de memoria 2206H. El contenido del registro de status (11111111_2) es sacado del tope de la pila y devuelto a la ALU.
2. El puntero de pila es incrementado de 2206H a 2207H.
3. El puntero de pila direcciona la posición 2207H de la pila. Se saca el tope de la pila y el contenido es devuelto al acumulador en la CPU.
4. El puntero de la pila es incrementado a 2208H y ahora apunta a la siguiente posición que se va a sacar.

Los contenidos del registro de status y acumulador se han restaurado para que fuesen igual al que tenían antes de la operación PUSH (INTRODUCIR) de la Figura 5.16*b*.

A continuación se saca de la pila, de la Figura 5.16*d*, el contenido del registro de datos/direcciones. De nuevo, seguir la secuencia de acciones a través de los números inscritos en los círculos de la Figura 5.16*d*.

5. El puntero de pila direcciona el tope de la pila (posición 2208H). Es sacado el contenido de esta posición y devuelto al byte de orden inferior del registro par *HL*.
6. El puntero de la pila es incrementado a 2209H.
7. El puntero de la pila direcciona el tope de la pila, que ahora es la posición 2209H. El contenido (00000000_2) es devuelto al byte de orden superior del registro par *HL*.
8. Finalmente, el puntero de la pila es incrementado de 2209H a 220AH en preparación para la siguiente operación de introducción o de llamada (PUSH o CALL).

Sacar datos de la pila y restaurarlos en el registro de datos/direcciones invierte la acción de la instrucción PUSH (INTRODUCIR) ejecutada en la Figura 5.16*a*. Las instrucciones PUSH y POP son utilizadas en parejas; sin embargo, habitualmente hay otras muchas instrucciones entre estas dos que cambian los datos de los registros de la CPU.

El microprocesador genérico introducía y sacaba pares de registros. En algunas MPU, el puntero de pila puede apuntar a la posición de memoria vacía, inmediatamente encima del tope de la pila, en lugar de al tope de la pila como en el microprocesador genérico. Debe recordarse, en este ejemplo, que el programador, originalmente, pone el puntero de pila en 220AH para definir la posición de la pila de la RAM.

PROBLEMAS RESUELTOS

5.50. El puntero de pila de la Figura 5.7 es un registro contador especializado, de 16 bits, que siempre contiene _____ (una dirección, un código de op de instrucción).

Solución:

El puntero de pila siempre contiene una dirección.

5.51. La pila es una memoria especial de acceso secuencial organizada como un área de almacenamiento tipo _____ (cuatro letras).

Solución:

La pila es una memoria de acceso secuencial especial organizada como un área de almacenamiento tipo LIFO.

5.52. En el microprocesador genérico, el _____ (diseñador, programador) determina la posición de la pila en la RAM.

Solución:

El programador de la MPU determina la posición de la pila en la RAM cuando utiliza el microprocesador genérico.

5.53. Acudir a la Figura 5.15. La posición de memoria _____ (número hex) es la primera posición de la pila que será utilizada durante una operación de llamada o de introducción.

Solución:

La posición de memoria 2209H es la primera posición de la pila que será utilizada en la Figura 5.15 durante una operación de llamada o de introducción.

5.54. «Introducir en la pila» se refiere a una operación de _____ (leer, escribir) en una memoria LIFO.

Solución:

«Introducir en la pila» se refiere a una operación de escribir en una memoria LIFO.

5.55. Acudir a la Figura 5.17*a*. El paso 1 muestra que el _____ _____ se decrementa de 0009H a 0008H.

Solución:

El paso 1 de la Figura 5.17*a* muestra que se decrementa el puntero de pila.

5.56. Acudir a la Figura 5.17*a*. El paso 2 muestra que el byte de orden superior del registro de datos/direcciones se está _____ (sacando, introduciendo) en el _____ (registro *L*, pila).

Solución:

El paso 2 de la Figura 5.17*a* muestra que el byte de orden superior del registro de datos/direcciones se está introduciendo en la pila.

5.57. Acudir a la Figura 5.17*a*. El paso 4 muestra que el byte de orden inferior del registro de datos/direcciones se está introduciendo en la posición de la pila _____ (número hex).

Solución:

El paso 4 de la Figura 5.17*a* muestra que el byte de orden inferior del registro de datos/direcciones se está introduciendo en la posición 0007H.

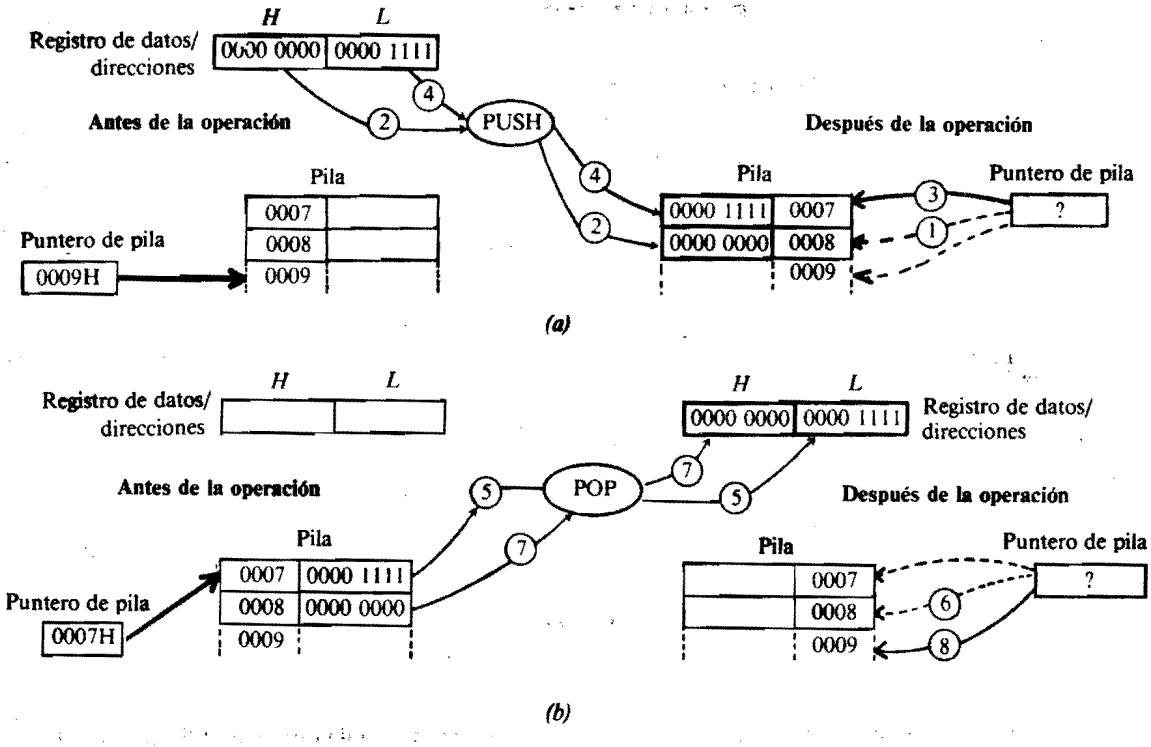


Figura 5.17.

5.58. Acudir a la Figura 5.17*b*. Antes de la operación, el puntero de pila señala el _____ (extremo inferior, tope) de la pila que actualmente es 0007H.

Solución:

Antes de la operación de la Figura 5.17*b*, el puntero de pila apunta al tope de la pila, que actualmente es 0007H.

5.59. Acudir a la Figura 5.17*b*. El paso 5 muestra que el byte de orden _____ (superior, inferior) del registro de datos/direcciones se está restaurando con el contenido del tope _____ (de la pila, del puntero de pila).

Solución:

El paso 5 de la Figura 5.17*b* muestra el byte de orden inferior del registro de datos/direcciones que se está restaurando con los contenidos del tope de la pila.

5.60. Acudir a la Figura 5.17*b*. El paso 6 muestra que el puntero de la pila se está _____ (decrementando, incrementando) 0007H a _____ (número hex).

Solución:

El paso 6 de la Figura 5.17*b* muestra que el puntero de la pila se está incrementando de 0007H a 0008H.

5.61. Acudir a la Figura 5.17*b*. El paso 7 muestra que el dato se _____ (saca, introduce) en la pila.

Solución:

La acción de lectura durante el paso 7 de la Figura 5.17*b* muestra que el dato se saca de la pila.

5.62. Acudir a las Figuras 5.16 y 5.17. El puntero de pila es _____ (decrementado, incrementado) antes de una operación PUSH, mientras que es _____ (decrementado, incrementado) después de una instrucción POP.

Solución:

Según las Figuras 5.16 y 5.17, el puntero de pila es decrementado antes de una operación PUSH, mientras que es incrementado después de una operación POP.

PROBLEMAS SUPLEMENTARIOS

5.63. Un _____ puede ser definido como una pastilla VLSI que realiza las tareas de la CPU de una microcomputadora.

Res. microprocesador.

5.64. El 8086, 8088, 68000, 65816 y Z8000 son todos ejemplos de microprocesadores de _____ bits.

Res. 16.

5.65. Las 16 líneas de dirección de los microprocesadores 8080/8085, 6800 ó 6502 pueden direccionar directamente _____ K de memoria.

Res. 64.

5.66. El registro _____ _____ sigue la pista del tope de la pila en la RAM.

Res. puntero de pila.

5.67. Es poco probable que la hoja de datos del fabricante contenga una _____ (lista de técnicas de programación, un diagrama de patillas) para el microprocesador.

Res. una lista de técnicas de programación.

5.68. Acudir a la Figura 5.2. La petición de interrupción en el microprocesador 8080 es un terminal de _____ (entrada, salida).

Res. entrada.

5.69. Acudir a la Figura 5.4. La instrucción CALL del 8080 tiene el código de op _____ (número hex), y la instrucción requiere _____ bytes de memoria de programa.

Res. CDH, 3.

5.70. Acudir a la Figura 5.5. Cuando la patilla $\overline{WR}$ está en BAJA, está _____ (habilitada, inhabilitada) y la MPU está _____ (leyendo de, escribiendo en) memoria o entrada/salida.

Res. habilitada, escribiendo en.

5.71. Acudir a la Figura 5.5. ¿Qué entrada a la unidad cambia el contador de programa a 0000H y hace que el microprocesador comience una rutina de inicialización?

Res. reinicialización.

5.72. El microprocesador típico es más probable que contenga _____ (una ALU, un área de memoria RAM).

Res. ALU.

5.73. Durante la primera lectura de memoria de programa de un ciclo de instrucción, el dato proveniente del bus de datos es transferido al _____ (acumulador, registro de instrucción).

Res. registro de instrucción (es el ciclo de máquina de búsqueda de código de op).

5.74. El microprocesador genérico tiene cuatro tipos de ciclos máquina: la búsqueda del código de op o ciclo de lectura, un ciclo de _____ de memoria, un ciclo de escritura en memoria y un ciclo de ejecución de una operación interna.

Res. lectura.

5.75. Los ciclos máquina y de instrucción varían en duración. Sin embargo, los períodos de tiempo denominados _____ _____ son de una duración dada.

Res. estados *T*.

5.76. En el microprocesador genérico, ¿qué período de tiempo es más corto: un ciclo de instrucción o un ciclo máquina?

Res. ciclo máquina.

5.77. El _____ _____ de la Figura 5.7 interpreta el contenido del registro de instrucción, decide qué microprograma interno seguir e indica a la unidad de control y temporización el procedimiento a seguir.

Res. decodificador de instrucciones.

5.78. La _____ (tres letras) del microprocesador de la Figura 5.7 realiza las operaciones de suma, resta, AND, OR y desplazamiento.

Res. ALU.

5.79. El microprocesador genérico contiene un señalizador de cero y uno de arrastre, que se encuentran en el _____ (control de interrupción, registro de status).

Res. registro de status.

5.80. El contador de programa y el registro de datos/direcciones del microprocesador genérico pueden _____ (apuntar a posiciones de memoria, almacenar instrucciones temporalmente).

Res. apuntar a posiciones de memoria.

5.81. ¿Qué tres unidades de la Figura 5.7 pueden apuntar a posiciones de memoria?

Res. contador de programa, puntero de pila y registro de datos/direcciones.

5.82. Acudir a la Figura 5.10, instrucción 2. El registro par *HL* es cargado con _____ (una dirección, un dato) en este programa.

Res. una dirección.

5.83. Acudir a las Figuras 5.10 y 5.11*d*. El registro _____ es incrementado desde _____ (número hexadecimal) a _____ (número hexadecimal) después de la operación.

Res. datos/direcciones (o *HL*), 2101H, 2102H.

5.84. Acudir a las Figuras 5.10 y 5.11*e*. La instrucción 5 es una instrucción de suma _____ (inmediata, registro indirecto).

Res. de registro indirecto.

5.85. Acudir a las Figuras 5.10 y 5.11*g*. La instrucción 7 es una instrucción de almacenamiento _____ (directo, de registro indirecto).

Res. de registro indirecto.

5.86. Acudir a la Figura 5.17*a*. El microprocesador está _____ (sacando, introduciendo) la pila en este diagrama.

Res. introduciendo.

5.87. El contenido del puntero de pila de la Figura 5.17*a después* de la operación es _____ (número hexadecimal).

Res. 0007H.

5.88. La pila es una memoria temporal especializada de acceso _____ (aleatorio, secuencial) utilizada durante las instrucciones de _____ (introducir, almacenar) y _____ (cargar, sacar).

Res. secuencial, introducir, sacar.

Capítulo 6

PROGRAMACION DEL MICROPROCESADOR

6.1. LENGUAJES MAQUINA Y ENSAMBLADOR

En el nivel más básico, el microprocesador responde a un listado de operaciones que se denomina *programa máquina.* El contenido de la memoria de programa de la Figura 6.1*a* representa un programa en código máquina, que comienza en la dirección 2000H con el código de operación 00111110_2 y finaliza en la dirección 2006H con 01110110_2. Los programas en esta forma son casi imposibles de comprender para los seres humanos.

El programa máquina de la Figura 6.1*a* puede hacerse algo más fácil de manejar si los números binarios se representan en notación hexadecimal. En la Figura 6.1*b* se repite el mismo programa en hexadecimal. El segmento del programa de la Figura 6.1*b* se considera todavía en código máquina, aún cuando el binario esté representado en notación hexadecimal. Los programas de esta forma también son muy difíciles de comprender.

En un nivel más humano, el programa anteriormente descrito en código máquina puede describirse como sigue:

1. Cargar el número binario (10110100) en el acumulador del microprocesador.
2. Complementar cada bit del número binario del acumulador para formar el complemento a 1.
3. Almacenar el resultado, del complemento a 1, en la posición 2100_{16} de la memoria de datos.

El segmento de programa descrito cambia un número binario de 8 bits a su forma equivalente de complemento a 1.

Surge la siguiente pregunta: ¿Cómo transformar la descripción del lenguaje humano del programa al código máquina? La respuesta se encuentra en un sencillo lenguaje de programación próximo al programa máquina observado en la Figura 6.1. Un *lenguaje ensamblador* utiliza palabras y frases para representar los códigos máquina del microprocesador. Generalmente, una frase o sentencia en lenguaje ensamblador constará de 1 a 3 bytes de código máquina. La idea fundamental de este proceso de ensamblamiento se muestra en la Figura 6.2*a*. Aquí, a título de ejemplo, la segunda instrucción del programa es representada por un *único nemotécnico de tres letras CMA (complementar* el *acumulador*). Las letras del nemotécnico son traducidas primero a su representación ASCII equivalente (American Standard Code for Information Interchange). Los tres códigos ASCII son evaluados entonces, en orden, por un *programa ensamblador* especial, y el código máquina para «complementar al acumulador» resulta ser 00101111_2, ó 2FH. El nemotécnico ha generado un simple byte de código de *lenguaje máquina.*

Memoria de programa

Dirección (hex)	Contenido (binario)	
2000	00111110	← Comienzo del programa
2001	10110100	
2002	00101111	
2003	00110010	
2004	00000000	
2005	00100001	
2006	01110110	
2007	·	Fin del programa
·	·	
·	·	

(a) Programa en código máquina binario

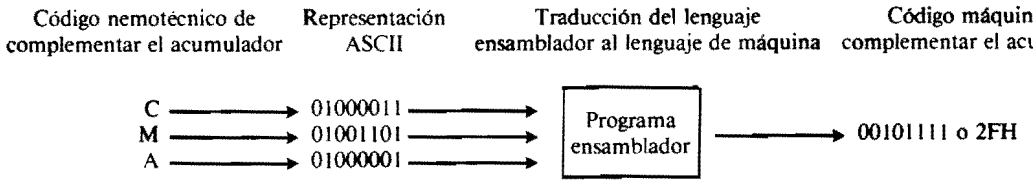
Memoria de programa

Dirección (hex)	Contenido (hex)	
2000	3E	← Comienzo del programa
2001	B4	
2002	2F	
2003	32	
2004	00	
2005	21	
2006	76	
2007	·	Fin del programa
·	·	
·	·	

(b) Programa en código máquina hexadecimal

Figura 6.1.

Un segmento de *programa en lenguaje ensamblador* escrito a mano, puede parecerse al de la Figura 6.2*a*. Se acostumbra a dividir las sentencias del lenguaje ensamblador en cuatro campos: (1) etiqueta, (2) nemotécnico, (3) operando y (4) comentarios. El campo de etiqueta no se utiliza siempre y aparece vacío en este sencillo segmento de programa. El *campo de nemotécnico* contiene el nemotécnico exacto que proporciona el fabricante. Este campo indica al programa ensamblador esencialmente la operación que se va a realizar. El *campo de operando* contiene información sobre *registros, datos* o *direcciones* asociados con la operación. Utilizando la información de los campos de nemotécnico y operando, el programa ensamblador puede generar el código correcto en lenguaje máquina. El programa ensamblador puede también asignar posiciones de la memoria de programa a los listados en código máquina. El *campo de comentario* no lo evalúa el programa ensamblador, solamente lo imprime. El campo de comentario es una ayuda valiosa para comprender qué ocurre en el programa.



(a) Traducción del nemotécnico del lenguaje ensamblador al código máquina por un programa ensamblador

Etiqueta	Nemotécnico	Operando	Comentarios
	MVI	*A*, B4H	; Carga el acumulador con el dato que sigue inmediatamente, que es B4H
	CMA		; Complementa el contenido del acumulador
	STA	2100H	; Almacena el contenido del acumulador en la posición de memoria 2100H
	HLT		; Se detiene la MPU

(b) Programa en lenguaje ensamblador

Dirección (hex)	Contenido (hex)	Etiqueta	Nemotécnico	Operando	Comentarios
2000	3E		MVI	*A*, B4H	; Carga acumulador con el dato que sigue inmediatamente, que es B4H
2001	B4				
2002	2F		CMA		; Complementa el contenido del acumulador
2003	32		STA	2100H	; Almacena el contenido del acumulador en la posición de memoria 2100H
2004	00				
2005	21				
2006	76		HLT		; Se detiene la MPU

(c) Programa en lenguaje ensamblador y máquina combinados

Figura 6.2.

Una vez que el programa ensamblador de la Figura 6.2 ha sido ensamblado, se obtiene un listado análogo al programa de la Figura 6.2*c*. Aquí se han añadido las direcciones y códigos máquina (listados bajo la columna «contenido»). En resumen, la tarea del ensamblador parece ser la de (1) traducir los nemotécnicos y operandos al código máquina equivalente y (2) asignar posiciones consecutivas de memoria a cada código de op y operando. La tarea de conseguir de la versión del lenguaje ensamblador de la Figura 6.2*b* la versión ensamblada de la Figura 6.2*c* puede hacerse manualmente o con un programa ensamblador especial.

Un programa compuesto por sentencias simbólicas, tal como el segmento de programa en lenguaje ensamblador de la Figura 6.2*b*, a veces se denomina *programa fuente*. Una vez que el programa fuente ha sido traducido (por el ensamblador o compilador) a su forma en código máquina, a ésta se denomina *programa objeto*. La programación en lenguaje ensamblador es un método de «humanizar» las operaciones de un microprocesador. Los lenguajes de alto nivel (BASIC, FORTRAN, Pascal, etc.) hacen la tarea de programar aún más sencilla. Por ejemplo, una sentencia en BASIC, FORTRAN o Pascal puede ser equivalente a unos 20 a 30 códigos máquina. El título del capítulo hace referencia a la programación de un microprocesador (opuesto a microcomputadora) puesto que el capítulo tratará de utilizar directamente el repertorio de instrucciones del microprocesador. Se utilizará la programación en lenguaje ensamblador porque ayuda a comprender las operaciones y el repertorio de instrucciones del microprocesador.

PROBLEMAS RESUELTOS

6.1. Los segmentos de programación mostrados en la Figura 6.1 se consideran programas _____ (ensambladores, máquina).

Solución:

Los segmentos de los programas mostrados en la Figura 6.1 se consideran programas máquina.

6.2. Un lenguaje _____ (ensamblador, máquina) utiliza palabras y frases para representar instrucciones de microprocesador.

Solución:

Un lenguaje ensamblador utiliza palabras y frases para representar instrucciones del microprocesador.

6.3. Un _____ (ensamblador, monitor) es un programa especial de computadora para traducir el lenguaje ensamblador al lenguaje máquina.

Solución:

Un ensamblador es un programa especial de computadora para traducir el lenguaje ensamblador al lenguaje máquina.

6.4. Listar los cuatro campos típicos de una sentencia de lenguaje ensamblador.

Solución:

Ver las etiquetas en la parte superior de la Figura 6.2*b*. Los cuatro campos de una sentencia en lenguaje ensamblador son etiqueta, nemotécnico, operando y comentarios.

6.5. Una vez que se ha ensamblado el programa ensamblador, éste contendrá las direcciones de memoria y el código _____ (BASIC, máquina) de cada sentencia del lenguaje ensamblador.

Solución:

Observar un segmento de programa ensamblado en la Figura 6.2*c*. Una vez que el programa ensamblador se ensambla, contendrá las direcciones de memoria y el código máquina de cada instrucción del lenguaje ensamblador.

6.2. REPERTORIO SIMPLIFICADO DE INSTRUCCIONES

Se presentará un repertorio de instrucciones para el microprocesador genérico ilustrado en la Figura 5.7. Los nemotécnicos y códigos de operación utilizados son un subconjunto del repertorio de instrucciones del microprocesador 8080/8085. Los derechos de copia de los nemotécnicos y de los códigos de op se utilizan con el permiso de Intel Corporation. Se pensó en utilizar nemotécnicos arbitrarios para el microprocesador genérico. Sin embargo, se decidió que utilizar un subconjunto del repertorio de instrucciones del 8080/8085 dará al estudiante un punto de partida sobre la MPU cuando se estudien los Capítulos 8 y 9. Debe mencionarse que los programas del 8080/8085 son compatibles con los sistemas 8086 y 8088.

Un resumen de los registros disponibles al programador, en el microprocesador genérico, se ilustra en la Figura 6.3. En la parte superior derecha está el acumulador (A) de propósito general de 8 bits. En la parte superior izquierda está el registro de status de 8 bits que contiene los señalizadores. En la visión expandida del registro de status, el señalizador de arrastre (CY) se muestra en la posición B_7, mientras que el señalizador de cero (Z) se localiza en la posición B_0. Las posiciones 1 a 6 no se utilizan en el registro de status del microprocesador genérico simplificado. Es costumbre que los microprocesadores reales tengan algunos señalizadores más.

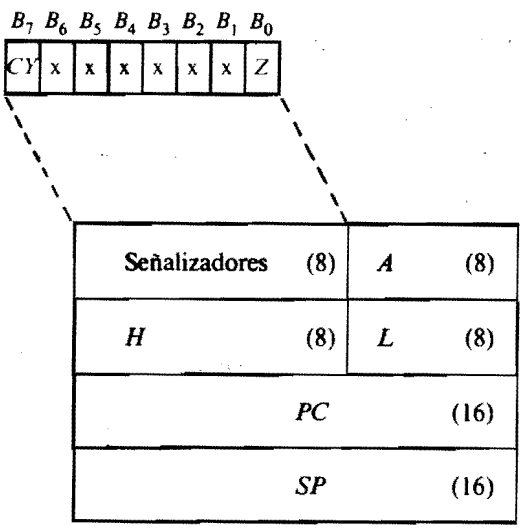


Figura 6.3. Modelo de programación para el microprocesador genérico.

En la segunda línea de la Figura 6.3 se encuentran los registros *H* y *L*. Estos son registros de datos/direcciones multipropósito. Pueden ser usados separadamente como registros de datos de propósito general o como un *registro par* (llamado el *registro par HL*). Como registro par son utilizados para apuntar o señalar direcciones.

En la parte inferior de la Figura 6.3 se localizan dos registros dedicados de 16 bits. El contador de programa (*PC*) señala la siguiente instrucción que se va a ejecutar en la CPU. El puntero de pila (*SP*) contiene la dirección del tope de la pila. En este sistema la pila está localizada en la RAM.

El repertorio de instrucciones del microprocesador genérico está dividido en las mismas siete categorías especificadas en el Capítulo 4. Estas categorías son:

1. Instrucciones aritméticas.
2. Instrucciones lógicas.
3. Instrucciones de transferencia de datos.
4. Instrucciones de bifurcación.
5. Instrucciones de llamada a subrutina.
6. Instrucciones de vuelta o de retorno.
7. Instrucciones misceláneas.

La MPU genérica puede ejecutar solamente 67 instrucciones diferentes comparadas con las 239 del repertorio completo de instrucciones del Intel 8085. Para ayudar a su estudio, solamente se considerará cada vez una categoría de instrucciones.

PROBLEMAS RESUELTOS

6.6. Las 67 operaciones ejecutadas por el microprocesador genérico son un subconjunto ¿de qué repertorio de instrucciones de una MPU real?

Solución:

Las operaciones ejecutadas por el microprocesador genérico son un subconjunto del repertorio de instrucciones de la MPU del Intel 8080/8085. Además, los programas 8080/8085 son compatibles con los sistemas de 16 bits 8086/8088.

6.7. Acudir a la Figura 6.3. Listar tres registros de 8 bits de propósito general del microprocesador genérico.

Solución:

El acumulador (*A*) y los registros de direcciones y datos *H* y *L* de la Figura 6.3 son considerados generalmente dispositivos de almacenamiento en el microprocesador genérico.

6.8. Acudir a la Figura 6.3. ¿Qué dos señalizadores contiene el registro de status en el microprocesador genérico?

Solución:

De acuerdo con la Figura 6.3, el registro de status del microprocesador genérico contiene los señalizadores de arrastre y cero.

6.9. Acudir a la Figura 6.3. Citar dos registros de 16 bits dedicados utilizados indirectamente por el programador en el microprocesador genérico.

Solución:

De acuerdo con la Figura 6.3, el microprocesador genérico contiene un contador de programa de 16 bits (*PC*) y un puntero de pila de 16 bits (*SP*).

6.3. REPERTORIO DE INSTRUCCIONES: OPERACIONES ARITMETICAS

Las *instrucciones aritméticas* son las primeras operaciones del microprocesador genérico que se van a considerar. Estas se resumen en la Figura 6.4. Incluyen las operaciones de sumar, restar, incrementar, decrementar y comparar. Observar en la Figura 6.4 que hay cuatro instrucciones de sumar. El acumulador, identificado como el registro *A* en la Figura 6.4, contiene uno de los números que se va a sumar. Cada instrucción de suma especifica una fuente diferente para el otro número que va a ser sumado.

Considerar la primera instrucción listada en la tabla de la Figura 6.4. La instrucción «sumar *A* inmediato» es una instrucción de 2 bytes cuyo formato se muestra cerca de la parte derecha de la tabla. Como siempre, el código de op (C6H en este caso) aparece en el primer byte de la memoria de programa, y el dato a ser sumado al acumulador en el segundo byte. La instrucción ADI se muestra en forma de diagrama en la Figura 6.5*a*. El dato inmediato de la memoria de programa (00010000_2) se suma al contenido del acumulador (00001111_2). La suma (00011111_2) es devuelta al acumulador después de la operación.

La segunda instrucción de la Figura 6.4 es «sumar el registro *L* a *A*» (nemotécnico ADD *L*). En la Figura 6.5*b* se muestra un problema ejemplo utilizando esta instrucción registro a registro. El contenido del acumulador (00001000_2) se suma al contenido del registro *L* (00000001_2). La suma (00001001_2) de la instrucción ADD *L* es devuelta al acumulador después de la operación.

La tercera línea de la Figura 6.4 muestra la instrucción de un solo byte «sumar *H* a *A*» (nemotécnico ADD *H*). Esta es otra instrucción de suma registro a registro. Un problema ejemplo que utiliza la instrucción ADD *H* se detalla en la Figura 6.5*c*. El contenido del registro *A* (00101100_2) se suma al contenido del registro H (00010011_2), dando una suma de 00111111_2. La suma se deposita en el acumulador una vez que se completa la operación.

La cuarta línea de la Figura 6.4 muestra la instrucción de un solo byte «sumar registro indirecto» (nemotécnico ADD *M*). La posición del dato que se va a sumar es algo más complicada utilizando este *modo de direccionamiento indirecto de registro.* Un problema ejemplo que utiliza la instrucción ADD *M* se muestra en la Figura 6.5*d*. El registro par *HL* contiene una dirección de 16 bits, que señala a la posición adecuada de la memoria de datos (LOC). El contenido de esta posición (00000011_2) se suma al contenido del acumulador (01100000_2). La suma (01100011_2) es devuelta al acumulador después de la operación de suma. Las instrucciones de registro indirecto utilizan un registro de 16 bits (habitualmente el registro par *HL*) como puntero de dirección.

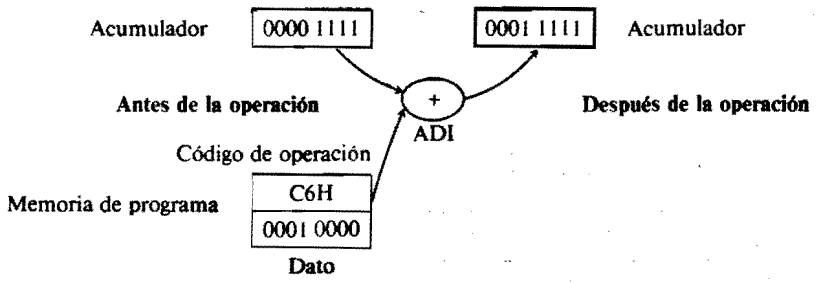
Observar de nuevo la instrucción «sumar registro indirecto» (ADD *M*) de la Figura 6.4. La descripción dice «Sumar LOC *(H&L)* a *A*» que se leerá como «sumar el contenido de la posición de memoria apuntada por el registro par *HL* al contenido del registro *A*». Al leer esta instrucción a través de la Figura 6.4, se ve que el modo de direccionamiento es indirecto de registro. El único

Instrucciones aritméticas

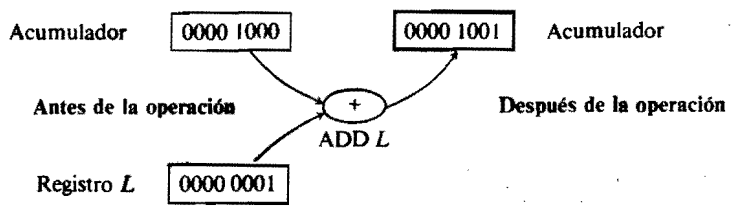
Descripción de la operación	Modo de direcciona-miento	Nemotécnico	Cód. de oper.	Bytes	Formato de la instrucción	Simbólico	Señalizadores afectados
Suma *A* al dato	Inmediato	ADI	C6	2	Código de op. datos	$(A) \leftarrow (A) + (\text{byte 2})$	*Z, CY*
Suma *L* a *A*	Registro	ADD *L*	85	1	Código de op.	$(A) \leftarrow (A) + (L)$	*Z, CY*
Suma *H* a *A*	Registro	ADD *H*	84	1	Código de op.	$(A) \leftarrow (A) + (H)$	*Z, CY*
Suma LOC (*H* y *L*) a *A*	Indirecto de registro	ADD *M*	86	1	Código de op.	$(A) \leftarrow (A) + ((H)(L))$	*Z, CY*
Resta el dato de *A*	Inmediato	SUI	D6	2	Código de op. datos	$(A) \leftarrow (A) - (\text{byte 2})$	*Z, CY*
Resta *L* de *A*	Registro	SUB *L*	95	1	Código de op.	$(A) \leftarrow (A) - (L)$	*Z, CY*
Resta *H* de *A*	Registro	SUB *H*	94	1	Código de op.	$(A) \leftarrow (A) - (H)$	*Z, CY*
Resta LOC (*H* y *L*) de *A*	Indirecto de registro	SUB *M*	96	1	Código de op.	$(A) \leftarrow (A) - ((H)(L))$	*Z, CY*
Incrementa *A*	Registro	INC *A*	3C	1	Código de op.	$(A) \leftarrow (A) + 1$	*Z*
Incrementa *HL*	Registro	INX *H*	23	1	Código de op.	$(HL) \leftarrow (HL) + 1$	
Decrementa *A*	Registro	DCR *A*	3D	1	Código de op.	$(A) \leftarrow (A) - 1$	*Z*
Decrementa *HL*	Registro	DCX *H*	2B	1	Código de op.	$(HL) \leftarrow (HL) - 1$	
Compara *A* con el dato	Inmediato	CPI	FE	2	Código de op. datos	$(A) - (\text{byte 2})$	*Z* = 1 si $(A) = (\text{byte 2})$ *CY* = 1 si $(A) < (\text{byte 2})$
Compara *A* con *L*	Registro	CMP *L*	BD	1	Código de op.	$(A) - (L)$	*Z* = 1 si $(A) = (L)$ *CY* = 1 si $(A) < (L)$
Compara *A* con *H*	Registro	CMP *H*	BC	1	Código de op.	$(A) - (H)$	*Z* = 1 si $(A) = (H)$ *CY* = 1 si $(A) < (H)$
Compara *A* con LOC (*H* y *L*)	Indirecto de registro	CMP *M*	BE	1	Código de op.	$(A) - ((H)(L))$	*Z* = 1 si $(A) = ((H)(L))$ *CY* = 1 si $(A) < ((H)(L))$

( ) contenido de
(( )) = direccionamiento indirecto de registro
+ = suma
− = resta

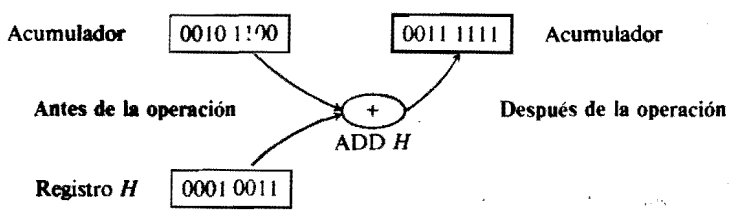
Figura 6.4. Instrucciones aritméticas para el microprocesador genérico (subconjunto de instrucciones Intel 8080/8085).



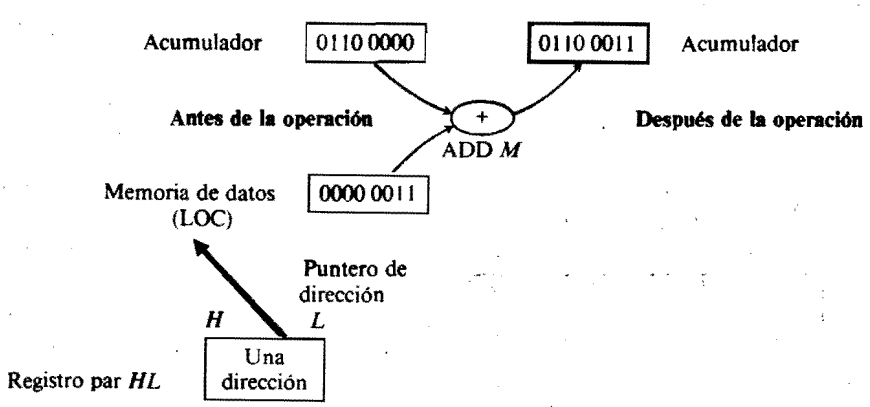
(a) La instrucción de suma *A* inmediata



(b) La instrucción de suma *L* y *A*



(c) La instrucción de suma *H* y *A*



(d) La instrucción de suma registro indirecto

Figura 6.5.

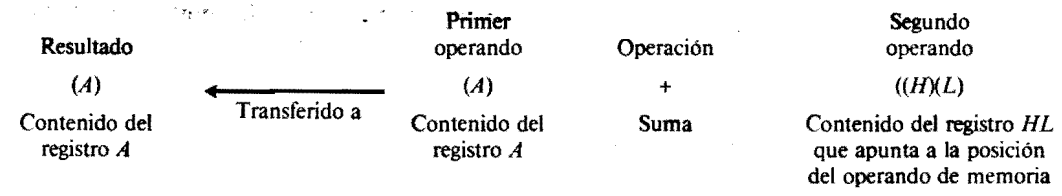
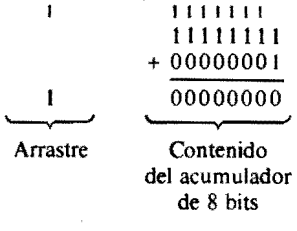


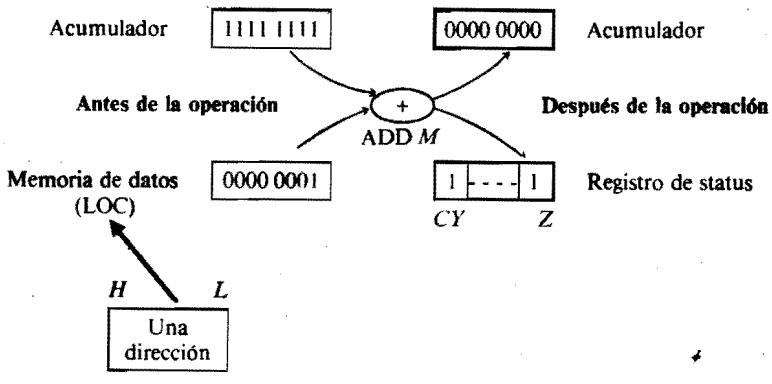
Figura 6.6. Interpretaciones de ADD; las operaciones utilizan el código simbólico de la tabla del repertorio de instrucciones.

nemotécnico del lenguaje ensamblador para esta instrucción es ADD *M*, cuyo código de op es 86_{16}. De acuerdo con la columna «Bytes» de la Figura 6.4, la instrucción «sumar registro indirecto» es una instrucción de 1 byte. El formato de instrucción muestra un solo byte que contiene el código de op de la instrucción. La segunda columna de la derecha en la Figura 6.4, detalla, en forma simbólica, la operación que está realizando la CPU. La sentencia simbólica para la instrucción ADD *M* se reescribe en la Figura 6.6. Siguiendo el patrón utilizado por Intel Corporation y cambiando algo la notación habitual, la sentencia simbólica se lee de derecha a izquierda. En la parte derecha de la flecha, el primer operando se identifica como el contenido de *A* (acumulador). Los paréntesis () significan «contenidos de» en esta notación. El signo + significa la operación que se va a realizar, que es una suma en este caso. Los dobles paréntesis (()) denotan una instrucción de registro indirecto. ((*H*)(*L*)) significa «el contenido del registro *HL* señala a la posición de memoria del segundo operando». En otras palabras, el segundo operando reside en una posición de memoria a la que apunta el contenido del registro par *HL*. Después de la operación, el resultado es transferido al registro *A*, o acumulador.

La columna de más a la derecha de la Figura 6.4 lista los señalizadores que son afectados durante la ejecución de una instrucción. Para dar un ejemplo de cómo los señalizadores son afectados por una operación, considerar la suma binaria de 11111111 con 00000001, como muestra la Figura 6.7. El método de lápiz y papel para sumar estos dos números de 8 bits se



(a) Suma de números binarios con arrastre



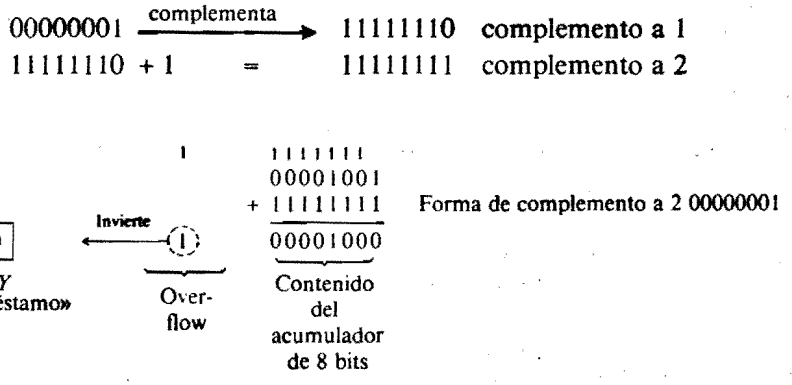
(b) El efecto de la instrucción ADD *M* sobre los señalizadores y el acumulador

Figura 6.7.

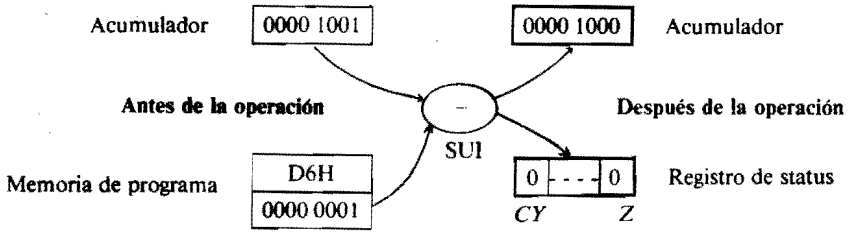
muestra en la Figura 6.7*a*. El mismo problema se muestra como es resuelto por el microprocesador genérico utilizando la instrucción ADD *M* en la Figura 6.7*b*. El acumulador contiene 11111111_2, mientras que la posición en la memoria de datos señalada por el registro par *HL* contiene el otro número que se va a sumar (00000001_2). Después de que se completa la suma, el acumulador contiene los 8 bits menos significativos de la suma, que son 00000000_2. El señalizador de arrastre del registro de status se pone a 1, indicando que hubo un arrastre del bit más significativo del acumulador. El señalizador de cero examina el contenido del acumulador después de la operación y encuentra 00000000_2. Al encontrar cero, el señalizador cero se pone a 1, indicando que el acumulador es 00000000 después de la operación de suma.

A continuación, considerar las cuatro operaciones de resta del repertorio de instrucciones del microprocesador genérico de la Figura 6.4. Cada instrucción resta el contenido de algunos registros o posiciones de memoria del contenido del acumulador. Internamente la ALU del microprocesador no tiene hardware para restar; por tanto realiza la sustracción convirtiendo el sustraendo a su forma de complemento a 2 y sumando.

Como ejemplo, considerar el problema de resta detallado en la Figura 6.8. El binario 00000001 se resta de 00001001 (09H – 01H = 08H). El procedimiento de papel y lápiz se muestra en la Figura 6.8*a*. Para revisar, la resta binaria puede realizarse *sumando* el minuendo binario a la forma en complemento a 2 del sustraendo despreciando el «overflow». El sustraendo en este problema es 00000001_2, que se convierte a su forma en complemento a 2:

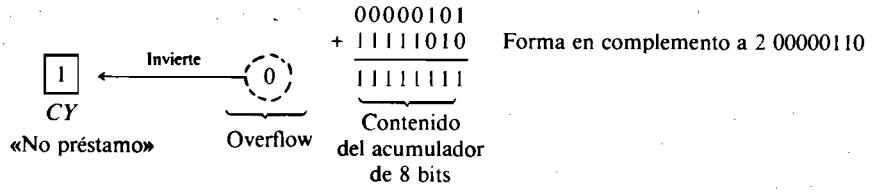


(a) Resta binaria convirtiendo el sustraendo a su forma de complemento a 2 y sumando

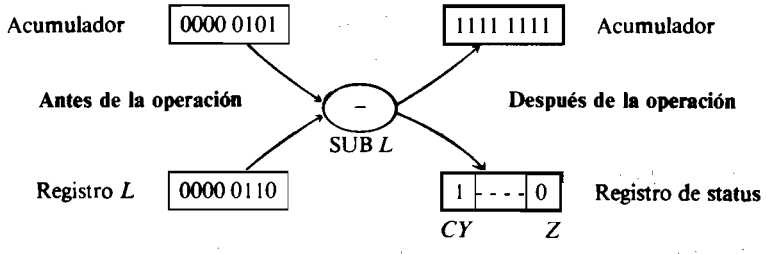


(b) Instrucción de resta inmediata

Figura 6.8.



(a) Resta binaria mostrando el efecto del overflow sobre el señalizador de arrastre



(b) La instrucción de restar *L* de *A*

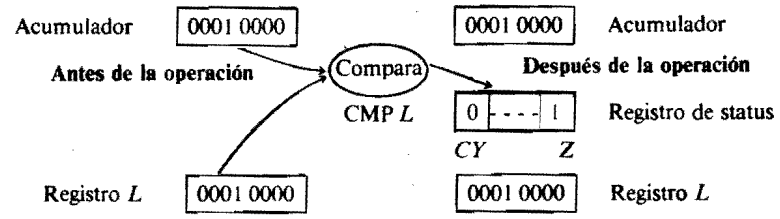
Figura 6.9.

La forma en complemento a 2 (11111111) del sustraendo se suma entonces al minuendo, dando una suma de 1 0000 1000, como se ve en la Figura 6.8*a*. El 1 de la posición más significativa de la suma es un «overflow» y no es parte de la diferencia 00001000_2. El microprocesador utiliza el overflow para afectar el señalizador de arrastre (o «préstamo» (borrow)). Al restar, la CPU invierte el overflow, y éste se convierte en el contenido del señalizador de arrastre *(CY)*. En la Figura 6.8*a*, el overflow de 1 se invierte y pone a cero el señalizador de arrastre, lo cual significa que no se ha presentado ningún préstamo o que el minuendo es mayor que el sustraendo.

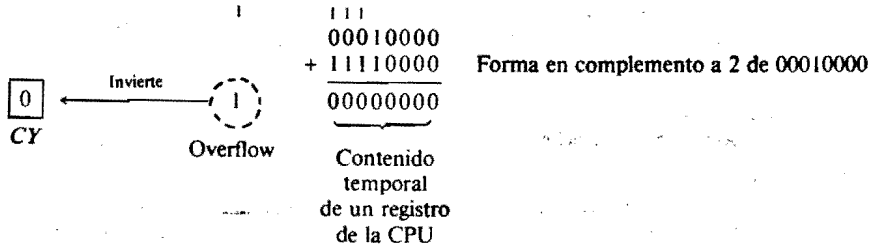
La instrucción de «resta inmediata» (nemotécnico SUI) se utiliza en el ejemplo de resta de la Figura 6.8*b*. El dato inmediato del segundo byte de la memoria del programa (00000001_2) es restado del contenido del acumulador (00001001_2). La diferencia de 00001000_2 es transferida al acumulador después de la resta. El señalizador de arrastre («préstamo») se pone a 0, indicando que no ha ocurrido ningún préstamo o que el número del acumulador antes de la operación era mayor que el número de memoria. El señalizador de cero examina el contenido del acumulador después de la operación de resta. El acumulador contiene 00001000, que es *distinto de cero,* por tanto el señalizador de cero se pone a 0.

Considerar otro problema de resta donde el minuendo es menor que el sustraendo. El problema es restar el binario 00000110 del número más pequeño 00000101 (05H − 06H = FFH o -1_{10}). El método de lápiz y papel para calcular este problema se ilustra en la Figura 6.9*a*. El sustraendo se convierte a complemento a 2 de una forma parecida a ésta:

$$00000110 \xrightarrow{\text{complementa}} 11111001 \quad \text{complemento a 1}$$
$$11111001 + 1 = 11111010 \quad \text{complemento a 2}$$



(a) La instrucción compara *A* con *L*



(b) Efecto sobre el señalizador de arrastre cuando se realiza la resta binaria durante la operación de comparación

Figura 6.10.

El minuendo (00000101_2) se suma entonces a la forma en complemento a 2 del sustraendo (11111010), dando una diferencia de 11111111. La diferencia 11111111 es la representación en complemento a 2 de -1_{10} como muestra la Figura 2.12. La Figura 6.9*a* también muestra que la suma no provoca overflow, o un 0 en el registro de overflow. Este se invierte (debido a la operación de resta), dando un 1 en la posición de señalizador *CY* o «arrastre». Cuando el señalizador *CY* se pone a 1 después de una operación de resta, significa que el número del acumulador es menor que el número del registro o posición de memoria. El señalizador de arrastre a 1 implica que el número del acumulador después de la sustracción es una representación en complemento a 2 de un número negativo. El 11111111 del acumulador de la Figura 6.9 representa -1_{10}.

El repertorio de instrucciones del microprocesador genérico tiene cuatro *instrucciones de comparación* que se muestran en las cuatro últimas líneas de la Figura 6.4. Las instrucciones de comparación restan el contenido del registro o memoria del contenido del acumulador pero *no cambian* sus contenidos. Los *señalizadores son afectados* por las instrucciones de comparación.

La instrucción «comparar el registro *L*» se utiliza como ejemplo en la Figura 6.10. Números iguales (00010000_2) en el acumulador y registro *L* son comparados por el microprocesador. Observar en la Figura 6.10*a* que ni el acumulador ni el registro *L* cambian los contenidos después de la operación de comparación. Sin embargo, son afectados los señalizadores del registro de status.

De acuerdo con la información de la columna simbólica de la Figura 6.4, el contenido del registro *L* es restado del contenido del registro *A* cuando se utiliza la instrucción «CMP *L*». La Figura 6.10*b* muestra este proceso utilizando el método de lápiz y papel. El sustraendo (contenido del registro *L*) es cambiado a su forma de complemento a 2 por el procedimiento:

$$00010000 \xrightarrow{\text{complementa}} 11101111 \quad \text{complemento a 1}$$
$$11101111 + 1 = 11110000 \quad \text{complemento a 2}$$

El minuendo (00010000_2) y sustraendo en complemento a 2 (11110000) se suman entonces, dando 1 0000 0000. Las ocho posiciones menos significativas son evaluadas para ver si son cero. Son igual a cero, por tanto el señalizador de cero se pone a 1. El overflow de 1 es invertido por la ALU, y el señalizador de arrastre es igual a 0 en este ejemplo. Un señalizador de arrastre en reinicialización ($CY = 0$) significa que el contenido del acumulador es mayor o igual que el contenido del registro *L*.

En resumen, las operaciones aritméticas que pueden ser realizadas por la MPU genérica son típicas de lo que la mayoría de los repertorios de instrucciones pueden realizar. Muchos microprocesadores tienen más instrucciones aritméticas y señalizadores en sus registros de status. Los señalizadores se utilizarán más tarde para afectar el resultado de las instrucciones de bifurcación o de salto incondicional.

PROBLEMAS RESUELTOS

6.10. Acudir a la Figura 6.4. La instrucción «resta inmediata» tiene el nemotécnico _____ y el código de op D6H y es una instrucción de _____ bytes.

Solución:

Según la quinta línea de la Figura 6.4, la instrucción de «resta inmediata» tiene el nemotécnico SUI y el código de op de D6H y es una instrucción de 2 bytes.

6.11. Acudir a la Figura 6.4. Incrementar significa _____ (sumar 1 al, restar 1 del) contenido de un registro del microprocesador.

Solución:

Incrementar significa sumar 1 al contenido de un registro del microprocesador.

6.12. Acudir a la Figura 6.4. ¿Qué instrucción de incrementar *no afecta* al señalizador cero?

Solución:

Según la información de la Figura 6.4, la instrucción INX *H* (incrementar registro par *HL*) no afecta al señalizador cero.

6.13. El contenido del acumulador de la Figura 6.11 después de la operación de decrementar es _____ (8 bits).

Solución:

La instrucción DCR *A* significa restar uno del valor del acumulador, y por tanto 00000001 − 1 = 00000000. El acumulador de la Figura 6.11 contendrá 00000000 después de la operación de decrementar.

6.14. Acudir a la Figura 6.4. El señalizador de cero de la Figura 6.11 está a _____ (0,1) después de la operación de decrementar.

Solución:

El contenido del acumulador es comprobado después de la operación de decrementar y se encuentra que vale cero; por tanto el señalizador de cero del registro de status de la Figura 6.11 se pone a 1.

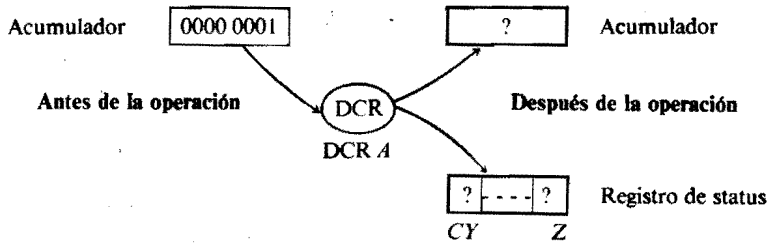


Figura 6.11.

6.15. Acudir a la Figura 6.4. ¿Cuál es la condición del señalizador de arrastre de la Figura 6.11 después de la operación de decrementar?

Solución:

De acuerdo con la Figura 6.4, el señalizador de arrastre *(CY)* no es afectado por la operación de decrementar *A*, y por tanto la condición del señalizador de arrastre del registro de status es impredecible.

6.16. Acudir a la Figura 6.4. Listar el contenido del acumulador y de dos posiciones de la memoria de programa después de la operación de «comparar inmediato» de la Figura 6.12.

Solución:

Las instrucciones de **comparación no cambian** el contenido de los registros o posiciones de memoria que se compara. Por tanto, el acumulador contendrá todavía 00001111 después de la operación de comparación en la Figura 6.12. El código de op para la instrucción CPI es FE_{16} (ver la Figura 6.4), mientras que el dato inmediato de la memoria de programa es todavía 00010000.

6.17. Acudir a la Figura 6.4. El señalizador de cero de la Figura 6.12 estará a _____ (0, 1) después de la operación de «comparación inmediata».

Solución:

El señalizador cero de la **Figura 6.12** estará en **reinicialización** después de la operación de comparación inmediata. La columna simbólica de la Figura 6.4 muestra que la instrucción de comparación es básicamente una operación de resta donde se evalúa la diferencia. En este caso la diferencia no es cero, por tanto el señalizador de cero no alcanzará la reinicialización.

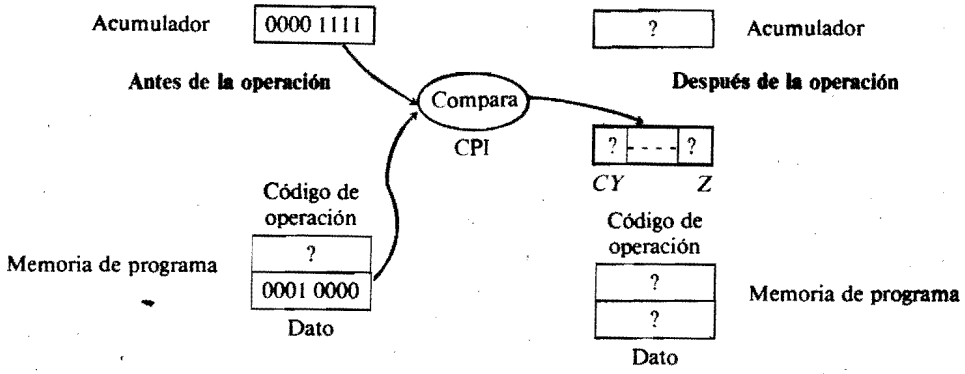


Figura 6.12.

6.18. Acudir a la Figura 6.4. El señalizador de arrastre de la Figura 6.12 estará a _____ (0, 1) después de la operación de comparación inmediata.

Solución:

El señalizador de arrastre de la Figura 6.12 estará a 1 después de la operación de comparación inmediata porque el resultado de la resta interna de la CPU detecta que el contenido del acumulador es menor que el contenido de la sección del dato de la memoria del programa. Recordar que el señalizador de arrastre cuando se resta es realmente un señalizador de «préstamo».

6.19. Acudir a la Figura 6.4. El contenido del registro par *HL* de la Figura 6.13 después de la operación de incrementar es _____ (hex).

Solución:

Incrementar el contenido del registro par *HL* significa 1111 1111 1111 1111 + 1 = = 1 0000 0000 0000 0000. El bit más significativo de la suma es llevado al extremo izquierdo del registro *H*. Por tanto, el contenido en el registro par *HL* después de la operación será 0000 0000 0000 0000, ó 0000_{16}.

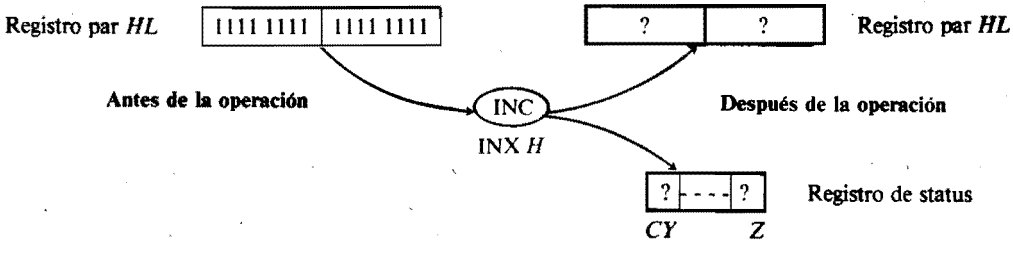


Figura 6.13.

6.20. Acudir a la Figura 6.4. ¿Cuál es la condición de los señalizadores de la Figura 6.13 después de la instrucción de incrementar el registro par *HL*?

Solución:

De acuerdo con la Figura 6.4, los señalizadores no son afectados por los resultados de la instrucción de incrementar el registro par *HL*, y por tanto el contenido del registro de status será impredecible. Observar que las operaciones aritméticas sobre el contenido del acumulador afectan a los señalizadores, mientras que esta operación se realizaba sobre el contenido del registro par *HL*.

6.21. Los señalizadores son afectados por la mayoría de las operaciones aritméticas pero son utilizados por el microprocesador durante la ejecución de instrucciones _____ (de bifurcación, lógicas).

Solución:

Los señalizadores son afectados por la mayoría de las operaciones aritméticas, pero lo utiliza el microprocesador durante las instrucciones de bifurcación (condicional).

6.4. REPERTORIO DE INSTRUCCIONES: OPERACIONES LOGICAS

Las *instrucciones lógicas* son las segundas operaciones del microprocesador genérico que se van a considerar. Estas están resumidas en la Figura 6.14. Incluyen las instrucciones AND, OR, OR exclusiva, complementación (NOT) e instrucciones de desplazamiento circular. La organización de la tabla de la Figura 6.14 es la misma que la utilizada para las operaciones aritméticas anteriores. De nuevo, el foco de la mayor parte de las operaciones está en el contenido del acumulador. Como con las instrucciones aritméticas, el modo de direccionamiento indica cómo y dónde se encuentra el otro dato en el sistema basado en microprocesador.

Considerar el uso de la instrucción «AND inmediato» del microprocesador genérico, como muestra la Figura 6.15*a*. El contenido del acumulador (00010011) realiza la operación AND con el segundo byte de datos de la memoria del programa (00000001). La operación AND se realiza bit a bit. Según la tabla de verdad de la Figura 3.1, sólo el bit menos significativo de cada número es un 1; por tanto la salida será 00000001. El resultado de la operación AND se coloca en el acumulador. De acuerdo con la última columna de la Figura 6.14, todas las operaciones AND hacen que el señalizador de arrastre se ponga a cero, lo cual también se pone de manifiesto en la Figura 6.15*a*. Se comprueba si es cero el resultado de la operación AND y como no lo es, el señalizador de cero está a 0. Observar que el punto (·) de la Figura 6.14 debajo de la columna «Simbólico» denota la operación AND.

En la Figura 6.15*b* se ilustra un segundo ejemplo que utiliza una instrucción AND. Aquí se usa la instrucción «AND registro indirecto» (nemotécnico ANA *M*). El contenido del acumulador realiza la operación AND (bit a bit) con el contenido de la posición de la memoria de datos señalada por el registro par *HL*. Al realizar la operación AND entre 00111100 y 00000001, se encuentra que el resultado de salida es 00000000, que se almacena en el acumulador. El señalizador de arrastre se pone a cero según la información de la Figura 6.14. Se comprueba el contenido del acumulador (00000000) y al ser cero se pone a 1 el señalizador de cero.

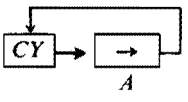
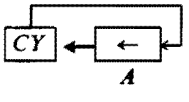
Observar cuidadosamente las Figuras 6.15*a* y *b*. En ambos ejemplos el segundo operando es 00000001. Este operando se utiliza como una *máscara*. En las Figuras 6.15*a* y *b* la máscara 00000001 puede utilizarse para poner a cero los siete bits más significativos. O posiblemente, se utilice el único 1 de la máscara 00000001 junto con el señalizador de cero, para comprobar la presencia de un 0 ó un 1 en la posición LSB del contenido del acumulador. Cuidado: Observar que el contenido del acumulador cambia cuando se utilizan instrucciones AND. Algunos microprocesadores tienen «instrucciones especiales de 'test' de bits», que realizan la operación AND del acumulador con un byte de máscara sin cambiar el contenido del acumulador mientras cambian los señalizadores.

Las cuatro instrucciones OR listadas en la Figura 6.14 realizan la operación OR del contenido del acumulador con el de otros registros o posiciones de memoria. Un ejemplo de una operación OR se muestra en la Figura 6.16. Aquí el contenido del acumulador es 11001100, mientras que el registro *L* contiene 00001111; el resultado de la operación OR es 11001111. La operación OR se realiza bit a bit según la tabla de verdad de la Figura 3.1. El 00001111 del registro *L* puede considerarse una máscara que siempre pone a 1 los 4 bits menos significativos. Observar el uso del símbolo lógico OR (+) en la columna simbólica de la Figura 6.14.

Las dos últimas operaciones lógicas listadas en la tabla de la Figura 6.14 son las de desplazamiento circular a través del arrastre. Los diagramas de la columna simbólica de la tabla muestran la acción de desplazamiento provocada por estas instrucciones.

Considerar el ejemplo de la Figura 6.17 donde se utiliza la instrucción de desplazamiento

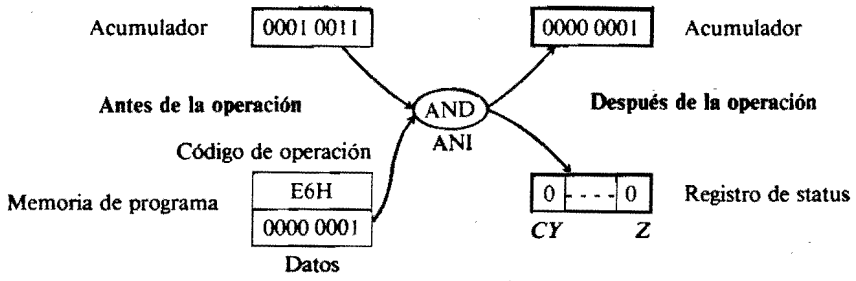
Instrucciones lógicas

Descripción de la operación	Modo de direcciona-miento	Nemotécnico	Cód. de oper.	Bytes	Formato de la instrucción	Simbólico	Señalizadores afectados
AND de *A* con dato	Inmediato	ANI	E6	2	Código de op. dato	$(A) \leftarrow (A) \cdot$ (byte 2)	Z CY se borra
AND de *A* con *L*	Registro	ANA *L*	A5	1	Código de op.	$(A) \leftarrow (A) \cdot (L)$	Z CY se borra
AND de *A* con *H*	Registro	ANA *H*	A4	1	Código de op.	$(A) \leftarrow (A) \cdot (H)$	Z CY se borra
AND de *A* con LOC (*H* & *L*)	Indirecto de registro	ANA *M*	A6	1	Código de op.	$(A) \leftarrow (A) \cdot ((H)(L))$	Z CY se borra
OR de *A* con dato	Inmediato	ORI	F6	2	Código de op. dato	$(A) \leftarrow (A) +$ (byte 2)	Z CY se borra
OR de *A* con *L*	Registro	ORA *L*	B5	1	Código de op.	$(A) \leftarrow (A) + (L)$	Z CY se borra
OR de *A* con *H*	Registro	ORA *H*	B4	1	Código de op.	$(A) \leftarrow (A) + (H)$	Z CY se borra
OR de *A* con LOC (*H* & *L*)	Indirecto de registro	ORA *M*	B6	1	Código de op.	$(A) \leftarrow (A) + ((H)(L))$	Z CY se borra
OR exclusivo de *A* con dato	Inmediato	XRI	EE	2	Código de op. dato	$(A) \leftarrow (A) \oplus$ (byte 2)	Z CY se borra
OR exclusivo de *A* con *A*	Registro	XRA *A*	AF	1	Código de op.	Borra el acumulador $(A) \leftarrow (A) \oplus (A)$	$Z = 1$ CY se borra
OR exclusivo de *A* con *L*	Registro	XRA *L*	AD	1	Código de op.	$(A) \leftarrow (A) \oplus (L)$	Z CY se borra
OR exclusivo de *A* con *H*	Registro	XRA *H*	AC	1	Código de op.	$(A) \leftarrow (A) \oplus (H)$	Z CY se borra
OR exclusivo de *A* con LOC (*H* & *L*)	Indirecto de registro	XRA *M*	AE	1	Código de op.	$(A) \leftarrow (A) \oplus ((H)(L))$	Z CY se borra
Complementar *A* (complemento a 1)	Inherente	CMA	2F	1	Código de op.	$(A) \leftarrow (\bar{A})$	
Desplazamiento circular a la derecha a través del arrastre	Inherente	RAR	1F	1	Código de op.		CY
Desplazamiento circular a la izquierda a través del arrastre	Inherente	RAL	17	1	Código de op.		CY

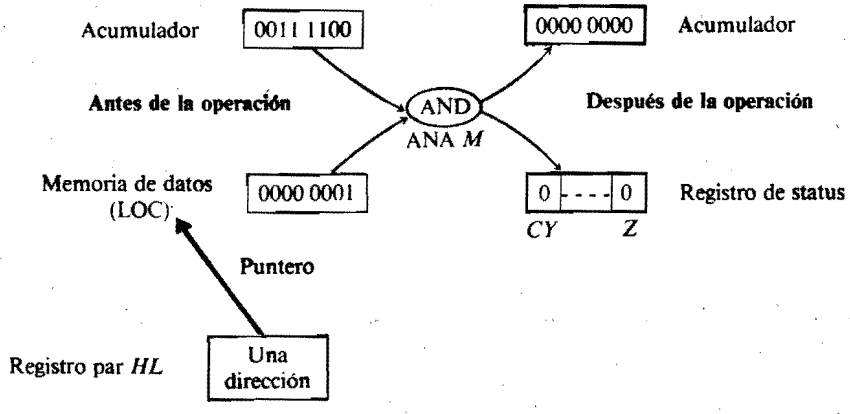
( ) contenido
(( )) = direccionamiento indirecto de registro

· = AND
+ = OR
⊕ = XOR

Figura 6.14. Instrucciones lógicas para el microprocesador genérico (subconjunto de las instrucciones del Intel 8080/8085).



(a) La instrucción AND inmediato



(b) La instrucción ANA *M*

Figura 6.15.

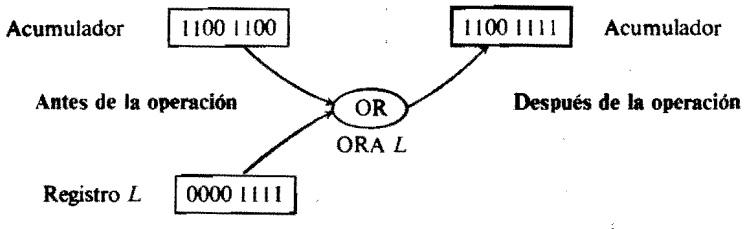


Figura 6.16. La instrucción OR de *A* con *L*.

circular a la derecha del acumulador a través del arrastre. El contenido del acumulador (00110001) se desplaza una posición a la derecha transfiriéndose el bit menos significativo (un 1 en este ejemplo) a la posición del bit del señalizador de arrastre. El arrastre actual entrará en la posición del bit MSB del acumulador. El acumulador contiene 00011000 después de la operación, mientras que el señalizador de arrastre se pone a 1 como muestra la Figura 6.17. El señalizador de cero no es afectado.

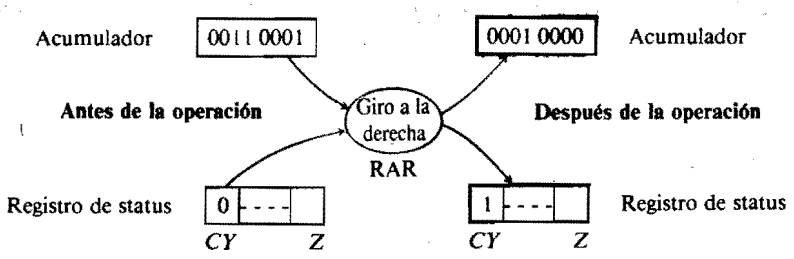


Figura 6.17. La instrucción desplazamiento circular a la derecha de *A* a través del arrastre.

Con una o más instrucciones de desplazamiento circular, puede comprobarse la posición de cualquier bit y el señalizador de arrastre puede ponerse a 1 ó a 0, pudiendo entonces ser examinado por una instrucción de bifurcación condicional. Otro uso de las instrucciones de desplazamiento circular es para comprobar la paridad. La *paridad par* sería un número binario con un número par de unos en la palabra.

Observar que las instrucciones de desplazamiento circular operan solamente en el acumulador y no necesitan ningún operando de ningún registro o posición de memoria. Por esta razón, el modo de direccionamiento se denomina *inherente,* o a veces no se cita ningún modo de direccionamiento para estas instrucciones. La mayoría de los microprocesadores tienen varios tipos de instrucciones de desplazamiento y desplazamiento circular además de las utilizadas por la MPU genérica.

En resumen, las instrucciones lógicas se utilizan para realizar manipulaciones del álgebra booleana en las variables. También se utilizan para poner a 1 ó 0 bits específicos de una palabra y para comprobar y comparar bits. Las aplicaciones de estas instrucciones habitualmente no son tan obvias como las aplicaciones de las instrucciones de suma y de resta.

PROBLEMAS RESUELTOS

6.22. Acudir a la Figura 6.14. El contenido del acumulador en la Figura 6.18 después de la operación AND es _____ (8 bits).

Solución:

La operación AND de 11110011 con 10000000, bit a bit, da 10000000. El contenido del acumulador en la Figura 6.18 después de la operación AND es 10000000.

6.23. Acudir a la Figura 6.14. Listar la condición de los señalizadores de la Figura 6.18 después de la operación AND.

Solución:

Según la información de la línea 3 de la Figura 6.14, el señalizador de arrastre se pondrá a 0. Después de la operación AND el contenido del acumulador no es cero; por tanto el señalizador de cero se pone en reinicialización (a 0).

6.24. Si el propósito del byte del registro *H* de la Figura 6.18 es comprobar si el MSB del acumulador es 0 ó 1, la palabra 10000000 en el registro *H* se denomina _____ (máscara, palabra de test).

Solución:

La palabra 10000000 en el registro *H* de la Figura 6.18 se denomina máscara si su propósito es comprobar el MSB del acumulador para determinar si es 0 ó 1. Al utilizar esta máscara en la instrucción ANA *H*, si el señalizador de cero es 1, entonces el MSB del byte del acumulador es 0; si el señalizador de cero es 0, entonces el MSB es 1.

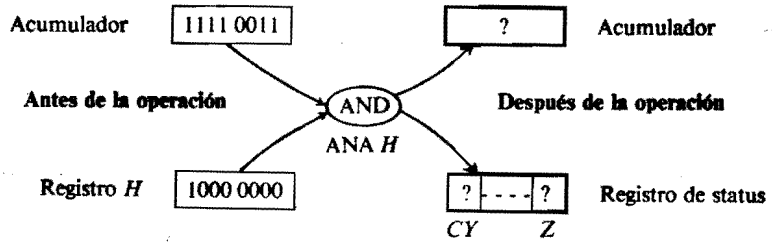


Figura 6.18.

6.25. Acudir a la Figura 6.14. Cuando se utiliza la instrucción «OR *A* inmediato» (nemotécnico de ORI), el operando que realizará la operación OR con el contenido del acumulador proviene de la memoria de _____ (datos, programa).

Solución:

La instrucción «OR *A* inmediato» busca su operando a partir del segundo byte de la memoria del programa.

6.26. El símbolo + utilizado en la tabla de instrucciones lógicas de la Figura 6.14 denota la operación _____ (suma, OR).

Solución:

El símbolo + utilizado en la tabla de instrucciones lógicas de la Figura 6.14 denota la operación OR.

6.27. Acudir a la Figura 6.14. La fuente de la dirección (LOC) del operando en la memoria de datos de la Figura 6.19 es el _____ (registro par *HL*, puntero de pila).

Solución:

Según la información sobre la instrucción ORA *M* de la Figura 6.14, la fuente de la dirección (LOC) del operando en la memoria de datos de la Figura 6.19 es el registro par *HL*.

6.28. Acudir a la Figura 6.14. El contenido del acumulador después de la operación OR en la Figura 6.19 es _____ (8 bits).

Solución:

El contenido del acumulador después de la operación OR, bit a bit, en la Figura 6.19 es 10000111.

6.29. Acudir a la Figura 6.14. Listar la condición de los señalizadores del registro de status de la Figura 6.19 después de la operación OR.

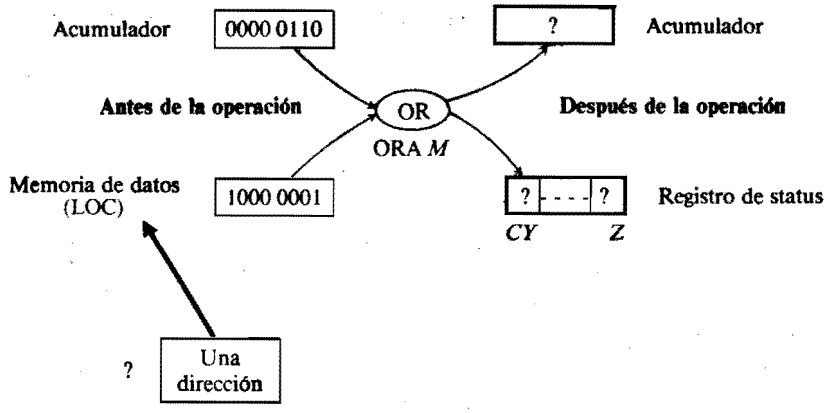


Figura 6.19.

Solución:

Según la información de la tabla de la Figura 6.14, el señalizador de arrastre se pone a 0. El contenido del acumulador no es cero; por tanto el señalizador de cero está a 0.

6.30. Acudir a la Figura 6.14. El contenido del acumulador de la Figura 6.20 después de la operación de desplazar circularmente a la izquierda es _____ (8 bits).

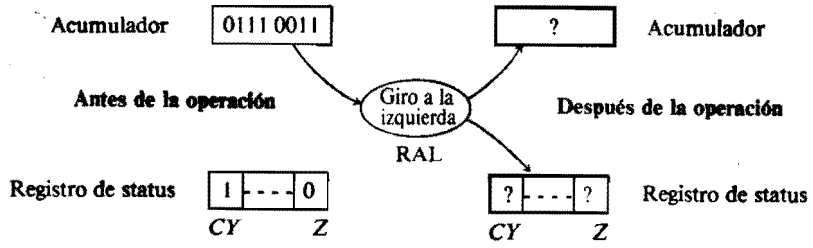


Figura 6.20.

Solución:

La acción de la instrucción de desplazamiento circular a la izquierda está contemplada en la columna simbólica de la Figura 6.14. El contenido del acumulador, en la Figura 6.20, después de la operación de desplazamiento circular a la izquierda es 11100111.

6.31. Acudir a la Figura 6.14. Listar la condición de los señalizadores del registro de status de la Figura 6.20 después de la operación de desplazamiento circular a la izquierda.

Solución:

De acuerdo con la Figura 6.14, solamente es afectado el señalizador de arrastre por la instrucción de desplazamiento circular a la izquierda; por tanto, el señalizador de cero permanece a 0. El señalizador de arrastre se pone a 0 porque el MSB del acumulador es desplazado a la izquierda a la posición del arrastre.

6.5. REPERTORIO DE INSTRUCCIONES: OPERACIONES DE TRANSFERENCIA DE DATOS

Las *instrucciones de transferencia de datos* son la tercera categoría de operaciones, del microprocesador genérico, que se van a considerar. Estas están resumidas en la Figura 6.21. Entre ellas se incluyen las instrucciones de transferencia de registro a registro, de carga de memoria, de almacenamiento en memoria, de entrada y salida y de poner a 1 el señalizador de arrastre. La mayoría de los programas contienen más instrucciones de transferencia de datos que otros tipos de operaciones. Casi ninguna instrucción de transferencia de datos afecta a los señalizadores de la MPU. En este grupo hay muchas instrucciones, ya que los datos pueden ser transferidos de cualquier posición de memoria a cualquier registro o desde cualquier registro a cualquier posición de memoria u otro registro. Los microprocesadores con más registros de CPU tienen más instrucciones de transferencia de datos. Tener en cuenta que cada instrucción de transferencia tendrá una fuente y un destino para el dato. El modo de direccionamiento determina cómo y dónde se encuentra el dato fuente.

Considerar la primera operación de transferencia listada en la tabla de la Figura 6.21. El dato se va a transferir del registro *L* registro *A*, o acumulador. El nemotécnico del lenguaje ensambla-

Instrucciones de transferencia de datos

Descripción de la operación	Modo de direcciona-miento	Nemotécnico	Cód. de oper.	Bytes	Formato de la instrucción	Simbólico	Señalizadores afectados
Transfiere *L* a *A*	Registro	MOV *A,L*	7D	1	Código de op.	$(A) \leftarrow (L)$	
Transfiere *H* a *A*	Registro	MOV *A,H*	7C	1	Código de op.	$(A) \leftarrow (H)$	
Transfiere *A* a *L*	Registro	MOV *L,A*	6F	1	Código de op.	$(L) \leftarrow (A)$	
Transfiere *A* a *H*	Registro	MOV *H,A*	67	1	Código de op.	$(H) \leftarrow (A)$	
Transfiere *HL* a *PC*	Registro	PCHL	E9	1	Código de op.	$(PC) \leftarrow (HL)$	
Transfiere *HL* a *SP*	Registro	SPHL	F9	1	Código de op.	$(SP) \leftarrow (HL)$	
Carga *A* con dato	Inmediato	MVI *A*	3E	2	Código de op. datos	$(A) \leftarrow$ (byte 2)	
Carga *L* con dato	Inmediato	MVI *L*	2E	2	Código de op. datos	$(L) \leftarrow$ (byte 2)	
Carga *H* con dato	Inmediato	MVI *H*	26	2	Código de op. datos	$(H) \leftarrow$ (byte 2)	
Carga LOC (*H* y *L*) a *A*	Indirecto de registro	MOV *A,M*	7E	1	Código de op.	$(A) \leftarrow ((H)(L))$	
Carga *HL* con dato	Inmediato	LXI *H*	21	3	Código de op. dato de orden inferior dato de orden superior	$(L) \leftarrow$ (byte 2) $(H) \leftarrow$ (byte 3)	

Figura 6.21. Instrucciones de transferencia de datos para el microprocesador genérico (subconjunto de instrucciones del Intel 8080/8085).

Instrucciones de transferencia de datos

Descripción de la operación	Modo de direcciona-miento	Nemotécnico	Cód. de oper.	Bytes	Formato de la instrucción	Simbólico	Señalizadores afectados
Carga *SP* con dato	Inmediato	LXI *SP*	31	3	Código de op. dato de orden inferior dato de orden superior	(*SP*) ← (byte 2 + 3)	
Carga *HL* de LOC *aa*	Directo	LHLD	2A	3	Código de op. direc. de orden inferior direc. de orden superior	(*L*) ← ((byte 2 + 3)) (*H*) ← ((byte 2 + 3)+ 1)	
Carga *A* de LOC *aa*	Directo	LDA	3A	3	Código de op. direc. de orden inferior direc. de orden superior	(*A*) ← ((byte 2 + 3))	
Almacena *A* en LOC *aa*	Directo	STA	32	3	Código de op. direc. de orden inferior direc. de orden superior	(Dirección) ← (*A*)	
Almacena *HL* en LOC *aa*	Directo	SHLD	22	3	Código de op. direc. de orden inferior direc. de orden superior	(Dirección) ← (*L*) (Dirección + 1) ← (*H*)	
Almacena *A* en LOC (*H* y *L*)	Indirecto de registro	MOV *M,A*	77	1	Código de op.	((*H*)(*L*)) ← (*A*)	
Almacena *H* en LOC (*H* y *L*)	Indirecto de registro	MOV *M,L*	75	1	Código de op.	((*H*)(*L*)) ← (*L*)	
Almacena *H* en LOC (*H* y *L*)	Indirecto de registro	MOV *M,H*	74	1	Código de op.	((*H*)(*L*)) ← (*H*)	
Entrada de *A* del puerto en LOC *a*	Directo	IN	DB	2	Código de op. direc. del puerto	(*A*) ← (Dirección del puerto)	
Salida de *A* al puerto en LOC *a*	Directo	OUT	D3	2	Código de op. direc. del puerto	(Dirección del puerto) ← (*A*)	
Coloca señalizador de arrastre	Inherente	STC	37	1	Código de op.	(*CY*) ← 1	*CY* se pone a 1

( ) = contenido de (( )) = direccionamiento indirecto de registro *PC* = contador de programa *SP* = puntero de pila

Figura 6.21. *(continuación).*

dor (MOV *A, L*) sugiere primero la acción, que es transferir (MOV); la siguiente letra del nemotécnico (*A* en este ejemplo) es el *destino* del dato, mientras que la última letra (*L* en este ejemplo) es la *fuente* del dato. Esto parece un poco al revés, pero es el convenio utilizado por Intel Corporation con las instrucciones del 8080/8085. Algunos microprocesadores identifican la fuente y destino de forma diferente y en el orden opuesto.

Considerar la operación de transferencia esbozada en la Figura 6.22. La fuente del dato es el registro *A* (acumulador), mientras que el destino es el registro *L*. El contenido del acumulador (11110000) es transferido al registro *L* sin ninguna modificación. El acumulador también contiene este dato después de que se completa la operación de transferencia.

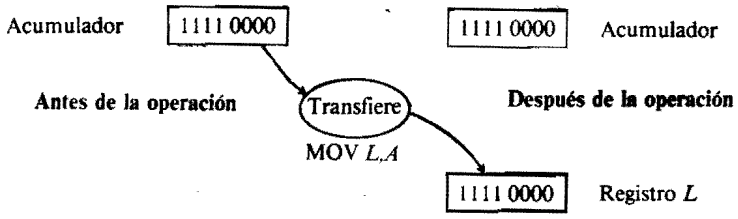


Figura 6.22. La instrucción de transferir *A* a *L*.

Una segunda operación de transferencia está en la Figura 6.23. Aquí la fuente del dato es el registro par *HL* de 16 bits, mientras que el destino es el puntero de pila de 16 bits. Si se observa cuidadosamente el nemotécnico de la instrucción «transferir *HL* a *SP*», tiene sentido cuando se recuerda que se cita primero el registro destino (*SP* en este ejemplo) y después el registro fuente (*HL* en este ejemplo). El nemotécnico para la instrucción «transferir *HL* a *SP*» es entonces SPHL.

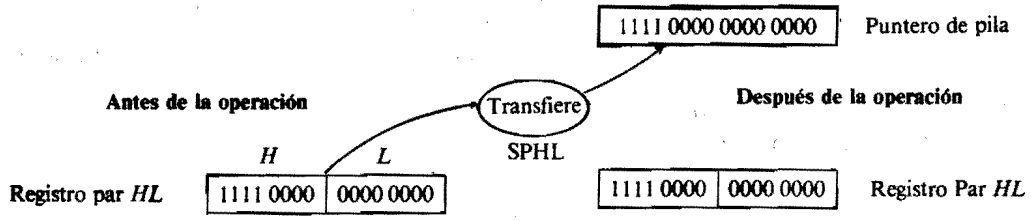


Figura 6.23. Instrucción de transferir *HL* a *SP*.

Hay cinco instrucciones de «carga inmediata» en la categoría de carga. Las instrucciones de «carga inmediata» son utilizadas muchas veces para colocar un valor inicial al registro microprocesador antes que el programa. Un ejemplo de instrucción de carga se detalla en la Figura 6.24. El registro par de 16 bits *HL* se va a cargar con un dato inmediato de la memoria del programa. Observar en la Figura 6.21 que la instrucción «carga *HL* con datos» requiere una instrucción de 3 bytes. El primer byte es el código de op (21H en este ejemplo), mientras que los 2 siguientes bytes de la memoria de programa son bytes de datos. El segundo byte contiene el dato de orden inferior (00000011 en este ejemplo) que se carga en el registro *L*, según se muestra en la Figura 6.24; el tercer byte contiene el dato de orden superior (00111100 en este ejemplo) que se carga en el registro *H* del registro para *HL*.

El siguiente ejemplo ilustra el uso de la instrucción «carga directa de *HL*» (nemotécnico LHLD). El problema ejemplo está esbozado en la Figura 6.25. La instrucción LHLD utiliza el *modo de direccionamiento directo*, de acuerdo con la tabla de la Figura 6.21. El segundo y tercer bytes de la memoria del programa son la dirección de 16 bits de la posición de la memoria de datos donde se encuentra el dato que se va a cargar. Seguir en la Figura 6.25 y comenzar con la instrucción de 3 bytes de la parte inferior izquierda de la memoria de programa. El código de op de la instrucción LHLD es 2AH. La CPU entonces ensambla los 2 bytes siguientes del programa

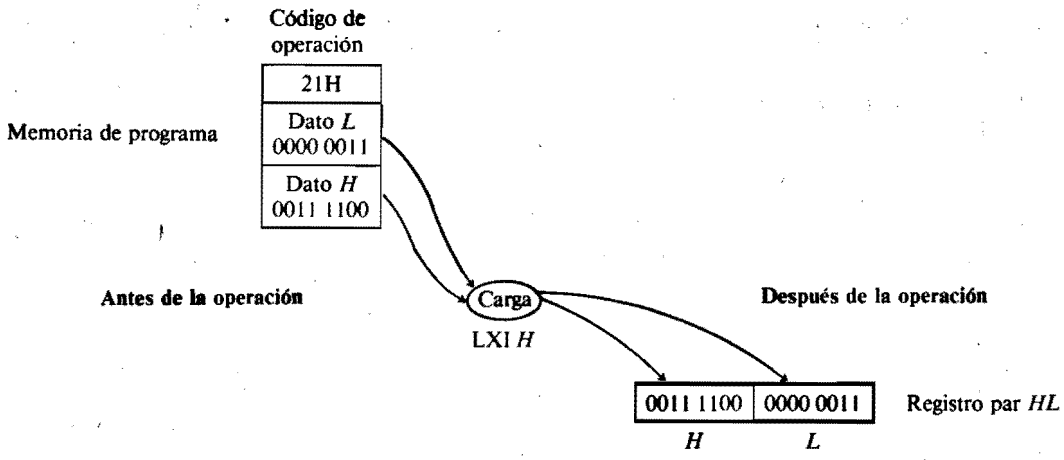


Figura 6.24. La instrucción de carga inmediata *HL*.

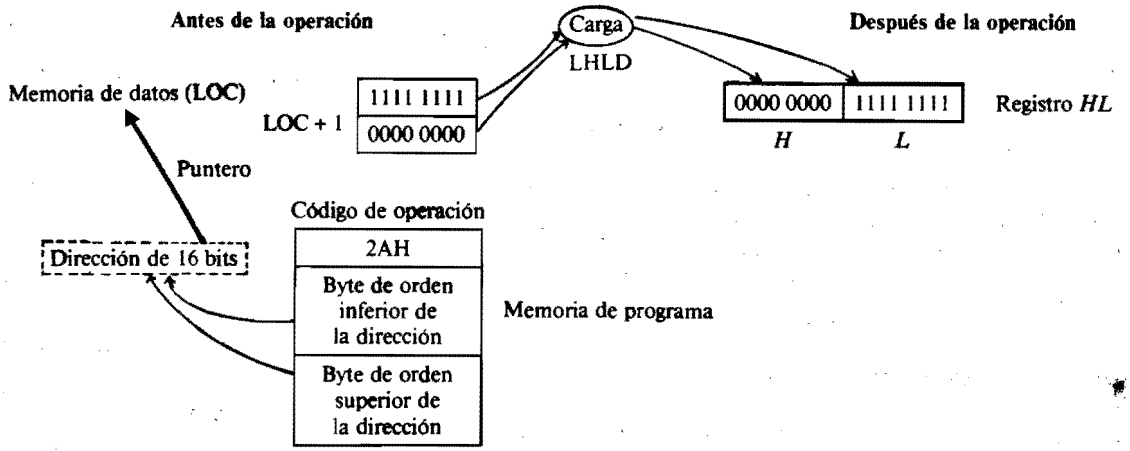


Figura 6.25. La instrucción de carga directa *HL*.

en una dirección de 16 bits (lo indica el recuadro a trazos), que sirve como puntero de dirección para encontrar la posición (LOC) en la memoria de datos. Este byte de memoria de datos se carga entonces en el registro *L*. El contenido de la siguiente posición de la memoria de datos (etiquetado LOC + 1 en este ejemplo) se carga entonces en el registro *H*.

El microprocesador genérico tiene cinco instrucciones de almacenamiento, que se resumen en la Figura 6.21. Las operaciones de almacenamiento son muy utilizadas para depositar resultados y otros datos en memoria. Considerar el problema ejemplo de la Figura 6.26 que utiliza la instrucción «almacena *A* directo». El contenido del acumulador (registro *A*) se almacena en la posi-

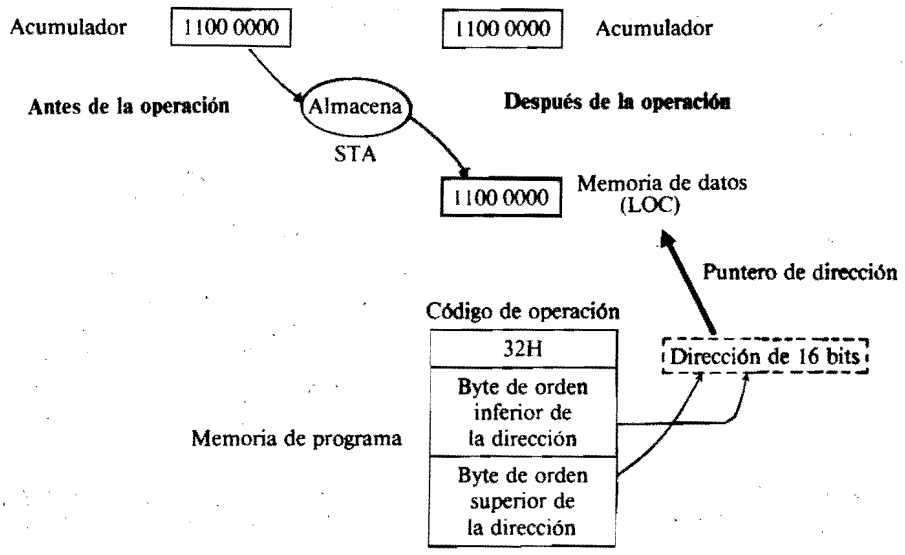


Figura 6.26. La instrucción de almacenar *A* directo.

ción de memoria de datos (LOC), señalada por una dirección de 16 bits, y formada por el segundo y tercer bytes de la instrucción. Después de la operación, la posición de memoria de datos (LOC) y el acumulador contienen el mismo dato (11000000 en este ejemplo).

La instrucción de entrada (input) listada en la tercera línea del extremo inferior de la Figura 6.21 es similar a la instrucción de cargar. La fuente de la transferencia de datos es un puerto de entrada identificado por un número binario de 8 bits ($0–255_{10}$), el destino es el acumulador de la CPU. Un ejemplo se muestra en la Figura 6.27 donde se utiliza la instrucción de entrada. El dato del puerto de entrada señalado por el segundo byte de la instrucción es transferido al acumulador del microprocesador. En este ejemplo, 00001111 es introducido y transferido al acumulador desde un puerto identificado por LOC.

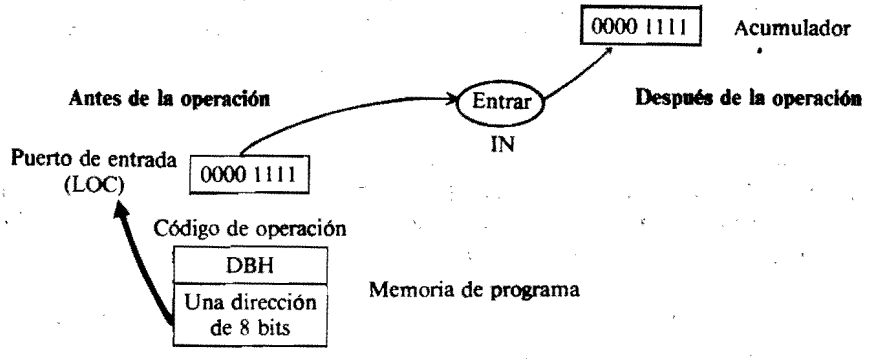


Figura 6.27. La instrucción de entrada directa.

PROBLEMAS RESUELTOS

6.32. Acudir a la Figura 6.21. Durante el uso de la instrucción MOV *H,A,* el registro _____ (*A, H*) es la fuente del dato mientras que el registro _____ (*A, H*) es el destino.

Solución:

Según la información de la Figura 6.21, el registro *A* es la fuente mientras que el registro *H* es el destino del dato durante la ejecución de la instrucción MOV *H,A.*

6.33. Acudir a la Figura 6.21. En la instrucción PCHL, el registro _____ (*HL, PC*) es la fuente del dato mientras que el _____ _____ de 16 bits es el destino.

Solución:

Según la información de la Figura 6.21, el registro *HL* es la fuente mientras que el contador de programa de 16 bits es el destino del dato durante la ejecución de la instrucción PCHL.

6.34. Acudir a la Figura 6.21. Los señalizadores _____ (son afectados, no son afectados) por la mayoría de las operaciones de transferencia de datos en el repertorio de instrucciones de la microcomputadora genérica.

Solución:

Según la información de la columna derecha de la Figura 6.21, los señalizadores no son afectados por la mayoría de las operaciones de transferencia de datos.

6.35. Acudir a la Figura 6.21. El nemotécnico del lenguaje ensamblador para la instrucción de carga utilizada en la Figura 6.28 es _____.

Solución:

Según la información de la tabla de la Figura 6.21, el nemotécnico del lenguaje ensamblador para la instrucción de carga utilizada en la Figura 6.28 es MVI *A.*

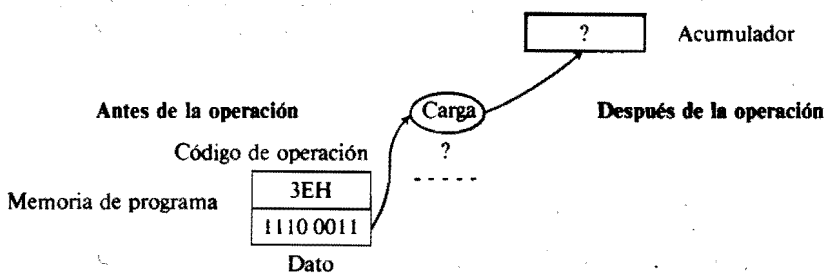


Figura 6.28.

6.36. El contenido del acumulador de la Figura 6.28 después de la operación de carga es _____ (8 bits).

Solución:

El contenido del acumulador de la Figura 6.28 después de la operación de carga es 11100011.

6.37. Acudir a la Figura 6.21. La operación de **carga** mostrada en la Figura 6.28 utiliza direccionamiento _____ (directo, inmediato).

Solución:

Según la información de la tabla de la Figura 6.21, la operación de carga mostrada en la Figura 6.28 utiliza direccionamiento inmediato.

6.38. Acudir a la Figura 6.21. La instrucción MOV *A,M* tiene el código de op 7EH, utiliza direccionamiento _____ (directo, indirecto de registro) y la fuente de datos es una posición de la memoria de _____ (datos, programa) señalada por el registro par *HL*.

Solución:

Según la información de la Figura 6.21, la instrucción MOV *A,M* tiene el código de op 7EH, utiliza direccionamiento indirecto de registro y la fuente de datos es una posición de memoria señalada por el registro par *HL*.

6.39. Acudir a la Figura 6.21. La instrucción SHLD tiene el código de op 22H y utiliza direccionamiento _____ (directo, inmediato).

Solución:

Según la información de la Figura 6.21, la instrucción SHLD tiene el código de op 22H y utiliza direccionamiento directo.

6.40. Acudir a la Figura 6.21. La instrucción «almacena directo *HL*» (código de op = 22H) es una instrucción de _____ (número) bytes. Los 2 últimos bytes de la instrucción del programa contienen información sobre _____ (datos, direcciones).

Solución:

Según la información de la Figura 6.21, la instrucción «almacena directo *HL*» (código de op = 22H) es una instrucción de 3 bytes. Los 2 últimos bytes de la instrucción del programa contienen información sobre direcciones.

6.41. Acudir a la Figura 6.21. La fuente del dato cuando se utiliza la instrucción SACAR (OUT) en la Figura 6.29 es el registro _____ (*A*, *H*, *L*).

Solución:

De acuerdo con la Figura 6.21, la fuente del dato cuando se utiliza la instrucción OUT en la Figura 6.29 es el registro *A*, que también se denomina acumulador.

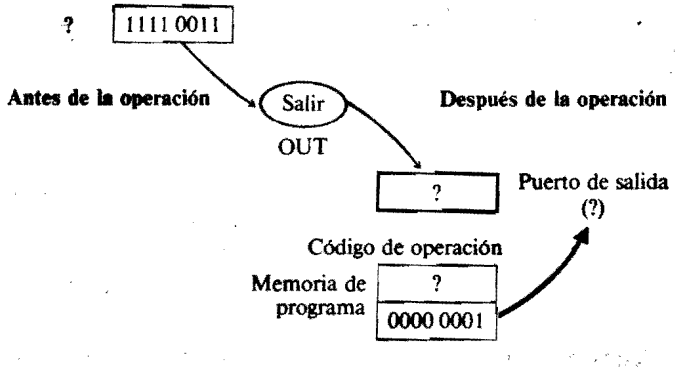


Figura 6.29.

6.42. Después de la operación sacar en la Figura 6.29, el puerto de salida número _____ (hex) contendrá el dato _____ (8 bits).

Solución:

Después de la operación sacar de la Figura 6.29, el puerto de salida número 01H contendrá el dato 11110011. La dirección del puerto de salida proviene del segundo byte de la memoria del programa.

6.6. REPERTORIO DE INSTRUCCIONES: OPERACIONES DE BIFURCACION

Las *instrucciones de bifurcación* son la cuarta categoría de las operaciones del microprocesador genérico que se van a considerar y están resumidas en la Figura 6.30. Observar que las operaciones se referencian como *instrucciones de salto,* aunque los términos *salto* y *bifurcación* se considerarán sinónimos en este capítulo. Algunos fabricantes hacen distinciones entre instrucciones de bifurcación y de salto. Las operaciones de bifurcación o de salto a veces se denominan *instrucciones de transferencia de control.* Como las operaciones de la Figura 6.30 son un subconjunto del

Instrucciones de bifurcación

Descripción de la operación	Modo de direcciona-miento	Nemotécnico	Cód. de oper.	Bytes	Formato de la instrucción	Simbólico	Señalizadores afectados
Salta a LOC *aa*	Inmediato	JMP	C3	3	Código de op. Direc. de orden inferior Direc. de orden superior	(*PC*) ← (Dirección)	
Salta a LOC *aa* si es 0	Inmediato	JZ	CA	3	Código de op. Direc. de orden inferior Direc. de orden superior	Si el señalizador de 0 = 1, entonces (*PC*) ← (dirección)	
Salta a LOC *aa* si no es 0	Inmediato	JNZ	C2	3	Código de op. Direc. de orden inferior Direc. de orden superior	Si el señalizador de 0 = 0, entonces (*PC*) ← (dirección)	
Salta a LOC *aa* si se coloca el señalizador de arrastre	Inmediato	JC	DA	3	Código de op. Direc. de orden inferior Direc. de orden superior	Si el señalizador de arrastre = 1, entonces (*PC*) ← (dirección)	
Salta a LOC *aa* si no se coloca el señalizador de arrastre	Inmediato	JNC	D2	3	Código de op. Direc. de orden inferior Direc. de orden superior	Si el señalizador de arrastre = 0, entonces (*PC*) ← (dirección)	

( ) = contenido de
PC = contador de programa

Figura 6.30. Instrucciones de bifurcación para el microprocesador genérico (subconjunto de instrucciones del Intel 8080/8085).

repertorio de instrucciones del 8080/8085, el convenio de Intel de llamar a estas instrucciones de salto se utilizará más tarde en este capítulo.

La microcomputadora normalmente ejecuta instrucciones en orden secuencial. El contador de programa de 16 bits del microprocesador genérico siempre sigue la pista de la dirección de la siguiente instrucción que se va a buscar en memoria y ejecutar. El contador de programa normalmente sigue una secuencia ascendente aumentando de uno en uno. Las instrucciones de salto o bifurcación proporcionan un método para cambiar el valor del contador de programa, alterando así la secuencia normal de ejecución del programa.

Las instrucciones de salto o bifurcación se subdividen normalmente en *operaciones de salto incondicional* y *de salto condicional.* La primera instrucción listada en la Figura 6.30 es una instrucción de salto incondicional. La instrucción «saltar inmediato» es una instrucción de 3 bytes, utilizada para cargar una dirección específica en el contador del programa de la CPU. Un problema ejemplo que utiliza esta instrucción, de salto incondicional, se muestra en la Figura 6.31 donde la dirección 2000H se está cargando en el contador del programa. La información de la dirección sigue inmediatamente al código de op, y por tanto, se denomina direccionamiento inmediato. Observar en la Figura 6.31 que la parte de orden inferior de la dirección de 16 bits se localiza en el segundo byte de la memoria de programa, mientras que el tercer byte contiene la parte de orden superior de la dirección. Una instrucción de este tipo se utiliza en un principio para inicializar el contador de programa cuando empiece la ejecución de un nuevo programa. La instrucción de salto o bifurcación incondicional puede considerarse como un método de cargar una nueva dirección en el contador de programa.

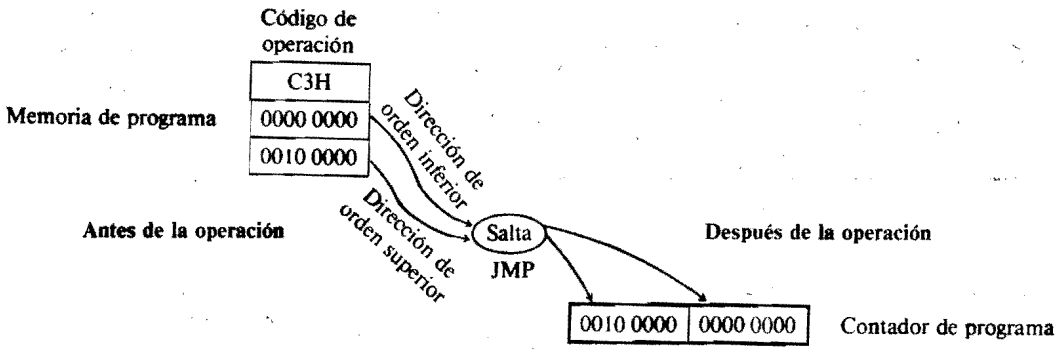


Figura 6.31. Instrucción de bifurcación incondicional.

Las cuatro últimas operaciones de bifurcación, detalladas en la Figura 6.30, son instrucciones de salto condicional. Estas instrucciones harán que la dirección inmediata se coloque en el contador de programa *sólo si se satisface la condición especificada.* En cualquier otro caso el contador de programa, simplemente, contará en secuencia ascendente. Un problema ejemplo que utiliza la instrucción «salta si cero» aparece en la Figura 6.32. En este ejemplo, el contador de programa contiene 2013H antes de la operación y continuará en secuencia ascendente hasta que el señalizador de cero se ponga a 1. La CPU evalúa la condición del señalizador de cero en el registro de status y encuentra que está a 1. Esto significa que el resultado de la última operación lógica o aritmética fue cero; las condiciones son correctas para un salto, y la MPU carga el contador de

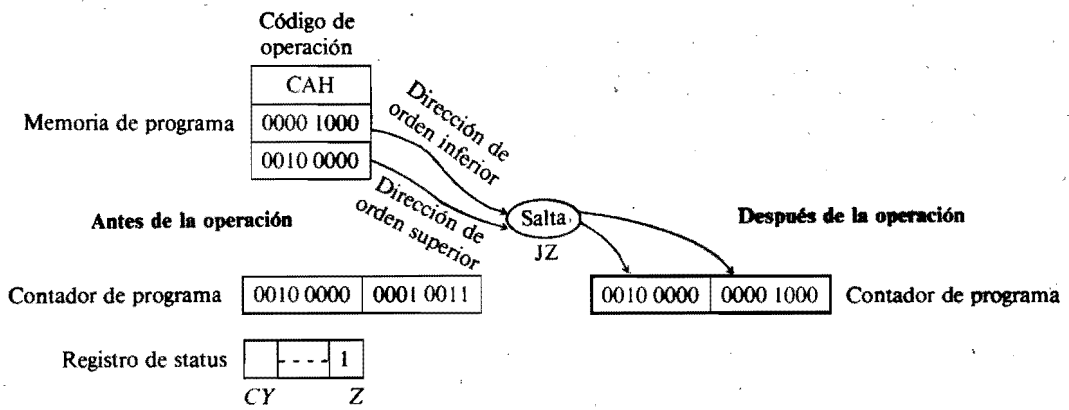


Figura 6.32. La Instrucción de salta si cero.

programa con la nueva dirección 2008H. La nueva dirección proviene de la memoria de programa, por lo que se dice que esta instrucción utiliza direccionamiento inmediato. La siguiente instrucción que se va a ejecutar será la instrucción de la posición de memoria 2008H y no la de la 2013H como podría esperarse. El microprocesador genérico salta a una nueva posición de la memoria del programa. Los saltos o bifurcaciones pueden hacerse hacia atrás o hacia adelante en la memoria del programa. El ejemplo de la Figura 6.32 ilustra un salto hacia atrás, que probablemente sea el más común.

Las instrucciones de salto o bifurcación se encuentran en casi todos los programas de microprocesadores. Son valiosas debido a su característica de toma de decisiones. Las instrucciones de salto o bifurcación se utilizan para formar ciclos y saltos en el programa. Los ciclos y saltos son dos técnicas de programación que se estudiarán más tarde en este capítulo.

PROBLEMAS RESUELTOS

6.43. Las instrucciones de bifurcación también se denominan de _____ (brinco, salto) o _____ de control.

Solución:

Las instrucciones de bifurcación también se denominan instrucciones de salto o de transferencia de control.

6.44. Se dice que las operaciones _____ (lógicas, de salto condicional) son instrucciones de toma de decisión.

Solución:

Las operaciones de salto condicional son instrucciones de toma de decisión.

6.45. Acudir a la Figura 6.30. El nemotécnico del lenguaje ensamblador para la operación de salto, esbozada en la Figura 6.33, es _____ .

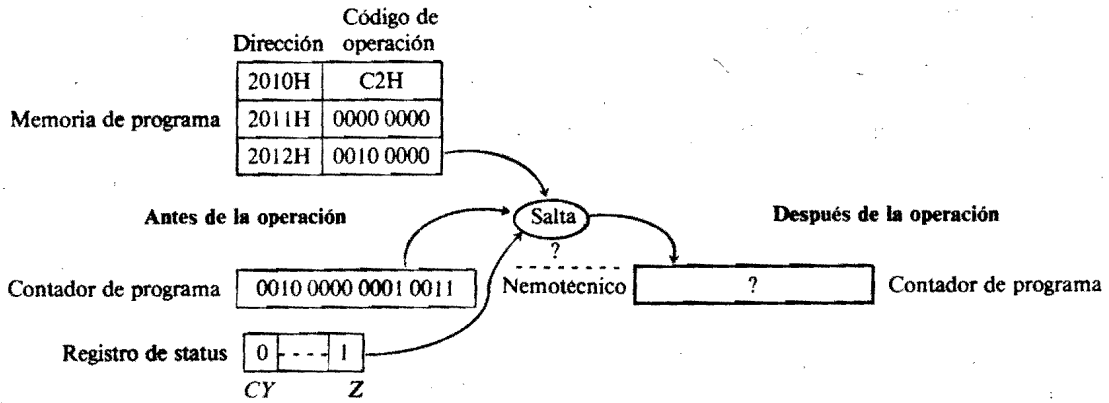


Figura 6.33.

Solución:

El código de operación para la operación «salta si no cero» de la Figura 6.33 es C2H. A partir de la tabla de la Figura 6.30 se determina que el nemotécnico del lenguaje ensamblador para esta operación de salto es JNZ.

6.46. Acudir a la Figura 6.30. El contenido del contador de programa de la Figura 6.33 después de la operación «salta si no cero» es _____ (hex).

Solución:

El señalizador de cero del registro de status de la Figura 6.33 se pone a 1, lo cual significa que el resultado de la última operación aritmética o lógica fue cero. La condición para el salto *no se satisface*. La dirección del segundo y tercer bytes de la memoria de programa no son cargadas en el contador de programa, por lo que su contenido permanece a 2013H, como estaba antes de la instrucción de salto condicional. La siguiente instrucción que se va a ejecutar se localiza en 2013H.

6.47. Acudir a la Figura 6.30. El nemotécnico de la instrucción «salta si arrastre» de la Figura 6.34 es JC, mientras que el código de operación es _____ (hex).

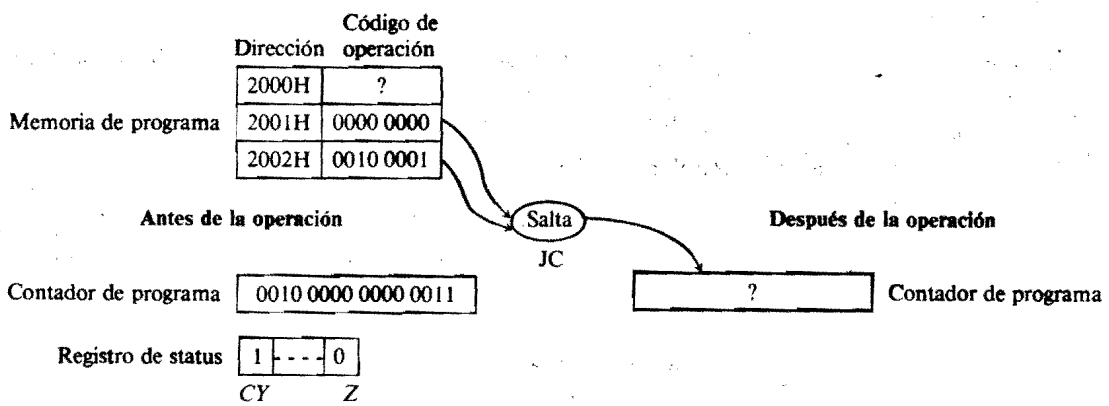


Figura 6.34.

Solución:

El nemotécnico de la instrucción de «salta si arrastre» de la Figura 6.34 es JC, mientras que la tabla de la Figura 6.30 indica que el código de op de esta instrucción es DAH.

6.48. El contenido del contador de programa después de la operación de salto de la Figura 6.30 es _____ (hex).

Solución:

El señalizador de arrastre se pone a 1, y por tanto se satisface la condición para la operación de salto. La MPU carga la nueva dirección 2100H en el contador de programa. La siguiente instrucción que se va a ejecutar se carga en 2100H en la memoria del programa.

6.7. REPERTORIO DE INSTRUCCIONES: OPERACIONES DE LLAMADA Y VUELTA DE SUBRUTINA

Las *instrucciones de llamada y vuelta de subrutina* son la quinta categoría de operaciones del microprocesador genérico que se van a considerar. Estas están resumidas en la Figura 6.35. Observar que el microprocesador genérico tiene solamente dos instrucciones en este grupo. Las instrucciones de llamada y vuelta son utilizadas siempre en parejas.

La instrucción de llamada de 3 bytes se utiliza en el programa principal para hacer que el microprocesador salte a una subrutina. En el ejemplo mostrado en la Figura 6.36, la subrutina es un pequeño grupo de instrucciones que consiguen un retardo de 1 segundo. Cuando la MPU encuentra la primera instrucción de llamada en la posición 200DH, encuentra la dirección de salto en los siguientes bytes de programa. La dirección de la instrucción que sigue a la de llamada se guarda en la pila (no mostrada en la Figura 6.36). La MPU entonces bifurca al comienzo de la subrutina en la posición 1000H. Las instrucciones de la subrutina que consigue un retardo de 1 segundo se ejecutan hasta que la CPU encuentra la instrucción de vuelta. La dirección guardada

Instrucciones de llamada y vuelta de subrutina

Descripción de la operación	Modo de direcciona-miento	Nemotécnico	Cód. de oper.	Bytes	Formato de la instrucción	Simbólico	Señalizadores afectados
Llama a la subrutina en LOC *aa*	Inmediato/ indirecto de registro	CALL	CD	3	Código de op. Direc. de orden inferior Direc. de orden superior	$((SP)-1) \leftarrow (PCH)$ $((SP)-2) \leftarrow (PCL)$ $(SP) \leftarrow (SP)-2$ $(PC) \leftarrow$ (dirección)	
Vuelta de subrutina	Indirecto de registro	RET	C9	1	Código de op.	$(PCL) \leftarrow ((SP))$ $(PCH) \leftarrow ((SP)+1)$ $(SP) \leftarrow (SP)+2$	

( ) = contenido de
(( )) = direccionamiento indirecto de registro
PC = contador de programa
SP = puntero de pila

Figura 6.35. Instrucciones de llamada y vuelta de subrutina para el microprocesador genérico (subconjunto de instrucciones del Intel 8080/8085).

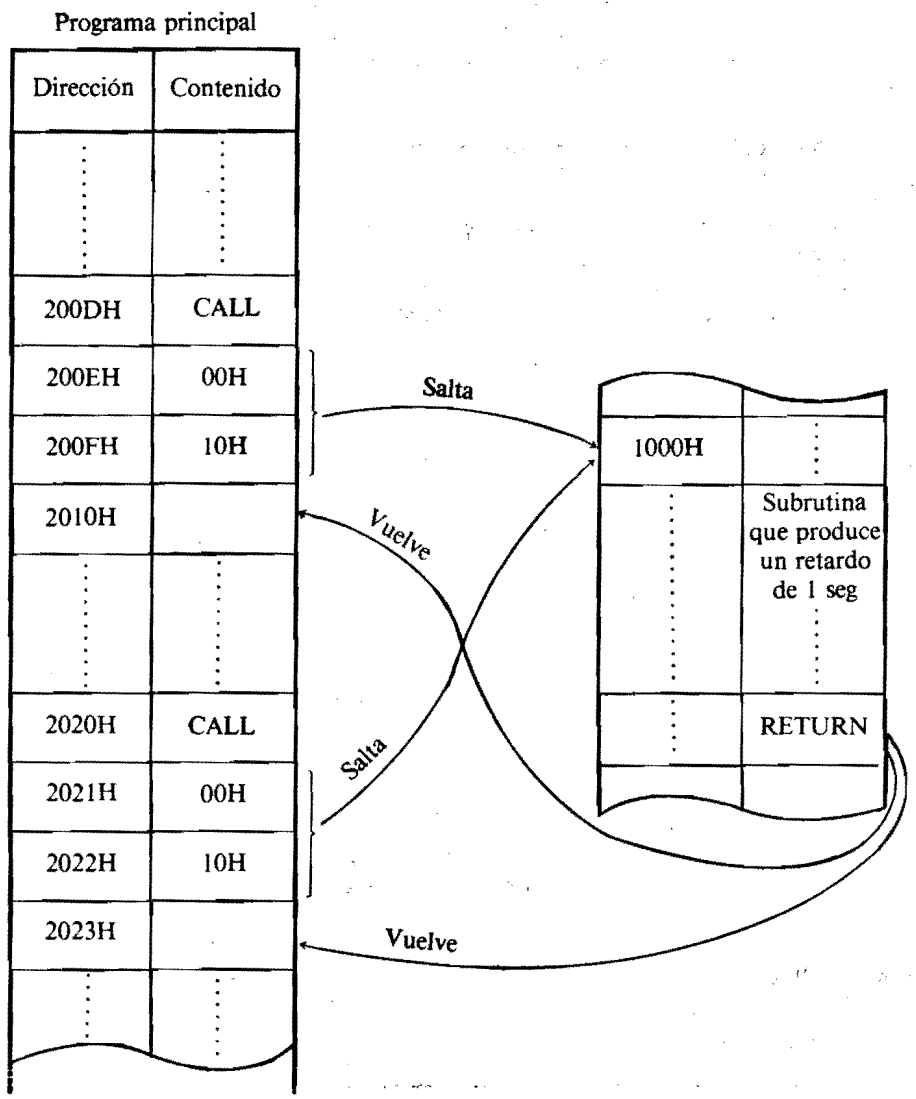


Figura 6.36. Flujo de programa del programa principal a la subrutina y vuelta utilizando las instrucciones de retorno y llamada.

en la pila (2010H) es devuelta al contador de programa, y la MPU continúa el programa principal donde lo dejó cuando se llamó por primera vez a la subrutina. La MPU ejecuta las instrucciones en secuencia hasta que encuentra la segunda instrucción de llamada en la dirección 2020H. La MPU guarda en la pila la dirección de la siguiente instrucción (2023H) y salta a la subrutina en la dirección 1000H. Hasta que se completa de nuevo la subrutina de retardo de tiempo, la instrucción de vuelta saca de la pila la dirección de la siguiente instrucción del programa principal (2023H) y la transfiere al contador de programa. Se puede utilizar muchas veces una subrutina en la ejecución de un programa principal. La subrutina puede almacenarse en una RAM o ROM.

La *instrucción de llamada combina las funciones de las operaciones de introducir y saltar.* La Figura 6.37 detalla el uso de la instrucción de llamada. Primero introduce el contenido actual del contador de programa en la pila. A continuación, el contador de programa se carga con una nueva dirección, para realizar el salto a la de la subrutina. La secuencia de acciones puede ser seguida a través de los números inscritos en círculos en la Figura 6.37.

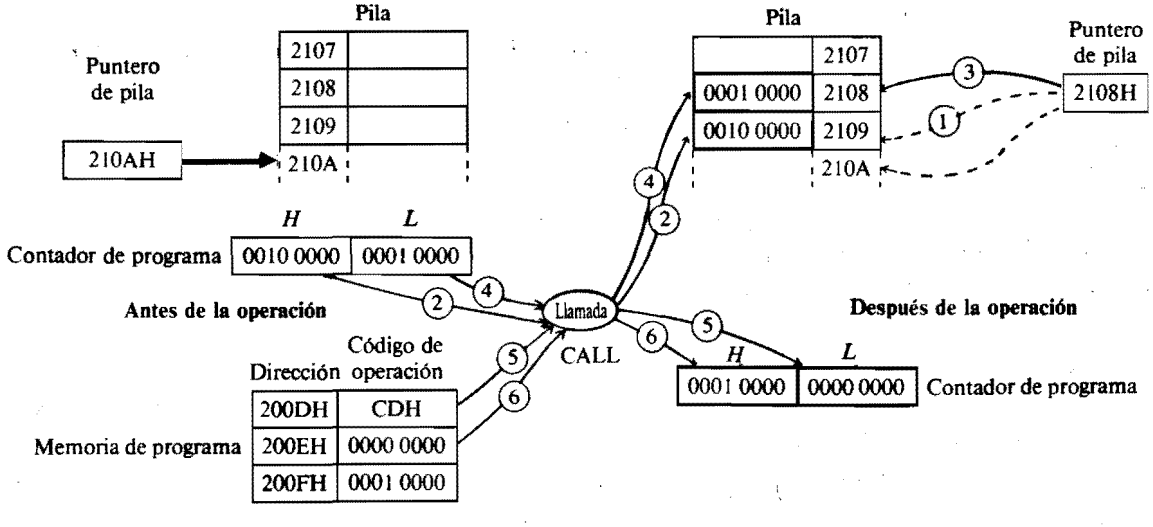


Figura 6.37. La instrucción de llamada a subrutina.

1. El puntero de pila se decrementa de 210AH a 2109H.
2. El byte de orden superior del contador de programa es introducido en la pila, en la posición 2109H.
3. El puntero de pila se decrementa de 2109H a 2108H.
4. El byte de orden inferior del contador de programa es introducido en la pila en la posición 2108H.
5. La dirección de orden inferior es transferida desde el segundo byte de la memoria de programa al byte inferior del contador de programa.
6. La dirección de orden superior es transferida desde el tercer byte de la memoria de programa al byte superior del contador de programa.

La MPU irá ahora a la nueva dirección señalada por el contador de programa (1000H en este ejemplo). Esta es la dirección de la primera instrucción de la subrutina.

Considerar el ejemplo mostrado en la Figura 6.38 utilizando la instrucción de vuelta de subrutina. La instrucción de retorno (o de vuelta) debe transferir el contenido de la pila al contador de programa. La secuencia de acciones puede ser seguida a través de los números inscritos en círculos de la Figura 6.38.

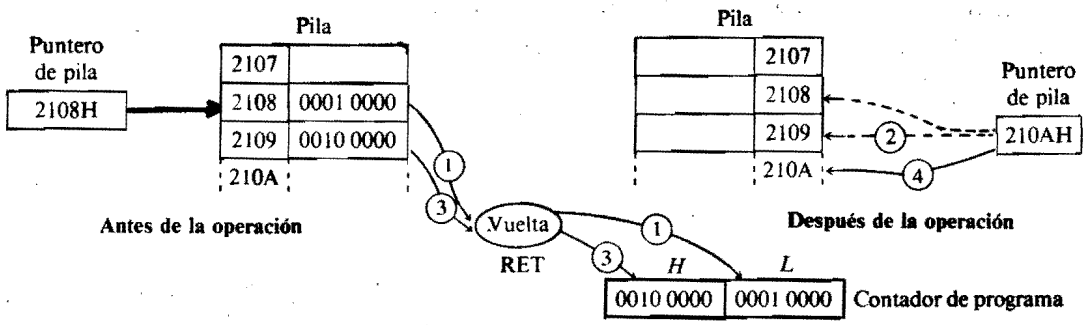


Figura 6.38. La instrucción de vuelta de subrutina.

1. El tope de la pila (posición 2108H) es sacado, y su contenido transferido al byte inferior del contador de programa.
2. El puntero de pila es incrementado de 2108H a 2109.
3. La parte superior de la pila (ahora posición 2109H) es sacada, y su contenido transferido al byte superior del contador de programa.
4. El puntero de pila es incrementado de 2109H a 210AH.

El contador de programa ahora contiene la dirección de 16 bits (2010H) de la siguiente instrucción que se va a buscar en la memoria del programa.

PROBLEMAS RESUELTOS

6.49. La instrucción de _____ (llamada, vuelta) es parte del programa principal, mientras que la instrucción de _____ (llamada, vuelta) es el fin de una subrutina.

Solución:

Ver la Figura 6.36. La instrucción de llamada es parte del programa principal, mientras que la instrucción de vuelta está al final de una subrutina.

6.50. La llamada es una instrucción de _____ (número) bytes. Los 2 últimos bytes contienen la dirección de la primera instrucción de _____ (memoria principal, una subrutina).

Solución:

La llamada es una instrucción de 3 bytes, según la tabla de la Figura 6.35. Los 2 últimos bytes contienen la dirección de la primera instrucción de la subrutina.

6.51. La operación de llamada realiza las funciones de _____ (sacar, introducir) y de _____ (sumar, saltar).

Solución:

La operación de llamada realiza las funciones de las instrucciones de introducir y saltar.

6.52. Acudir a la Figura 6.39. El contenido de la posición de la pila 2109H después de la operación de llamada será de _____ (8 bits).

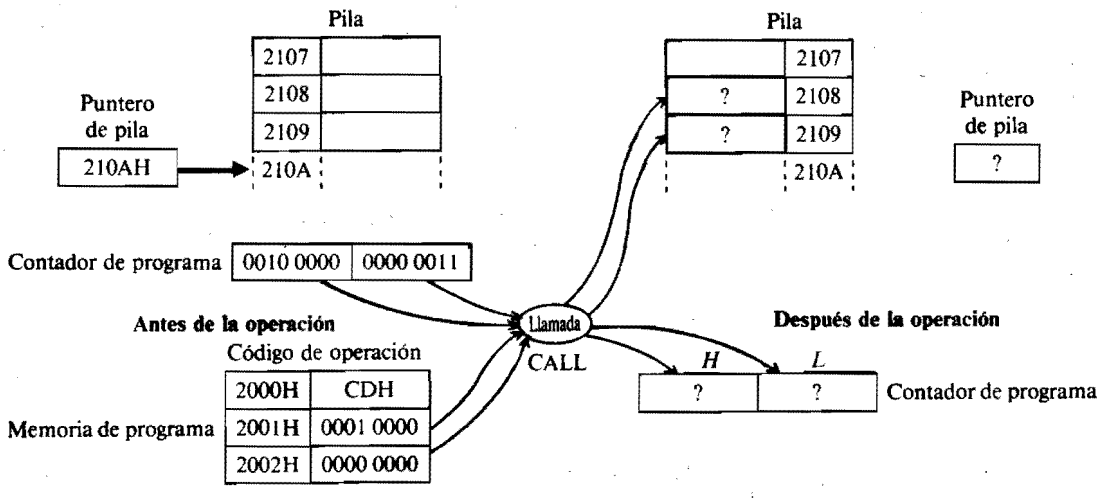


Figura 6.39.

Solución:

El contenido de la posición de pila 2109H de la Figura 6.39 después de la operación de llamada será 00100000. El byte de orden superior del contador de programa será almacenado primero en la pila, de acuerdo con la tabla de la Figura 6.35 y el diagrama de la Figura 6.37.

6.53. Acudir a la Figura 6.39. El contenido de la posición de la pila 2108H después de la operación de llamada será _____ (8 bits).

Solución:

El contenido de la posición de la pila 2108H en la Figura 6.39 después de la operación de llamada será 00000011. El byte de orden inferior del PC será almacenado en la segunda pila.

6.54. Acudir a la Figura 6.39. El puntero de pila contendrá _____ (hex) después de la operación de llamada.

Solución:

Después de la operación de llamada en la Figura 6.39, el puntero de pila señalará a la parte superior de la pila, que estará en la posición 2108H.

6.55. Acudir a la Figura 6.39. El contenido del contador de programa después de la operación de llamada será _____ (hex).

Solución:

El contenido del contador de programa después de la operación de llamada de la Figura 6.39 será 0010H. Esta es la dirección de la primera instrucción de la subrutina.

6.56. Las instrucciones de llamada y vuelta de la subrutina se utilizan casi siempre _____ (en parejas, individualmente).

Solución:

Las instrucciones de llamada y vuelta de subrutina casi siempre se utilizan en parejas.

6.8. REPERTORIO DE INSTRUCCIONES: OPERACIONES MISCELANEAS

Las *instrucciones misceláneas* son la última categoría de las operaciones genéricas del microprocesador que se van a considerar. Estas están resumidas en la Figura 6.40. Incluyen las instrucciones de introducir, sacar, no operación y alto.

Las instrucciones de introducir y sacar (push, pop) se utilizaron en la Sección 5.6 cuando se explicaron el puntero de pila y la pila; estas instrucciones colocan y toman datos de la pila de la microcomputadora y se utilizan por parejas porque lo que se hace en la pila con una instrucción de introducir tiene que deshacerse con una instrucción de sacar. Las instrucciones de introducir y sacar son muy utilizadas en las subrutinas. Una instrucción de «introducir *A* y señalizadores en pila» puede utilizarse como la primera instrucción de una subrutina como, por ejemplo, la esbozada en la Figura 6.36. De esta forma se guarda el contenido del acumulador y de los señalizadores para que no puedan ser alterados por la subrutina; justo antes de la operación de vuelta en la Figura 6.36, una instrucción de «sacar *A* y los señalizadores de la pila» deberá restaurar el contenido original del acumulador y los señalizadores.

Considerar la primera instrucción listada en la tabla de la Figura 6.40. Es la instrucción de «introducir *A* y señalizadores en la pila» y tiene el nemotécnico PUSH *PSW*. La parte *PSW* del nemotécnico significa palabra de status del programa, que en este caso es el contenido del acumu-

Instrucciones misceláneas

Descripción de la operación	Modo de direcciona-miento	Nemotécnico	Cód. de oper.	Bytes	Formato de la instrucción	Simbólico	Señalizadores afectados
Introduce *A* y señalizadores a la pila	Indirecto de registro	PUSH *PSW*	F5	1	Código de op.	((*SP*) − 1) ← (*A*) ((*SP*) − 2) ← (señali.) (*SP*) ← (*SP*) − 2	
Introduce *HL* a la pila	Indirecto de registro	PUSH *H*	E5	1	Código de op.	((*SP*) − 1) ← (*H*) ((*SP*) − 2) ← (*L*) (*SP*) ← (*SP*) − 2	
Saca *A* y señalizadores de la pila	Indirecto de registro	POP *PSW*	F1	1	Código de op.	(señali.) ← ((*SP*)) (*A*) ← ((*SP*) + 1) (*SP*) ← (*SP*) + 2	
Saca *HL* de la pila	Indirecto de registro	POP *H*	E1	1	Código de op.	(*L*) ← ((*SP*)) (*H*) ← ((*SP*) + 1) (*SP*) ← (*SP*) + 2	
No operación	Inherente	NOP	00	1	Código de op.	(*PC*) ← (*PC*) + 1	
Alto	Inherente	HLT	76	1	Código de op.		

( ) = contenido de
(( )) = direccionamiento indirecto de registro
A = acumulador
SP = puntero de pila

Figura 6.40. Instrucciones misceláneas para el microprocesador genérico (subconjunto de las instrucciones Intel 8080/8085).

lador y del registro de status (señalizadores). La instrucción PUSH *PSW* es una instrucción de 1 byte, que primero coloca el acumulador en la pila y después los señalizadores. Para una explicación más detallada de las instrucciones de introducir y sacar revisar la Sección 5.6.

La instrucción «no operación» no hace nada, excepto consumir 1 ó 2 μs. Es una instrucción de 1 byte que so' mente hace que se incremente el contador de programa. No afecta a los demás registros. Esta instrucción se utiliza como relleno cuando durante la depuración se eliminan una o dos sentencias del programa. Las instrucciones de no operación sirven de puente entre dos secciones del programa para que la MPU pueda avanzar de un segmento a otro; también puede utilizarse para consumir tiempo en un ciclo.

La instrucción de alto se utiliza al final del programa para que el microprocesador deje de procesar más «instrucciones» en la memoria de programa. Solamente una reinicialización o una entrada de petición de interrupción a la MPU puede arrancar al microprocesador genérico cuando esté en el modo de alto.

PROBLEMAS RESUELTOS

6.57. Acudir a la Figura 6.40. La operación PUSH *PSW* que se utiliza en la Figura 6.41 tiene un código de op de _____ (hex).

Solución:

De acuerdo con la tabla de la Figura 6.40, la operación PUSH *PSW* tiene el código de operación F5H.

6.58. Acudir a la Figura 6.40. Durante la ejecución de la instrucción PUSH *PSW* utilizada en la Figura 6.41, el puntero de pila es _____ (decrementado, incrementado) primero y el contenido del _____ (acumulador, registro de status) es transferido a la posición de la pila _____ (hex).

Solución:

Usar la tabla de la Figura 6.40 y el procedimiento de la Figura 5.16*b*. Durante la ejecución de la instrucción PUSH *PSW* utilizada en la Figura 6.41, el puntero de pila es decrementado primero y el contenido del acumulador es transferido a la posición de la pila 2209H.

6.59. Acudir a la Figura 6.40. Citar el contenido del puntero de pila y de las posiciones de la pila 2208H y 2209H después de la operación de introducir (push) de la Figura 6.41.

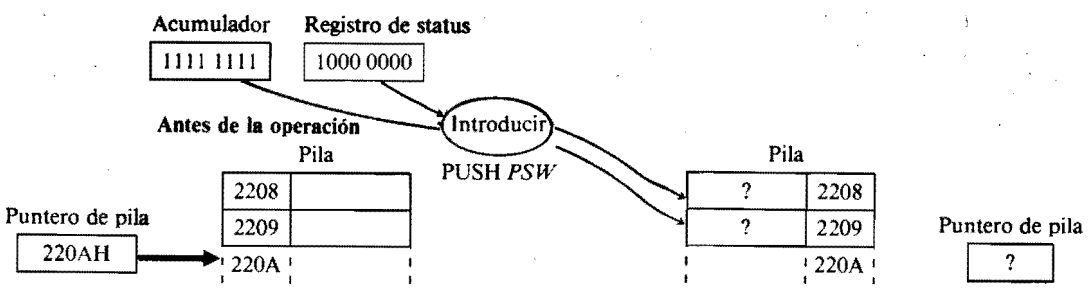


Figura 6.41.

Solución:

Usar como referencia la tabla de la Figura 6.40 y el procedimiento de la Figura 5.16*b*. Después de la operación de introducir (push), mostrada en la Figura 6.41, el puntero de pila señalará a la posición 2208H y su contenido será el del registro de status que es 10000000. El contenido de la posición de la pila 2209H será el del acumulador que es 11111111.

6.60. Acudir a la Figura 6.40. Si la instrucción PUSH *PSW* se utiliza para colocar el contenido de los registros de la CPU en la pila, entonces la instrucción _____ (nemotécnico) se utilizará para restaurar el contenido de estos registros.

Solución:

Si la instrucción PUSH *PSW* se utiliza para colocar el contenido de los registros de la CPU en la pila, entonces la instrucción POP *PSW* se utilizará para restaurar el contenido de estos registros.

6.61. Las instrucciones de introducir y sacar (push y pop) son utilizadas frecuentemente, a pares, en _____ (las reinicializaciones, las subrutinas).

Solución:

Estas instrucciones son utilizadas frecuentemente, a pares, en las subrutinas.

6.62. La instrucción _____ (alto, no operación) evitará que el microprocesador prosiga la búsqueda y ejecución de otra instrucción hasta que reciba una señal de reinicialización o de petición de interrupción.

Solución:

La instrucción de alto evitará que el microprocesador prosiga la búsqueda y ejecución de otra instrucción hasta que reciba una señal externa de reinicialización o de petición de interrupción. La instrucción de alto se coloca normalmente al final de un programa.

6.9. ESCRITURA DE UN PROGRAMA

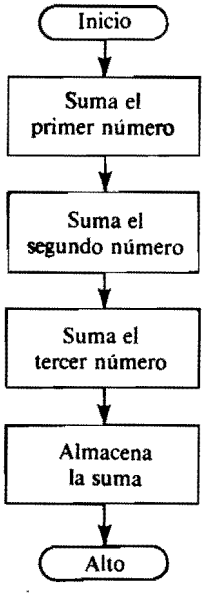
El programador de un sistema basado en microprocesador debe estar muy familiarizado con el repertorio de instrucciones. Las tablas de las Figuras 6.4, 6.14, 6.21, 6.30, 6.35 y 6.40 detallan las operaciones que pueden ser realizadas por el microprocesador genérico. El programador también debe estar familiarizado con los registros disponibles cuando escribe un programa. Un *modelo de programación* del microprocesador genérico se muestra en la Figura 6.3. Resume los registros que utiliza el programador, el cual también debe conocer la arquitectura general de la microcomputadora y comprender el mapa de memoria de la unidad.

Los puntos para desarrollar un programa pueden resumirse como sigue:

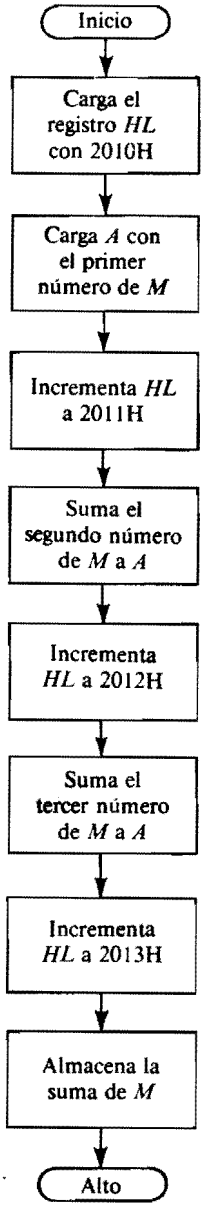
1. Definir y analizar el problema.
2. Dibujar un diagrama de flujo de la solución del problema.
3. Escribir el programa en lenguaje ensamblador.
4. Escribir o generar la versión en lenguaje máquina del programa.
5. Depurar el programa.
6. Documentar el programa.

Considerar la sencilla tarea de escribir un programa que sume el contenido de tres posiciones consecutivas de memoria y almacene la suma. Es común para los programadores construir una

carta o *diagrama de flujo* del problema. Un *diagrama de flujo funcional* para el problema ejemplo se muestra en la Figura 6.42, donde se ve el procedimiento general para sumar tres números consecutivos y almacenar la suma en memoria. Este diagrama de flujo carece de los detalles adecuados para traducirlo directamente en un segmento de programa en lenguaje ensamblador o lenguaje máquina.



(a) Diagrama de flujo funcional para sumar tres números



(b) Diagrama de flujo detallado

Figura 6.42.

En la Figura 6.42*b* se ha desarrollado un *diagrama de flujo detallado.* En este diagrama cada bloque rectangular corresponde a una orden u operación del microprocesador. El diagrama detallado es, por tanto, dependiente del sistema microprocesador específico que se está programando, mientras que el diagrama funcional puede utilizarse con cualquier MPU.

A continuación, se desarrolla una versión, en lenguaje ensamblador, del diagrama de flujo detallado; el segmento de programa se muestra en la Figura 6.43*a*. Los cuatro campos típicos —etiqueta, nemotécnico, operando y comentario— se listan en la parte superior. Cuidadosamente copiados debajo están los nemotécnicos *exactos* y los operandos del microprocesador. Recordar que el campo de etiqueta no necesita entradas; los comentarios son una gran ayuda para comprender la función de cada instrucción del segmento de programa. Al comparar el diagrama de flujos detallado, de la Figura 6.42*b*, con el listado del programa en lenguaje ensamblador, de la Figura 6.43*a*, se observa que cada bloque del diagrama de flujo tiene una sentencia del lenguaje

Etiqueta	Nemotécnico	Operando	Comentarios
	LXI	*H*,2010H	; Carga el registro par *HL* con la dirección 2010H (se utilizará como puntero de dirección)
	MOV	*A,M*	; Carga el acumulador con el primer número de la posición de memoria 2010H
	INX	*H*	; Incrementa el registro par *HL* a 2011H
	ADD	*M*	; Suma el segundo número de la posición de memoria 2011H al acumulador
	INX	*H*	; Incrementa el registro par *HL* a 2012H
	ADD	*M*	; Suma el tercer número de la posición de memoria 2012H al acumulador
	INX	*H*	; Incrementa el registro par *HL* a 2013H
	MOV	*M,A*	; Almacena la suma del acumulador en la posición de memoria 2013H
	HLT		; Detiene la MPU

(a) Programa en lenguaje ensamblador para sumar tres números y almacenar la suma

Figura 6.43.

Dirección (hex)	Contenido (hex)	Etiqueta	Nemotécnico	Operando	Comentarios
2020	21		LXI	*H*,2010H	; Carga el registro par *HL* con la dirección 2010H (se utilizará como puntero de dirección)
2021	10				
2022	20				
2023	7E		MOV	*A,M*	; Carga el acumulador con el primer número de la posición de memoria 2010H
2024	23		INX	*H*	; Incrementa el registro par *HL* a 2011H
2025	86		ADD	*M*	; Suma el segundo número de la posición de memoria 2011H al acumulador
2026	23		INX	*H*	; Incrementa el registro par *HL* a 2012H
2027	86		ADD	*M*	; Suma el tercer número de la posición de memoria 2012H al acumulador
2028	23		INX	*H*	; Incrementa el registro par *HL* a 2013H
2029	77		MOV	*M,A*	; Almacena la suma del acumulador en la posición de memoria 2013H
202A	76		HLT		; Detiene la MPU

(b) Programas combinados en lenguaje ensamblador y máquina

Figura 6.43. *(continuación).*

ensamblador equivalente. Tomar como ejemplo la primera caja rectangular de la Figura 6.42*b*. «Cargar el registro *HL* con 2010H» en este bloque es equivalente a la sentencia en lenguaje ensamblador «LXI *H*,2010H; Cargar el registro par *HL* con la dirección 2010H» de la Figura 6.43*a*.

Una vez escrita la versión en lenguaje ensamblador, el programa debe ser traducido a patrones de bits comprensibles por el microprocesador, es decir, a código máquina. Este proceso puede hacerse a mano o por programas de computadora especiales denominados ensambladores. La codificación manual se hace:

1. Utilizando las tablas del repertorio de instrucciones (Figuras 6.4, 6.14, 6.21, 6.30, 6.35 y 6.40 para la MPU genérica) para buscar los códigos de op de cada nemotécnico.
2. Determinando los operandos (datos o direcciones) cuando sean requeridos por una instrucción multibyte.
3. Asignando direcciones secuenciales de la memoria de programa a cada instrucción y operando.

Considerar nuestro problema ejemplo que sumaba los contenidos de tres posiciones consecutivas de memoria y almacenaba la suma en la cuarta posición. La versión en lenguaje ensamblador está reproducida a la derecha en la Figura 6.43*b*. Las dos columnas de la izquierda muestran las direcciones de memoria de programa asignadas y el código de op u operando hexadecimal. Los contenidos representan las palabras de 8 bits que deben ser almacenadas en la memoria del programa.

Considerar la primera sentencia del lenguaje ensamblador en la Figura 6.43*b*. El nemotécnico y operando se leen «LXI H,2010H», que deben ser asignados a un código de op y, en este caso, a dos operandos. La tabla de la Figura 6.21 contiene la información necesaria sobre este tipo de operación (instrucción de transferencia de datos); el nemotécnico «LXI H» se encuentra hacia la mitad de la tabla de la Figura 6.21, y su código de op, que es 21H, está escrito en la columna «Contenidos» de la Figura 6.43*b*. En la tabla de la Figura 6.21 se observa que esta instrucción requiere 2 bytes de datos extra; el byte del dato de orden inferior (10H en este ejemplo) se coloca, primero, en la dirección 2021H y el del dato de orden superior (20H en este ejemplo) se coloca, a continuación, en la dirección 2022H. El resto de los nemotécnicos del lenguaje ensamblador de la Figura 6.43*b* representan instrucciones de un solo byte, cuyos códigos de op hexadecimales son buscados en las tablas adecuadas del repertorio de instrucciones y registrados en la columna de «Contenidos», como muestra la Figura 6.43*b*. El programa de las columnas izquierdas está en código máquina y puede ser introducido en el sistema microprocesador.

El programa se examina a continuación para ver si funciona adecuadamente, a este proceso se le denomina *depuración* (sacar los errores o «chinches»). El procedimiento, de sentido común, utilizado incluye la comparación del diagrama de flujo con el programa, doble comprobación de los códigos de op e ir paso a paso a través del programa revisándolo con un problema simulado. A continuación, es común introducir el programa y datos conocidos en un sistema microprocesador y comprobar las respuestas. Habitualmente se realizan varias pruebas de los problemas.

El último paso en la programación es la documentación. La documentación es una descripción del programa diciendo lo que debe realizar y cómo utilizarlo, incluyendo detalles como diagramas de flujo, listados del programa, direcciones de datos y programas utilizados y comentarios. La documentación se complica mucho en los grandes programas, los cuales necesitan niveles de documentación tales como documentos de diseño, manuales de operador y documentos descriptivos.

Considerar el problema ejemplo del segmento de programa de la Figura 6.44. Aquí se ha introducido el código máquina en la memoria del programa (direcciones 2020-202AH). Los números que se van a sumar (0F, 09, 06H) en el problema ejemplo se han introducido en las posiciones 2010 a 2012H de la memoria de datos. Observar si se puede seguir paso a paso la operación de este programa para sumar 0F + 09 + 06, y obtener una suma de 1EH. Este programa puede ejecutarse realmente en la mayoría de los microprocesadores de entrenamiento basados en el 8080/8085.

Aunque los programas reales de microcomputadora son bastante largos, están formados por un grupo de segmentos de programa similares al del ejemplo. Normalmente, cada segmento se escribe y comprueba antes de que el programa total se unifique para la prueba final (test). El segmento del programa ejemplo desarrollado es sólo una forma posible de resolver el problema propuesto. El programa desarrollado se denomina programa lineal ya que no tiene saltos o bifurcaciones.

Memoria de datos

Dirección (hex)	Contenido (hex)	
2010	0F	
2011	09	
2012	06	
2013		← Suma

Memoria de programa

2020	21	Instrucción 1; carga *HL*
2021	10	
2022	20	
2023	7E	Instrucción 2; carga *A* con el primer número
2024	23	Instrucción 3; incrementa *HL*
2025	86	Instrucción 4; suma el segundo número
2026	23	Instrucción 5; incrementa *HL*
2027	86	Instrucción 6; suma el tercer número
2028	23	Instrucción 7; incrementa *HL*
2029	77	Instrucción 8; almacena la suma
202A	76	Instrucción 9; alto

Figura 6.44. Contenidos de memoria e instrucciones para sumar tres números y almacenar la suma.

PROBLEMAS RESUELTOS

6.63. El programador de un sistema microprocesador debe estar familiarizado con la arquitectura del sistema, el repertorio de instrucciones de la MPU, el mapa de memoria del sistema y los registros de la CPU. Un resumen de los registros de la MPU disponibles al programador se da en un diagrama denominado _____ (modelo, conjunto) de programación.

Solución:

Un resumen de los registros de la MPU disponibles al programador se da en un diagrama denominado modelo de programación.

6.64. Citar seis pasos secuenciales en el desarrollo de un programa de un sistema basado en microprocesador, como se ha sugerido en esta sección.

Solución:

Los seis pasos secuenciales en el desarrollo de un programa citados en esta sección son: definir y analizar el problema, dibujar el diagrama de flujo de la solución del problema, escribir la versión del programa en lenguaje ensamblador, escribir o generar la versión del programa en lenguaje máquina, depurar el programa y documentarlo.

6.65. La representación gráfica de una solución de problema se denomina _____ .

Solución:

La representación gráfica de una solución del problema se denomina diagrama o carta de flujo. Dos ejemplos se muestran en la Figura 6.42.

6.66. Después de dibujar un diagrama de flujo detallado (como, por ejemplo, el de la Figura 6.42*b*), el siguiente paso de programación es escribir el programa en lenguaje _____ (ensamblador, máquina).

Solución:

El programa en lenguaje ensamblador, normalmente se escribe después de dibujar un diagrama de flujo detallado.

6.67. Al escribir la versión en lenguaje máquina de un programa, las instrucciones son asignadas en _____ (direcciones, interrupciones) secuenciales de la memoria de programa.

Solución:

Al escribir la versión en lenguaje máquina de un programa, las instrucciones son asignadas en direcciones secuenciales de la memoria del programa.

6.68. Al escribir la versión en lenguaje máquina de un programa, los _____ (ASCII, códigos de op) y operandos son listados en _____ (decimal, hexadecimal) para cada posición del programa.

Solución:

Al escribir una versión de un programa en lenguaje máquina, los códigos de op y operandos son listados en forma hexadecimal para cada posición del programa.

6.69. ¿Cuál es el contenido del registro par *HL* después de la ejecución de la instrucción 1 de la Figura 6.44?

Solución:

Acudir a la Figura 6.43. El contenido del registro par *HL* es 2010H después de la ejecución de la instrucción 1 de la Figura 6.44.

6.70. ¿Cuál es el contenido del acumulador después de la ejecución de la instrucción 4 de la Figura 6.44?

Solución:

Sumar 0F + 09 = 18H. El contenido del acumulador después de la ejecución de la instrucción 4 de la Figura 6.44 es 18H.

6.71. ¿Cuál es el contenido del acumulador después de la ejecución de la instrucción 6 de la Figura 6.44?

Solución:

Sumar 0F + 09 + 06 = 1EH. El contenido del acumulador después de la ejecución de la instrucción 6 de la Figura 6.44 es 1EH.

6.72. ¿Cuál es el contenido de la posición 2013H de la memoria de datos después de la ejecución de la instrucción 7 en la Figura 6.44?

Solución:

La suma *no* se ha almacenado todavía en la posición 2013H de la memoria de datos, y por tanto su contenido es, de alguna forma, un patrón aleatorios de bits.

6.73. ¿Cuál es el contenido de la posición 2013H de la memoria de datos después de la ejecución de la instrucción 8 de la Figura 6.44?

Solución:

La suma de 0F + 09 + 06 = 1EH es almacenada en la posición 2013H de la memoria de datos después de la ejecución de la instrucción 8 de la Figura 6.44.

6.74. Encontrar y corregir los errores en un programa es un proceso denominado comúnmente _____ (depuración, detección de fallos).

Solución:

Encontrar y corregir errores en un programa es un proceso denominado, comúnmente, depuración.

6.75. El paso final en la escritura de un programa se denomina _____ .

Solución:

El paso final en la escritura de un programa se denomina documentación. Esta debe incluir varios niveles como, por ejemplo, documentación de diseño, un manual de operador y documentación descriptiva.

6.10. MODOS DE DIRECCIONAMIENTO

Considerar la instrucción de sumar del microprocesador. En el microprocesador genérico se suponía que uno de los números que se iban a sumar estaba en el acumulador. Sin embargo, ¿dónde estaba el segundo número y cómo se recuperaba? Los distintos *métodos de recuperación*

se denominan *modos de direccionamiento* del microprocesador. Cada microprocesador tiene sus propios métodos de recuperación, o modos de direccionamiento. Estos deben ser bien comprendidos por el programador. Los modos de direccionamiento se han mencionado muchas veces en las secciones anteriores, pero debido a que muchos estudiantes tienen dificultad con los modos de direccionamiento, en esta sección se revisarán los utilizados por el microprocesador genérico.

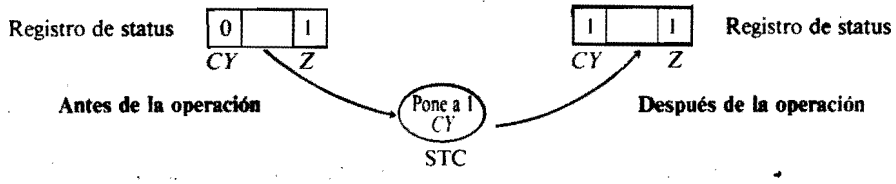
Los modos de direccionamiento listados en el repertorio de instrucciones del microprocesador genérico son:

1. Inherente.
2. De registro.
3. Inmediato.
4. Directo.
5. Indirecto de registro.

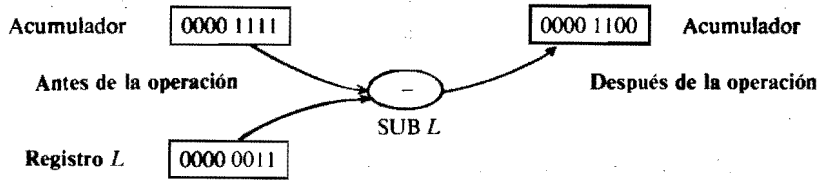
Los dos primeros modos de direccionamiento de la lista (inherente y de registro) se refieren a operandos que están localizados en la misma CPU. Los últimos tres modos de direccionamiento (inmediato, directo e indirecto de registro) se refieren a operandos que están localizados fuera de la CPU, en posiciones de memoria o puertos de E/S. Estos son los mismos modos de direccionamiento utilizados por el microprocesador Intel 8080/8085.

Considerar un ejemplo de la *instrucción inherente:* «poner a 1 el señalizador de arrastre» del grupo de transferencia de datos de la Figura 6.21. Las instrucciones inherentes son siempre de 1 byte porque no se requiere ningún dato para completar la operación y todas las acciones ocurren en la CPU. La acción de la instrucción «poner a 1 el señalizador de arrastre» del microprocesador genérico está esbozada en la Figura 6.45*a*. No es necesario recuperar ningún dato ni dirección de otros registros de la CPU, posiciones de memoria o puertos de E/S. La instrucción STC pone a 1 el señalizador de arrastre (*CY*) sin afectar a los demás registros o señalizadores.

En las *instrucciones de registro,* el operando es recuperado de un registro interno de la CPU. Considerar el ejemplo de la instrucción de registro, «restar *L* de *A*» del grupo aritmético de la



(a) La instrucción de poner a 1 el señalizador de registro (direccionamiento inherente)



(b) La instrucción restar *L* de *A* (direccionamiento de registro)

Figura 6.45.

Figura 6.4. Las operaciones de registro (como las instrucciones inherentes) son siempre instrucciones de 1 byte, ya que no se necesitan datos ni direcciones fuera de la CPU y todas las acciones ocurren en la CPU. La acción de la instrucción «restar *L* de *A*» se detalla en la Figura 6.45*b*. El minuendo (00001111 en este ejemplo) está en el acumulador, mientras que el sustraendo (00000011 en este ejemplo) está contenido en el registro *L*; después de la operación de resta interna, la diferencia (00001100 en este ejemplo) es transferida al acumulador.

En las *instrucciones inmediatas*, el operando viene del siguiente byte (o a veces los 2 siguientes) de la *memoria del programa*. El operando está *inmediatamente después* del código de op de la instrucción. Considerar el ejemplo de la instrucción inmediata, «cargar *SP* con datos» del grupo de transferencia de datos de la Figura 6.21. A partir de la tabla de la Figura 6.21 se determina que es una instrucción de 3 bytes. Generalmente, las operaciones inmediatas se especifican como instrucciones de 2 ó 3 bytes. El ejemplo de la Figura 6.46 muestra 2 bytes de datos que se recuperan de la memoria del programa y se cargan en el puntero de pila (*SP*). El byte de orden inferior (01110000 en este ejemplo) se carga primero, y después el byte de orden superior (00000010 en este ejemplo). Las instrucciones de modo inmediato son convenientes para inicializar los registros de la CPU o el puntero de pila.

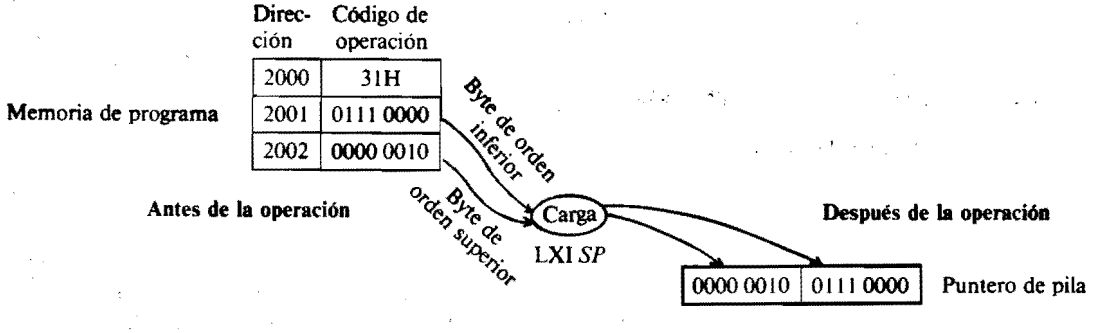


Figura 6.46. La instrucción de cargar el puntero de pila (direccionamiento inmediato).

En las *instrucciones directas*, el segundo y tercer bytes de la memoria del programa *apuntan directamente a la dirección* del operando en la memoria de datos. El segundo y tercer bytes de la memoria del programa son direcciones en una instrucción directa, mientras que eran operandos en el caso del direccionamiento inmediato. Considerar el ejemplo de la instrucción directa, «entrar a *A* desde el puerto LOC *a*» del grupo de transferencia de datos de la Figura 6.21. A partir de la tabla de la Figura 6.21 se determina que se trata de una instrucción de 2 bytes. Generalmente, las operaciones directas se especifican como instrucciones de 2 ó 3 bytes. El ejemplo de la Figura 6.47 muestra el contenido del puerto de entrada (11000011 en este ejemplo) transferido al acumulador del microprocesador. El puerto exacto de entrada está señalado por la dirección del segundo byte de la instrucción del programa (esta dirección es 00001010, ó 10_{10} en este ejemplo.

En *las instrucciones indirectas de registro*, el registro par especial *HL apunta a la dirección* del operando en la memoria de datos. Considerar el ejemplo de la instrucción indirecta de registro «cargar LOC (H & L) en *A*» (nemotécnico de MOV *A,M*) del grupo de transferencia de datos de la Figura 6.21. A partir de la tabla de la Figura 6.21 se determina que es una instrucción de 1 byte.

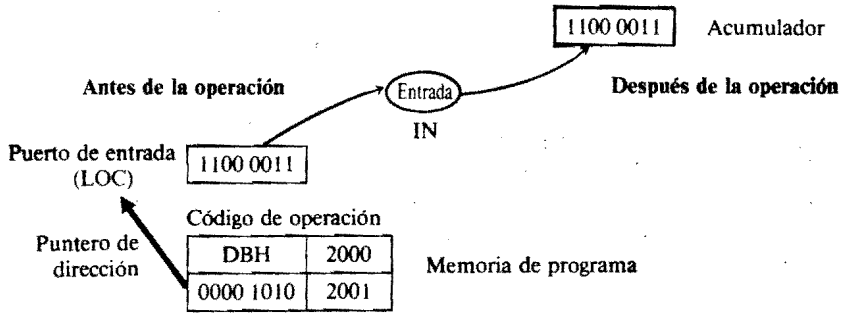


Figura 6.47. La instrucción de entrar (direccionamiento directo).

Las operaciones indirectas de registro se especifican siempre por instrucciones de 1 byte. El ejemplo de la Figura 6.48 muestra el contenido de la posición 2080H (11110000 en este ejemplo) de memoria cargándose en el acumulador. La dirección (2080H en este ejemplo) de la posición correcta de la memoria de datos está señalada por el contenido del registro par *HL*. Este registro se denomina registro de datos/direcciones del microprocesador genérico. En este caso el registro *HL* se está utilizando como un registro de direcciones.

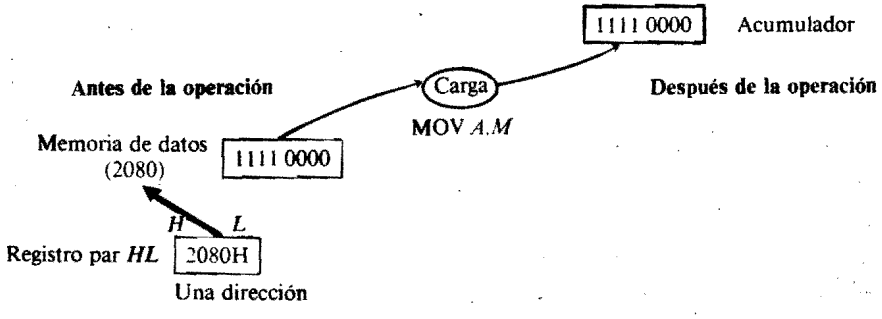


Figura 6.48. La instrucción cargar *A* desde memoria (direccionamiento indirecto de registro).

Otros microprocesadores tienen variaciones en el modo de direccionamiento directo, denominándose por nombres tales como página cero o direccionamiento de página base, direccionamiento absoluto, direccionamiento de página actual y direccionamiento extendido. Otros modos no utilizados por nuestro microprocesador genérico incluyen el muy popular direccionamiento indexado y también el direccionamiento relativo.

PROBLEMAS RESUELTOS

6.76. Citar los cinco modos de direccionamiento utilizados por el microprocesador genérico.

Solución:

Los cinco modos de direccionamiento utilizados por el microprocesador genérico son inherente, de registro, inmediato, directo e indirecto de registro. Estos modos de direccionamiento también se utilizan en el microprocesador 8080/8085.

6.77. Cuando se utiliza una instrucción _____ (directa, inherente), no es necesario recuperar ningún operando.

Solución:

Cuando se utiliza una instrucción inherente, no se necesita recuperar ningún operando.

6.78. Cuando se utilizan el segundo y tercer bytes de la instrucción para apuntar a la dirección del operando, se está utilizando el modo de direccionamiento _____ (directo, inmediato).

Solución:

Cuando se utilizan el segundo y tercer bytes de la instrucción para apuntar a la dirección del operando, se está utilizando el modo de direccionamiento directo.

6.79. Cuando se usa una instrucción «suma inmediato», el operando se recupera de la memoria de _____ (datos, programa).

Solución:

Cuando se utiliza una instrucción «suma inmediato», el operando se recupera de la memoria de programa.

6.80. Cuando se utiliza una instrucción de _____ (registro, registro indirecto), el registro par *HL* de la CPU apunta a la dirección del operando.

Solución:

Cuando se utiliza una instrucción de registro indirecto, el registro par *HL* de la CPU apunta a la dirección del operando.

6.81. Cuando ambos operandos (como, por ejemplo, minuendo y sustraendo en un problema de resta) están localizados en los registros internos de la CPU, puede utilizarse el modo de direccionamiento _____ (inmediato, registro) para recuperar los operandos.

Solución:

Cuando ambos operandos están localizados en registros internos, el modo de direccionamiento de registro puede ser utilizado para recuperar los operandos.

6.82. Las operaciones de registro e inherentes están especificadas por una instrucción de _____ (número) bytes.

Solución:

Las operaciones de registro e inherentes están especificadas por instrucciones de 1 byte.

6.83. Las operaciones que utilizan el modo de direccionamiento directo están especificadas por instrucciones de _____ (número) bytes.

Solución:

Las operaciones que utilizan el modo de direccionamiento directo están especificadas por instrucciones de 2 ó 3 bytes.

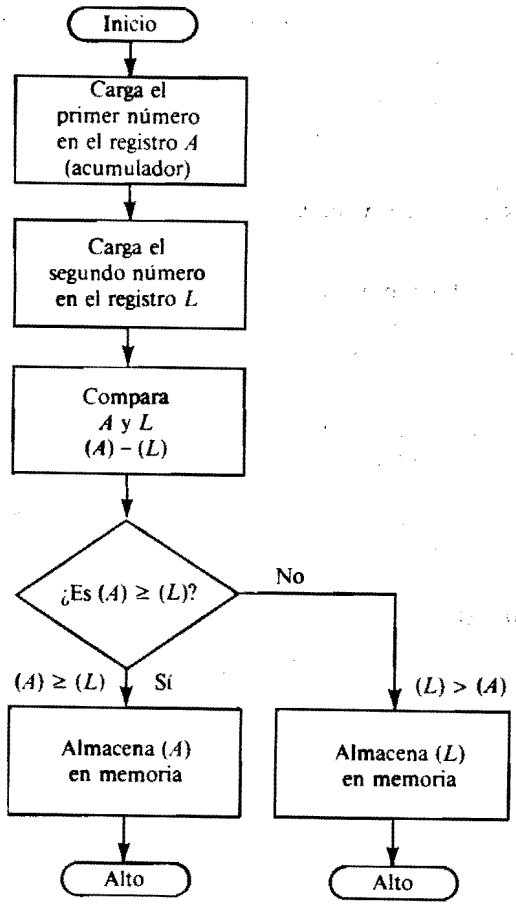
6.84. Las operaciones que utilizan el modo de direccionamiento indirecto de registro están especificadas por instrucciones de _____ (número) bytes.

Solución:

Las operaciones que utilizan el modo de direccionamiento de registro indirecto están especificadas por instrucciones de 1 byte.

6.11. BIFURCACION EN PROGRAMAS

Considerar la sencilla tarea de decidir cuál es el mayor de dos números y almacenar el mayor en una posición específica de memoria. Un diagrama de flujo para resolver este problema se muestra en la Figura 6.49*a*. Comenzando por la parte superior del diagrama de flujo, los dos primeros rectángulos representan la operación de cargar los dos números en los registros *A* (acumulador) y *L* (datos). El siguiente rectángulo representa la operación de comparación, donde el contenido del registro *L* se resta del contenido del registro *A*. Las operaciones de comparación no afectan el contenido de los registros pero sí a los señalizadores. La siguiente caja, en forma de diamante, del diagrama de flujo se llama *símbolo de decisión,* que representa un punto de decisión en el programa. Siempre se realiza una pregunta en este símbolo de decisión. En este ejemplo, el símbolo presenta la pregunta: ¿Es $(A) \geq (L)$? Si la respuesta es afirmativa, el programa continúa y el contenido del registro *A* es almacenado en la memoria y después el procesador se detiene por el símbolo de «stop». Sin embargo, si la respuesta a la pregunta del símbolo de decisión, es negativa,



(a) Diagrama de flujo detallado para comparar y almacenar el número mayor

Figura 6.49.

Etiqueta	Nemotécnico	Operando	Comentarios
	MVI	*A*,0FH	; Carga acumulador con el primer número (15_{10})
	MVI	*L*,06H	; Carga el registro *L* con el segundo número (6_{10})
	CMP	*L*	; Compara (*A*) y (*L*) Señalizador *CY* = 1 si *A* < *L*
	JC	STORE *L*	; Salta a la posición de STORE *L* si *CY* = 1 (si *A* < *L*); de otra forma continúa con la siguiente instrucción
	STA	2040H	; Almacena (*A*) en la posición de memoria 2040H
	HLT		; Detiene la MPU después de almacenar (*A*)
STORE *L*	MOV	*A*,*L*	; Transfiere el contenido del registro *L* al acumulador
	STA	2040H	; Almacena (*A*) en la posición de memoria 2040H
	HLT		; Detiene la MPU después de almacenar el contenido de registro *L*

(b) Programa en lenguaje ensamblador utilizando la bifurcación «almacenar *A*»

Etiqueta	Nemotécnico	Operando	Comentarios
	MVI	*A*,0AH	; Carga el acumulador con el primer número (10_{10})
	MVI	*L*,0EH	; Carga el acumulador con el segundo número (14_{10})
	CMP	*L*	; Compara (*A*) y (*L*) Señalizador *CY* = 1 si *A* < *L*
	JC	STORE *L*	; Salta a la posición de STORE *L* si *CY* = 1 (si *A* < *L*); de otra forma continúa
	STA	2040H	; Almacena (*A*) en la posición de memoria 2040H
	HLT		; Detiene la MPU después de almacenar (*A*)
STORE *L*	MOV	*A*,*L*	; Transfiere el contenido del registro *L* al acumulador
	STA	2040H	; Almacena (*A*) en la posición de memoria 2040H
	HLT		; Detiene la MPU después de almacenar el contenido de registro *L*

(c) Programa en lenguaje ensamblador utilizando la bifurcación «almacenar *L*»

Figura 6.49. *(continuación).*

el programa bifurca a la derecha y el contenido del registro L es almacenado en memoria, y después se para el procesador. El uso de símbolos de decisión en un diagrama de flujo nos conduce a lo que se denomina bifurcación en un programa o programa con ruptura de secuencia.

La versión en lenguaje ensamblador del programa que compara dos números y coloca el valor del mayor en la posición de memoria 2040H está detallado en la Figura 6.49*b*. Observar que en el programa ejemplo de la Figura 6.49*b*, el acumulador (A) se carga con 15_{10} mientras que el registro L se carga con 6_{10}. Después de comparar (la tercera instrucción en este ejemplo), se encuentra que (A) > (L) y, por tanto, el señalizador de arrastre (CY) se pone a 0. La instrucción de saltar (nemotécnico de JC en este ejemplo) examina el señalizador de arrastre y encuentra que $CY = 0$, lo que hace que la MPU continúe con la siguiente instrucción de la secuencia, que es «almacenar (A) en la dirección de memoria 2040H». El número mayor (0FH) se almacena en la posición de memoria 2040H mediante la operación «STA». A continuación la MPU la detiene por la instrucción de alto (HLT). Las tres últimas instrucciones, sombreadas, del programa en lenguaje ensamblador de la Figura 6.49*b* no se utilizan en este ejemplo. En el diagrama de flujo que se muestra en la Figura 6.49*a*, el programa continúa linealmente y no bifurca a la derecha.

Considerar el mismo programa en lenguaje ensamblador pero con el contenido del acumulador menor que el contenido del registro L. Un listado de dicho programa se muestra en la Figura 6.49*c*. En este ejemplo el acumulador se carga con 0AH (10_{10}), mientras que el registro L se carga con el valor mayor de 0EH 14_{10}). La instrucción de comparar pone a 1 el señalizador CY porque (A) < (L); la operación «salta si señalizador de arrastre es 1» (nemotécnico JC en este ejemplo), examina el señalizador CY y comprueba que está a 1. La instrucción de salto hace entonces que el contador de programa cambie a la dirección de la posición de la siguiente instrucción (STORE L en este ejemplo). Observar que la MPU ha saltado dos instrucciones para llegar a la *dirección simbólica* STORE L que aparece listada en la columna «etiquetas». La MPU entonces ejecuta las tres últimas instrucciones, las cuales almacenan el contenido del registro L (el número mayor, 0EH en este ejemplo) en la posición de memoria 2040H. En el diagrama de flujo de la Figura 6.49*a*, el programa progresaba al símbolo de decisión y después bifurcaba a la derecha y seguía hasta alcanzar el símbolo «alto» inferior derecho.

Observar de nuevo las tres últimas instrucciones de la Figura 6.49*c*. El objeto de estas instrucciones es almacenar el contenido del registro L en la posición de memoria 2040H. Según la tabla del repertorio de instrucciones de la Figura 6.21 el microprocesador genérico no tiene ninguna instrucción para almacenar directamente en memoria el registro L. Por tanto, ha sido necesario transferir primero el contenido del registro L a A y después almacenar A en memoria. A veces se encuentra que los microprocesadores con repertorios limitados de instrucciones (como la MPU genérica) son más difíciles de programar que las unidades con repertorios de mayor alcance.

En resumen, las técnicas de bifurcación observadas en el diagrama de flujo se ejecutan por instrucciones de salto (a veces denominadas de bifurcación). Las instrucciones de bifurcación condicional toman una decisión según la condición de un señalizador particular del registro de status. La técnica de bifurcación es muy utilizada en la mayoría de los programas.

PROBLEMAS RESUELTOS

6.85. La caja en forma de diamante de la Figura 6.49*a* se denomina _____ _____ (símbolo de decisión, diamante de bifurcación).

Solución:

La caja en forma de diamante de la Figura 6.49*a* se denomina símbolo de decisión.

6.86. La instrucción «comparar *A* con *L*» de la Figura 6.49*b* _____ (cambia, no cambia) el valor del acumulador.

Solución:

Las instrucciones de comparación no cambian el valor del acumulador aún cuando se realice una resta interna ($A - L$ en este ejemplo).

6.87. La instrucción «comparar *A* con *L*» de la Figura 6.49*b* afecta al señalizador _____ (*CY*, *Z*) en este programa.

Solución:

La instrucción «comparar *A* con *L*» de la Figura 6.49*b* afecta al señalizador *CY* en este programa. El señalizador de cero (*Z*) también es afectado pero no es comprobado posteriormente por ninguna instrucción de bifurcación en este programa.

6.88. Acudir a la Figura 6.49. Si el contenido del acumulador (*A*) y del registro *L* son iguales, el señalizador de arrastre es puesto a _____ (0, 1) por la instrucción de comparación y la bifurcación _____ («No», «Sí») se toma en el símbolo de decisión de la Figura 6.49*a*.

Solución:

Si $(A) = (L)$ en la Figura 6.49*a*, el señalizador de arrastre es puesto a 0 por la instrucción de comparación y se toma la bifurcación «Sí» en el símbolo de decisión.

6.89. Acudir a la Figura 6.49. Si el registro *A* contiene 1AH y el registro *L* contiene A5H, la instrucción de salto hará que la CPU _____ (continúe con la siguiente instrucción, salte a la posición STORE L).

Solución:

Si $A = 1AH$ y $L = A5H$ en la Figura 6.49, la instrucción de salto hará que la CPU salte a la posición STORE *L*. El salto es provocado, porque el señalizador de arrastre es puesto a 1 por la instrucción de comparación. La instrucción de comparación pone a 1 el señalizador de arrastre porque $A < L$.

6.90. Acudir a la Figura 6.49*b* y *c*. El operando siguiente a la instrucción «JC» se lista como STORE *L*; esto se denomina una dirección _____ (absoluta, simbólica).

Solución:

El operando de la Figura 6.49*b* y *c* que sigue a la instrucción «JC» está listado como STORE *L*; esto se denomina una dirección simbólica cuando se utiliza en un programa en lenguaje ensamblador.

6.91. Las instrucciones de bifurcación o de salto se utilizan _____ (con frecuencia, rara vez) en los programas reales de computadora.

Solución:

Las instrucciones de bifurcación o saltos se utilizan con frecuencia en los programas reales de computadoras.

6.12. PROGRAMAS CICLICOS

Los microprocesadores son especialmente útiles para realizar tareas repetitivas. Por ejemplo, el diagrama de flujo de la Figura 6.50 representa un programa que cuenta de 0 a 254 (0-FEH) y saca la cuenta a un dispositivo periférico adecuado. Este programa tiene las características de la mayor parte de los programas cíclicos (con bucles o lazos). Comenzando en la parte superior, es costumbre *inicializar* los registros. En este ejemplo el acumulador (A) se pone a 00H, a continuación el contenido del acumulador se saca al dispositivo periférico adecuado, lo que denomina el *proceso repetido*, después se incrementa el contenido del acumulador o la cuenta se *modifica*. Los símbolos de comparación y decisión realizan el *procedimiento de prueba* para determinar si la cuenta del acumulador ha alcanzado ya FFH. Si la respuesta a la pregunta en la caja de decisión (¿Es A = FFH? en este ejemplo) es negativa el programa bifurca a la izquierda y vuelve al bloque de salida. Esto se denomina *ciclo o bucle*. El programa seguirá el ciclo 254_{10} veces y después sale por el símbolo de alto. El diagrama de flujo de la Figura 6.50 representa un programa que puede tener de 15 a 30 instrucciones. Si los 255_{10} pasos del programa se hubiesen programado de forma lineal, el listado tendría miles de instrucciones. Los ciclos del programa son una forma adecuada de simplificar programas que tengan procesos repetitivos.

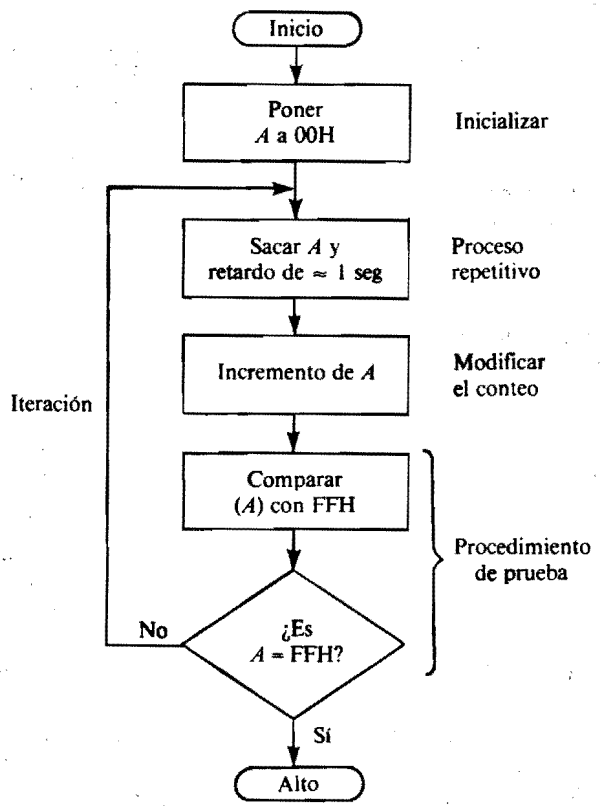
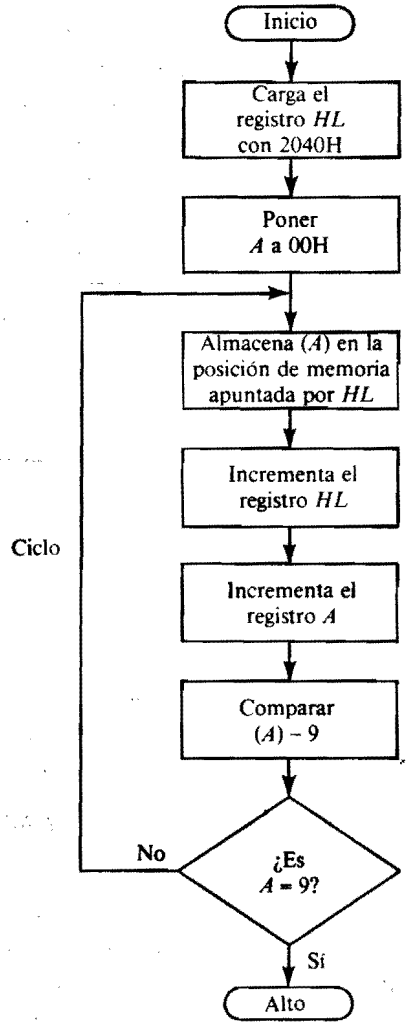


Figura 6.50. Diagrama de flujo del problema de cuenta que utiliza un programa cíclico.

Otro ejemplo de programa cíclico se muestra en la Figura 6.51. Este segmento de programa coloca la secuencia de números de 0 al 8 en las posiciones secuenciales de memoria 2040 2048H. Acudir al diagrama de flujo de la Figura 6.51*a*. Los dos primeros rectángulos de la parte superior son para inicializar el registro par *HL* con 2040H y el acumulador con 00H. El tercer rectángulo representa el proceso de almacenamiento, que se repetirá nueve veces en este programa. La MPU almacenará repetidamente el contenido del acumulador en la posición de memoria señalada por el registro par *HL*. En el primer pase, el contenido del acumulador (00H) se almacena en la posición de memoria 2040H.

Los rectángulos cuarto y quinto de la Figura 6.51*a* representan operaciones que modifican la



(a) Diagrama de flujo detallado del programa de «cuenta y almacena»

Figura 6.51.

Etiqueta	Nemotécnico	Operando	Comentarios
	LXI	*H*,2040	; Carga el registro *HL* con 2040H (utilizado como puntero de dirección)
	XRA	*A*	; Pone el acumulador a 00H [(*A*) ⊕ (*A*) = 00H en acumulador]
LOOP	MOV	*M,A*	; Almacena el contenido del acumulador en la posición de memoria apuntada por el registro *HL*
	INX	*H*	; Incrementa el registro *HL*
	INR	*A*	; Incrementa el acumulador
	CPI	09H	; Compara ¿es todavía (*A*) igual a 09 *H*? Si (*A*) = 09H, entonces el señalizador de 0 se coloca en 1
	JNZ	LOOP	; Salta a la posición LOOP si *Z* = 0[si (*A*) < 09H]; de otra forma continúa con la siguiente instrucción
	HLT		; Detiene la MPU

(b) Versión en lenguaje ensamblador del programa de «cuenta y almacena»

Figura 6.51. *(continuación).*

dirección del registro par *HL* y la cuenta del registro *A*. A título de ejemplo, durante el primer pase a través del programa, el registro par *HL* es incrementado a 2041H mientras que el acumulador es incrementado a 01H.

El rectángulo de comparación y el símbolo de decisión representan las operaciones de prueba. El rectángulo de comparación resta 09H del contenido de *A* para poner a 0 ó a 1 el señalizador de cero. Si $A < 9$, el señalizador de cero estará a 0, sin embargo, si $A = 9$, entonces el señalizador de cero estará a 1; el símbolo de decisión hace la pregunta: ¿Es $A = 9$? Si la respuesta es sí, el programa sale del ciclo y finaliza, sin embargo, si la respuesta es no, entonces el programa bifurca hacia atrás a la instrucción de almacenar, cercana a la parte superior del diagrama. Muchos procedimientos cíclicos utilizan las instrucciones de comparación y salto condicional para examinar una cuenta que se modifica y que determina cuándo finaliza el ciclo. La cuenta modificada que se examina en este ejemplo es el contenido del acumulador. El programa representado por el diagrama de flujo de la Figura 6.51*a* repetirá el proceso de almacenamiento nueve veces antes de salir por el símbolo de alto.

Una versión en lenguaje ensamblador de un programa que funciona como especifica el diagrama de flujo se muestra en la Figura 6.15*b*. El programa almacenará los números 0 a 8 en las posiciones secuenciales de memoria 2040 a 2048H. Las dos primeras instrucciones son equivalentes a los dos primeros bloques del diagrama de flujo, y están para inicializar el registro par *HL* con 2040H y el acumulador con 00H. La instrucción MOV *M,A* almacena el contenido del acumulador en la posición de memoria a la que apunta el registro par *HL;* las dos instrucciones siguientes (INX *H* e INR *A*) incrementan los registros *HL* y *A*, la instrucción CPI compara el contenido de *A* con la constante 09H. Si el registro *A* contiene un valor de 0 a 8, el señalizador de cero está a 0, pero si el contenido de *A* es 9, el señalizador de cero se pone a 1. La instrucción JNZ examina el señalizador de cero, si es = 0, el resultado de la resta interna (*A*) – 09H no es cero, y por tanto se realiza un salto a la posición especificada por la dirección simbólica LOOP. Si el señalizador de cero es = 1, el resultado de la resta interna (*A*) – 09H es cero y, por tanto, el programa sale del ciclo y continúa con la siguiente instrucción, que es HLT, la cual detiene el programa.

PROBLEMAS RESUELTOS

6.92. _____ (hacer ciclos, modular) en un programa es un método de acortar los programas que deben realizar tareas repetitivas.

Solución:

La realización de ciclos es un método de acortar los programas que deban realizar operaciones repetitivas.

6.93. Acudir a la Figura 6.51*a*. El rectángulo «poner *A* a 00H» representa una instrucción utilizada para _____ (inicializar, examinar) el contenido del acumulador.

Solución:

El rectángulo «poner *A* a 00H» de la Figura 6.51*a* representa una instrucción utilizada para inicializar el contenido del acumulador.

6.94. Acudir a la Figura 6.51*a*. Si el contenido del acumulador es 8 en el bloque de comparación, el programa _____ (sigue el ciclo, se detiene) después del bloque de decisión.

Solución:

Si el contenido del acumulador de la Figura 6.51*a* es 8 en el bloque de comparación, el programa sigue el ciclo después del bloque de decisión. Observar que el bloque de decisión pregunta: ¿Es *A* = 9? La respuesta es no, y por tanto el programa sigue el ciclo.

6.95. Acudir a la Figura 6.51*b*. La operación «CPI» _____ (compara, complementa) el contenido del registro _____ (*A, HL*) con la constante 09H. Esta operación se realiza por una resta interna.

Solución:

La operación «CPI» de la Figura 6.51*b* compara el contenido del registro *A* con la constante 09H. Acudir a la tabla de la Figura 6.4 para información detallada sobre la instrucción de comparación inmediata.

6.96. Acudir a la Figura 6.51*b*. Si el señalizador de cero se pone a 1, la instrucción JNZ hará que la MPU _____ (continúe con la siguiente instrucción, vuelva atrás en el ciclo a la instrucción MOV *M,A*).

Solución:

Si el señalizador de cero es puesto a 1 en la Figura 6.51*b*, la instrucción JNZ hará que la MPU continúe con la siguiente instrucción.

6.97. En un ciclo de un programa, la MPU ejecuta un grupo de operaciones repetidamente teniendo la posibilidad de _____ (cambiar, detener) las repeticiones en el instante adecuado.

Solución:

El ciclo de un programa hace que la MPU ejecute un grupo de operaciones repetidamente teniendo la posibilidad de detener las repeticiones en el instante adecuado. Hay algunos programas con ciclos que continúan en el ciclo hasta que son interrumpidos por una reinicialización y por tanto no tienen la posibilidad de salir del programa por medios software.

6.98. Acudir a la Figura 6.51*a*. Este programa repite la operación «almacenar (*A*) en memoria» nueve veces, y el programa sigue el camino del ciclo hasta que salga por el símbolo de decisión de forma de diamante _____ (número) veces.

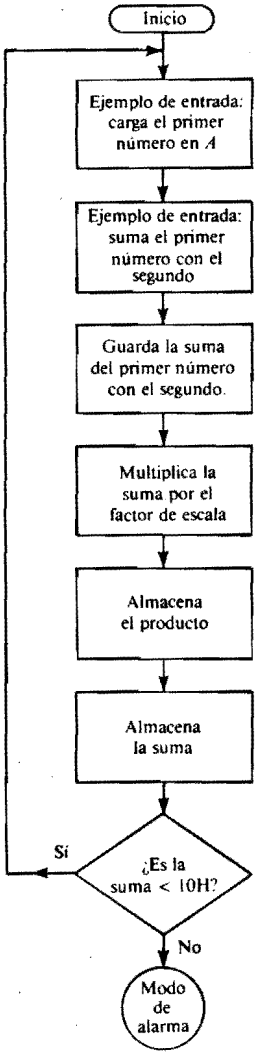
Solución:

El programa representado en la Figura 6.51*a* repite la operación «almacenar (*A*) en memoria» nueve veces aunque el programa realiza el ciclo, hasta el símbolo de decisión en forma de diamante, solamente ocho veces. Esto es debido a que antes del primer ciclo se ejecutó una operación de almacenamiento.

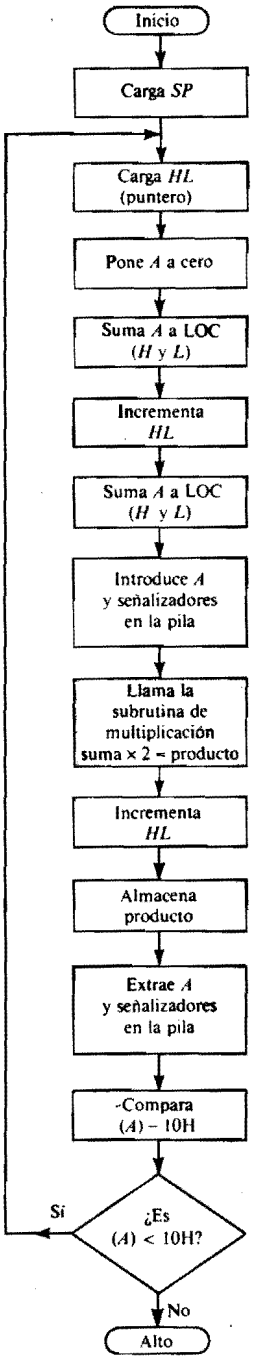
6.13. UTILIZACION DE SUBRUTINAS

Una *subrutina* es un subprograma que puede ser utilizado muchas veces en el programa principal o es, normalmente, utilizado por un programador en muchos programas. Una subrutina se utiliza a veces, con tanta frecuencia, que se coloca permanentemente en una ROM. La transferencia de control desde el programa principal a la subrutina se realiza por una instrucción de *llamada (call)*. Lo inverso, o transferencia de control de la subrutina al programa principal, lo realiza una instrucción de *vuelta (return)*. Las instrucciones *push, pop* y *load stack pointer (introducir, sacar* y *cargar puntero de pila)* también se utilizan normalmente en las subrutinas.

Un diagrama de flujo funcional para un problema ejemplo se muestra en la Figura 6.52*a*. La tarea del microprocesador es introducir dos muestras en forma de números binarios, sumarlas y guardar la suma. Los tres primeros rectángulos en la parte superior del diagrama de flujo sugieren este procedimiento; a continuación, la suma es multiplicada por un número denominado factor de escalamiento, el resultado de la multiplicación, o producto, se almacena entonces. La suma que se obtuvo en el tercer rectángulo se restaura ahora en el registro *A* de la MPU. Esta suma se examina en el bloque de decisión, en forma de diamante. Si la suma es igual o mayor que 10H (16_{10}), el programa sale al modo de alarma. Si la suma es menor que 10H, el programa vuelve al ciclo y entran nuevas muestras. Este tipo de programa se puede utilizar en instrumentación industrial donde ocurre lo siguiente:



(a) Diagrama de flujo funcional para el problema de instrumentación



(b) Diagrama de flujo detallado para el problema de instrumentación muestra-suma-multiplica-almacena-y-realiza un nuevo ciclo

Figura 6.52.

1. La suma de dos entradas debe ser vigilada continuamente.
2. La suma debe ser escalada al valor más apropiado y se almacena este valor.
3. La suma de dos entradas debe ser comprobada como mínimo cada segundo para determinar si es peligrosamente grande, y si es así, el programa debe salir al modo de alarma.

El programa se diseña para que esté en un ciclo continuamente, tomando nuevas muestras cada ciclo. Del ciclo se sale al modo de alarma sólo si existe algún problema. Un operador u otro programa deberá entonces manipular el procedimiento de alarma simbolizado en la Figura 6.52*a* por el símbolo conector circular.

Un diagrama de flujo detallado para el mismo problema se dibuja en la Figura 6.52*b*, donde cada rectángulo es equivalente a una operación del microprocesador. Sin embargo, el bloque sombreado representa un *grupo* de instrucciones denominado subrutina MULTIPLY.

A partir del diagrama de flujo detallado de la Figura 6.52*b*, se escribe un programa en lenguaje ensamblador. Cada bloque del diagrama de flujo corresponde a una sentencia del lenguaje ensamblador. En la Figura 6.53 se da un listado en lenguaje ensamblador para el programa principal, donde se utilizan solamente operaciones del repertorio de instrucciones del microprocesador genérico; el listado de la subrutina real para la operación de multiplicar no se incluye en la Figura 6.53.

Las tres primeras sentencias del lenguaje ensamblador de la Figura 6.53 inicializan el puntero de pila a 20C0H, el registro par *HL* a 2040H y el acumulador (*A*) a 00H. La cuarta sentencia suma el contenido de *A* (actualmente 00H) al de la posición de memoria señalada por el registro *HL* (dirección 2040H). Acudir a la Figura 6.54, que muestra el programa principal, datos, programa de subrutina y memoria de pila utilizada en este ejemplo. En esta figura se observa que el contenido sumado a *A* desde la posición 2040H es 05H. Al final de la cuarta instrucción, el contenido de *A* es 05H (00 + 05 = 05).

La quinta instrucción del programa principal de la Figura 6.53 incrementa el registro par *HL*, la sexta instrucción suma la posición 2041H de memoria de *A*; en la Figura 6.54 se determina que la posición 2041H contiene 09H, mientras que *A* contiene 05H. La suma de *A* después de la segunda instrucción ADD *M* es 0EH. Esta es la suma de las dos muestras.

La siguiente instrucción del programa principal de la Figura 6.53 es la operación PUSH *PSW*, que guarda el contenido del acumulador y señalizadores. Esto debe hacerse porque la siguiente instrucción de llamada destruirá el contenido de estos registros cuando se ejecute la subrutina de la multiplicación. La operación de llamada almacena la dirección de la siguiente instrucción del programa principal (dirección 200EH en este ejemplo) en la pila y después salta a la dirección de la primera instrucción de la subrutina (dirección 2050H). La llamada a la subrutina está esbozada en la Figura 6.54. Considerar temporalmente la instrucción de llamada como una sencilla operación de multiplicar, que multiplica el contenido de *A* por 2; la subrutina de este ejemplo entonces realiza esta multiplicación: 0E × 02 = 1CH, y el producto en el acumulador después de la vuelta de la subrutina es 1CH.

La siguiente instrucción del programa principal de la Figura 6.53 después de volver de la subrutina, está en la dirección 200EH, es la instrucción INX *H* que incrementa el registro para *HL*. El producto de la subrutina de multiplicación es ahora almacenado en la posición de memoria 2042H por la instrucción MOV *M,A*.

Recordar que las instrucciones de introducir y sacar (push y pop) se utilizan en parejas. Antes, en el programa principal el contenido del acumulador y de los señalizadores fue introducido en la pila por la operación PUSH *PSW*. Ahora estos contenidos son restaurados por la instrucción POP *PSW*. Es obvio que el contenido del acumulador ahora es igual a la suma (0EH en este ejemplo) en vez de al producto. La suma se examina con la siguiente instrucción.

Etiqueta	Nemotécnico	Operando	Comentarios
	LXI	*SP*,20C0H	; Carga puntero de pila con 20C0H
START	LXI	*H*,2040H	; Carga registro *HL* con 2040H (*HL* utilizado como puntero de dirección en el programa principal) 2040=5
	XRA	*A*	; Pone el acumulador a 00H
	ADD	*M*	; Suma (*A*) a la posición de memoria 2040H (el primer número que será sumado está en 2040H) =5
	INX	*H*	; Incrementa el registro *HL* a 2041H 2041=9
	ADD	*M*	; Suma (*A*) a la posición de memoria 2041H (el segundo número que será sumado está en 2041H) =E
	PUSH	*PSW*	; Introduce (*A*) y señalizadores en la pila
	CALL	MULTIPLY	; Llama a la subrutina de multiplicación a la posición de memoria 2050H
	INX	*H*	; Incrementa el registro *HL* a 2042H
	MOV	*M,A*	; Almacena el producto en la posición de memoria 2042H
	POP	*PSW*	; Extrae de la pila y restaura (*A*) y señalizadores
	CPI	10H	; Compara (*A*) con 10H [(*A*) −10H, si (*A*) < 10H, entonces *CY* = 1; de otra forma *CY* = 0]
	JC	START	; Salta a la posición de START (dirección 2003H) si *CY* = 1; de otra forma continúa con la siguiente instrucción
	HLT		; Detiene la MPU

Figura 6.53. Versión en lenguaje ensamblador del programa principal de instrumentación.

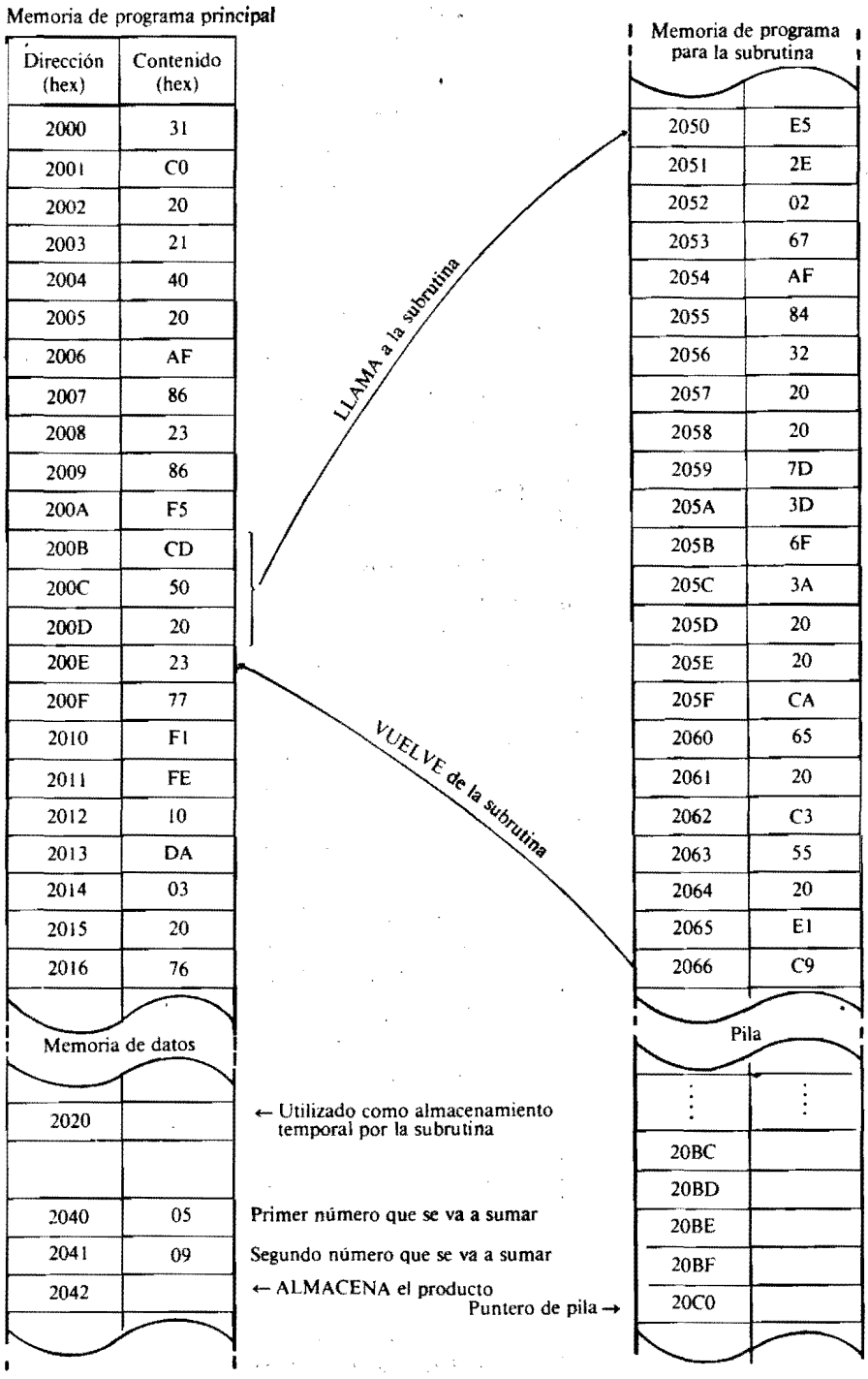


Figura 6.54. Contenidos de memoria y posiciones de almacenamiento del problema de instrumentación.

Es importante observar que la suma (0EH en este ejemplo) pasa a la subrutina para su tratamiento, sin embargo, es el resultado de la operación de multiplicar lo que sale de la subrutina y posteriormente se almacena. En la jerga de computadoras, el paso de datos a y desde subrutinas se denomina *paso de parámetros*. En este caso se utilizó el registro *A*, sin embargo, también puede utilizarse cualquier área de memoria.

Las dos siguientes instrucciones del programa principal de la Figura 6.53 son para probar la suma en el acumulador. La instrucción «compara inmediato» realiza la resta interna en la CPU: 0EH − 10H = FEH, que es la representación en complemento a 2 de -2_{10}. El contenido del acumulador es menor que 10H, así que el señalizador de arrastre (*CY*) se pone a 1. La instrucción «salta si arrastre» examina el señalizador *CY*. Al estar a 1, la MPU salta a una dirección simbólica llamada START, donde comienza el ciclo con la instrucción LXI *H* (dirección 2003H de la Figura 6.54).

Pueden utilizarse diversos métodos para resolver el problema de multiplicación binaria utilizando sólo las operaciones del repertorio de instrucciones del microprocesador genérico. El método más sencillo utiliza la suma repetida para resolver un problema de multiplicación, por ejemplo, la solución del problema $5 \times 3 = 15_{10}$ se ilustra en la Figura 6.55*a*, sin embargo, la Figura 6.55*b* muestra otro método para resolver el mismo problema: el *de la suma repetida*. Aquí el multiplicando se suma repetidamente, el número de veces que indica el multiplicador, el resultado de esta suma repetida es por tanto el producto. La multiplicación que realiza la subrutina de multiplicación está resumida en la Figura 6.55*c*. El multiplicando en el ejemplo es 0EH y el multiplicador es 02H. El multiplicando se suma dos veces, dando 1CH, que se considera el producto.

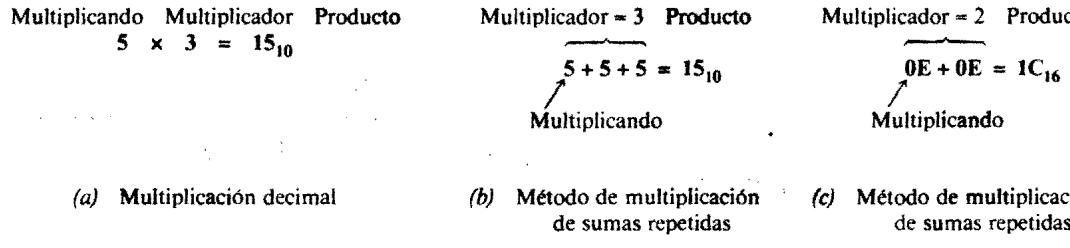


Figura 6.55.

En la Figura 6.56 se muestra un diagrama de flujo detallado para la subrutina de la multiplicación. Cada bloque del diagrama corresponde a una sentencia del lenguaje ensamblador en el programa de la subrutina de la Figura 6.57.

El primer bloque del diagrama de flujo de la Figura 6.56 guarda el contenido actual del registro par *HL* utilizando una operación de introducir (push). Es común en la mayoría de las subrutinas introducir el contenido de los registros de la MPU en la pila debido a que éstos pueden ser utilizados durante la ejecución de la subrutina y por tanto cambiar los valores que tenían. Por supuesto, el contenido del registro par *HL* será restaurado al final de la subrutina con una operación de sacar (pop) (ver la instrucción pop cercana al extremo inferior de este ejemplo).

Los tres siguientes bloques de la Figura 6.56 inicializan los registros *L*, *H* y *A*. El registro *L* contendrá el multiplicador (02H en este ejemplo), y será decrementado a 00H durante la rutina

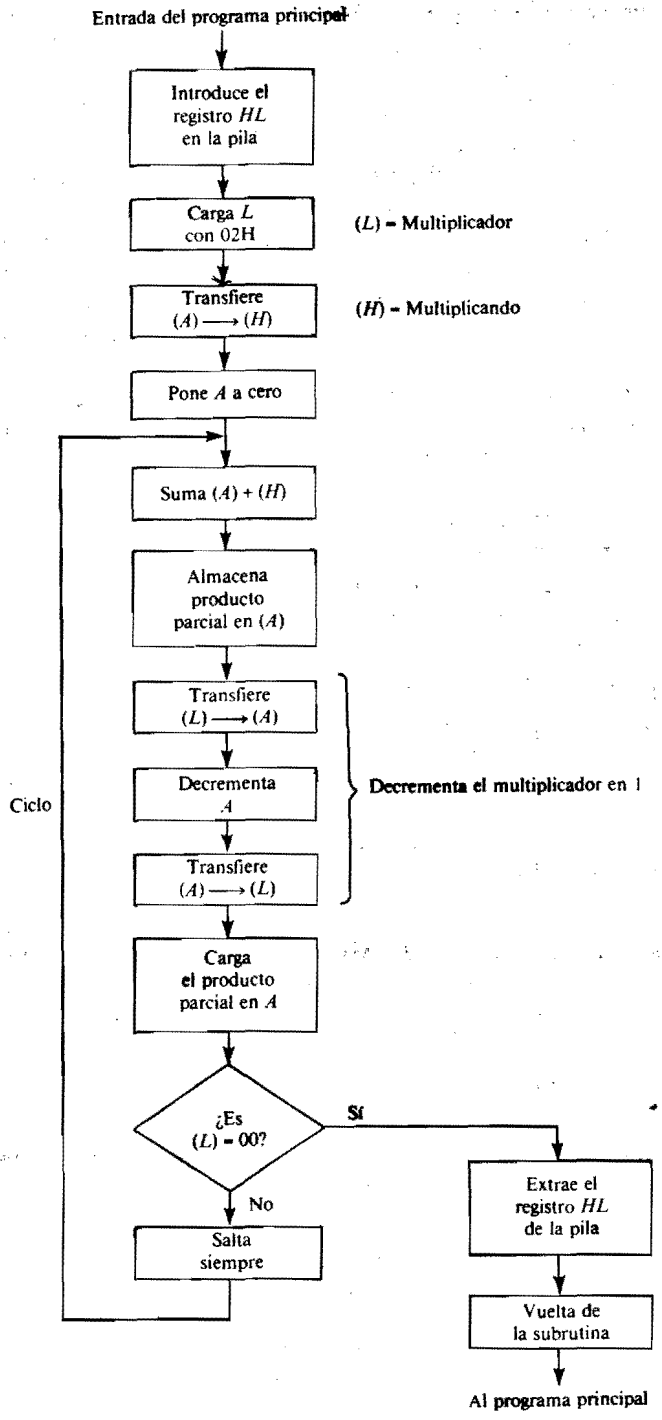


Figura 6.56. Diagrama de flujo detallado de la subrutina de multiplicación utilizada en el problema de instrumentación.

Etiqueta	Nemotécnico	Operando	Comentarios
	PUSH	*H*	; Introduce los registros (*H*) y (*L*) en la pila para guardar el contenido del registro *HL*
	MVI	*L*,02H	; Carga el registro *L* con 02H (este es el factor de escala); 02H es el *multiplicador*
	MOV	*H,A*	; Transfiere (*A*) → (*H*); contenido de *H* = *multiplicando;* guarda la suma en *H*
	XRA	*A*	; Pone *A* a cero
LOOP	ADD	*H*	; Suma (*H*) + (*A*); almacena la suma en *A*
	STA	2020H	; Almacena (*A*) en la posición de memoria 2020H (2020H se utiliza como almacenamiento temporal)
	MOV	*A,L*	; Transfiere (*L*) a *A*
	DCR	*A*	; Decrementa el registro *A*
	MOV	*L,A*	; Transfiere (*A*) a *L*
	LDA	2020H	; Carga el contenido de la posición de memoria 2020H en *A* (restaura el contenido de *A* de la memoria temporal 2020H)
	JZ	DONE	; Salta a la posición DONE (dirección 2065H) si *Z* = 1 o si (*L*) es decrementado a 00H; de otra forma, continúa con la siguiente instrucción
	JMP	LOOP	; Salta siempre a la posición LOOP (dirección 2055H)
DONE	POP	*H*	; Extrae de la pila y restaura el contenido del registro *HL*
	RET		; Vuelve de la subrutina

Figura 6.57. Versión en lenguaje ensamblador de la subrutina de multiplicación utilizada en el problema de instrumentación.

de multiplicación. El registro *H* contendrá el multiplicando (0EH en este ejemplo). Esta fue la suma pasada a la subrutina desde el programa principal, en el acumulador, y después transferida al registro *H*. El acumulador (*A*) entonces se pone a 00H.

El quinto bloque de la Figura 6.56 representa la suma del multiplicando (0EH en este ejemplo) con *A*. El producto parcial de 0EH se almacena entonces temporalmente en la posición de memoria 2020H, mientras que el acumulador se utiliza para decrementar el contenido del registro *L*. Después de que el multiplicador es decrementado de 02H a 01H, éste se vuelve a colocar en el registro *L*. A continuación el producto parcial se restaura en el acumulador.

El símbolo de decisión de la Figura 6.56 hace la pregunta: ¿Es (*L*) = 00? Si la respuesta es no, el programa vuelve a saltar al ciclo. Sin embargo, si la respuesta a la pregunta es sí, el programa bifurca a la derecha y el contenido del registro par *HL* es restaurado por la operación de sacar. Finalmente, la operación de vuelta bifurca de nuevo al programa principal. De acuerdo con el diagrama de la Figura 6.54, la vuelta de la subrutina se hace a la dirección 200EH del programa principal. Como el multiplicador en este ejemplo es 02H, el programa pasará dos veces a través de la secuencia suma-almacena-transfiere-decrementa-transfiere-carga y después sale al programa principal. Observar que el producto final 1CH pasará al programa principal al estar en el acumulador.

Un listado en lenguaje ensamblador para la subrutina de multiplicación se detalla en la Figura 6.57. Cada sentencia del lenguaje ensamblador corresponde a un bloque del diagrama de flujo detallado de la Figura 6.56. Intentar seguir el flujo del programa de la Figura 6.57, acudiendo al diagrama de flujo como ayuda.

En resumen, cuando se utilizan instrucciones de llamada y vuelta, asegurarse que estén emparejadas. Observar también que los registros adecuados están inicializados y utilizar las instrucciones de introducir y sacar a pares. Tener en cuenta el número de parámetros (datos) que son pasados a y desde la subrutina.

Las subrutinas son muy utilizadas por los programadores. Las ventajas de utilizar subrutinas son:

1. Añaden una estructura al programa que lo hace más fácil de comprender.
2. Hacen la depuración más fácil.
3. Abrevian el programa.
4. Permiten escribir grandes programas de una forma más fácil y eficiente.

La desventaja de utilizar subrutinas es que el programa se ejecutará generalmente de forma más lenta cuando se llama a las subrutinas.

PROBLEMAS RESUELTOS

6.99. El diagrama de flujo de la subrutina de la multiplicación de la Figura 6.56 utiliza el método de _____ (suma y desplaza, adición repetida) para calcular un producto.

Solución:

El diagrama de flujo de la subrutina de la multiplicación de la Figura 6.56 utiliza el método de la suma repetida para calcular un producto. El método de suma y desplazamiento es otro método más común de multiplicar que utiliza una serie de sumas y desplazamientos para calcular un producto.

6.100. Acudir a la Figura 6.54. ¿Qué tipos de instrucciones del microprocesador utilizan la posición de memoria llamada pila?

Solución:

Llamada, vuelta, introducir y sacar utilizan el área de memoria LIFO llamada pila en la Figura 6.54.

6.101. El propósito de la instrucción LXI *SP* de la Figura 6.53 es _____ (inicializar, examinar) el puntero de pila.

Solución:

El propósito de la instrucción LXI *SP* en la Figura 6.53 es inicializar el puntero de pila.

6.102. Si el señalizador de arrastre se pone a 1 en la Figura 6.53, la operación «JC» hará que la MPU busque la siguiente instrucción _____ (HLT, LXI *H*).

Solución:

Si el señalizador de arrastre se pone a 1 en la Figura 6.53, la operación «JC» hará que la MPU busque la siguiente instrucción LXI *H* en la dirección simbólica START.

6.103. La instrucción de llamada en la Figura 6.53 hace que el contenido del _____ (registro *A*, contador de programa) sea introducido en la pila y el contador de programa se ponga a _____ (dirección hex).

Solución:

La instrucción de llamada de la Figura 6.53 hace que el contenido del contador de programa sea introducido en la pila y el contador de programa se ponga a 2050H. La subrutina de multiplicar se localiza en la dirección 2050H como muestra la Figura 6.54.

6.104. La instrucción POP *PSW* de la Figura 6.53 saca el contenido del _____ (acumulador, contador de programa) y de los señalizadores de la pila y los devuelve a la CPU.

Solución:

La instrucción POP *PSW* de la Figura 6.53 saca el contenido del acumulador y de los señalizadores de la pila y los devuelve a la CPU. Ver la tabla de instrucciones misceláneas (Fig. 6.40).

6.105. Un símbolo de decisión en forma de diamante como el de la Figura 6.56 normalmente es sustituido por una instrucción de _____ (comparación, salto) cuando se escribe el programa.

Solución:

Un símbolo de decisión en forma de diamante como el de la Figura 6.56 normalmente es sustituido por una instrucción de salto cuando se escribe el programa. El programa en lenguaje ensamblador de la Figura 6.57 muestra que en este caso se utiliza la instrucción «JC» o «salta si cero».

6.106. En la subrutina de multiplicación de la Figura 6.57, el valor del multiplicando está en el registro _____ (*H*, *L*) mientras que el valor del multiplicador está en el registro _____ (*H*, *L*).

Solución:

En la subrutina de multiplicación de la Figura 6.57, el valor del multiplicando está en el registro *H* mientras que el del multiplicador está en el registro *L*.

6.107. En la subrutina de multiplicación, el contenido del registro _____ (*H, L*) es decrementado hasta 00H.

Solución:

En la subrutina de multiplicación, el contenido del registro *L* es decrementado hasta 00H.

6.108. Acudir a las Figuras 6.53 y 6.57. Los registros *H* y *L* son utilizados como _____ _____ (punteros de dirección, registros de datos) en el programa principal y como _____ _____ (punteros de dirección, registros de datos) en la subrutina.

Solución:

Según la información de las Figuras 6.53 y 6.57, los registros *H* y *L* son utilizados como punteros de dirección en el programa principal y como registros de datos en la subrutina de multiplicación.

PROBLEMAS SUPLEMENTARIOS

6.109. Un listado en lenguaje _____ (ensamblador, máquina) se caracteriza por el uso de operandos, códigos de op y direcciones de memoria hexadecimales.

Res. máquina.

6.110. Una sentencia de un programa en lenguaje ensamblador se divide en cuatro _____ (campos, segmentos) denominados etiqueta, nemotécnico, operando y comentario.

Res. campos.

6.111. Una versión en lenguaje ensamblador, a veces, se denomina programa _____ (objeto, fuente).

Res. fuente.

6.112. BASIC es considerado un lenguaje de alto nivel porque cada sentencia del programa representa _____ (menos, más) operaciones máquina del microprocesador que una sentencia en lenguaje ensamblador.

Res. más.

6.113. Los señalizadores de la Figura 6.3 son parte del dispositivo de almacenamiento de la CPU denominado registro _____ .

Res. de status.

6.114. Cuando los registros *H* y *L* se unen para formar el registro *HL*, habitualmente son utilizados para señalar _____ .

Res. direcciones.

6.115. La pila en el sistema microprocesador genérico (o en un sistema 8080/8085) está localizada en _____ .

Res. la RAM.

6.116. El registro de 16 bits etiquetado *SP* en la Figura 6.3 es el _____ _____ .

Res. puntero de pila.

6.117. El registro de 16 bits etiquetado *PC* en la Figura 6.3 es el _____.

Res. contador de programa.

6.118. Como regla general, las operaciones aritméticas del microprocesador _____ (afectan, no afectan) a los señalizadores.

Res. afectan.

6.119. Durante la instrucción comparar de un microprocesador, solamente el contenido del registro _____ (*A*, de status) es afectado por la operación.

Res. status (son afectados los señalizadores).

6.120. Acudir a la Figura 6.58. El contenido del acumulador después de la operación ADD *L* será _____ (8 bits).

Res. 00000001.

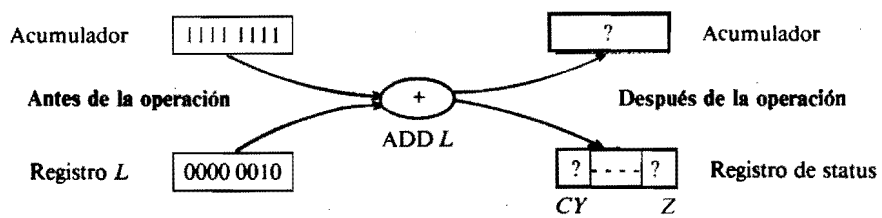


Figura 6.58.

6.121. Después de la operación ADD *L* de la Figura 6.58 el señalizador de arrastre estará a _____ (0, 1) mientras que el señalizador de cero es igual a _____ (0, 1).

Res. 1, $Z = 0$.

6.122. Internamente el microprocesador no tiene hardware para restar; por tanto realiza restas convirtiendo el _____ (minuendo, sustraendo) a su forma en complemento a 2 y sumando.

Res. sustraendo.

6.123. Al ejecutar una instrucción de resta de la tabla de la Figura 6.4, un señalizador de arrastre en reinicialización ($CY = 0$) significa _____ (préstamo, no préstamo) o que el minuendo es _____ (mayor, menor) que el sustraendo.

Res. no préstamo, mayor.

6.124. Acudir a la Figura 6.59. El contenido del acumulador después de la operación SUB *H* será _____ (8 bits), que es la representación en complemento a 2 de _____ (decimal).

Res. 11111110, -2_{10}.

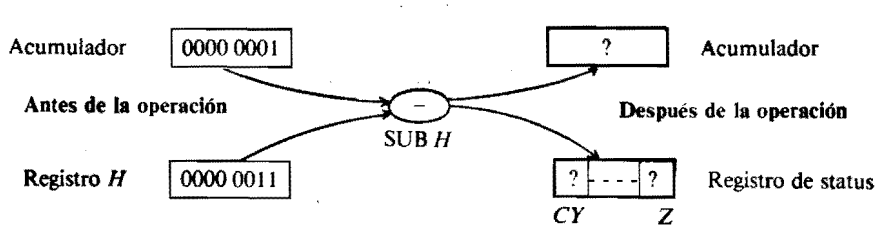


Figura 6.59.

6.125. Después de la operación SUB *H* de la Figura 6.59, el señalizador de arrastre («préstamo») estará a _____ (0, 1) mientras que el señalizador de cero será igual a _____ (0, 1).

Res. 1, $Z = 0$.

6.126. Como regla general, los señalizadores del registro de status _____ (son, no son) afectados por las instrucciones lógicas del microprocesador.

Res. son.

6.127. El símbolo (·) utilizado en la tabla de instrucciones lógicas de la Figura 6.14 denota la operación _____ (AND, multiplicación).

Res. AND.

6.128. Acudir a la Figura 6.14. Si el acumulador de la MPU = 11101110 y el señalizador de arrastre = 0, ¿cuáles son los contenidos del acumulador y señalizador de arrastre después de una instrucción «RAR»?

Res. acumulador = 01110111, señalizador de arrastre = 0.

6.129. Casi todas las instrucciones de transferencia de datos _____ (afectan, no afectan) a los señalizadores del registro de status del microprocesador.

Res. no afectan.

6.130. Acudir a la Figura 6.21. Cuando se utiliza una instrucción «MOV *L,A*», el registro _____ (*A, L*) es la fuente del dato, mientras que el registro _____ (*A, L*) es el destino.

Res. A, L.

6.131. Acudir a la Figura 6.21. Las instrucciones IN y OUT utilizan direccionamiento _____ (directo, inherente).

Res. directo.

6.132. Acudir a la Figura 6.21. La instrucción «LDA» utilizada en la Figura 6.60 tiene un código de op de _____ (hex) y utiliza una instrucción de _____ (número) bytes.

Res. 3AH, 3.

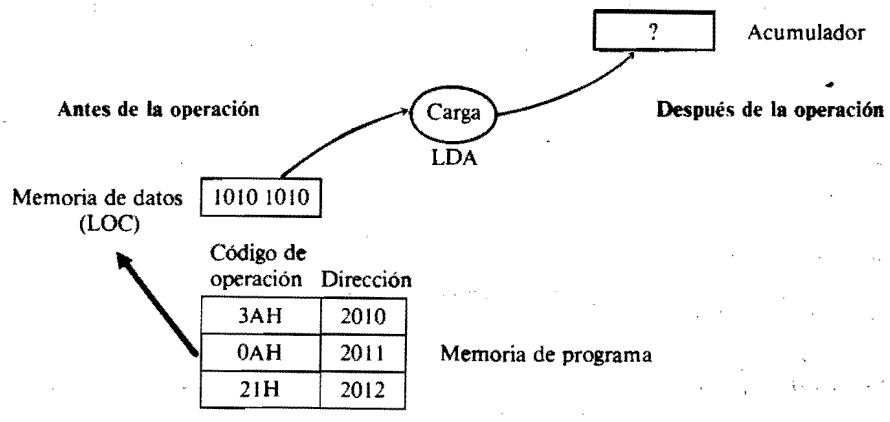


Figura 6.60.

6.133. El contenido del acumulador después de la operación de carga de la Figura 6.60 es _____ (8 bits).
Res. 10101010.

6.134. La fuente del dato de la Figura 6.60 era la posición _____ (hex) de la memoria de datos.
Res. 210AH.

6.135. Acudir a la Figura 6.21. La operación de carga mostrada en la Figura 6.60 utiliza direccionamiento _____ (directo, inmediato).
Res. directo.

6.136. Las instrucciones de salto también se denominan de transferencia de control o de _____.
Res. bifurcación.

6.137. Se dice que las operaciones de salto _____ (condicional, incondicional) son instrucciones de toma de decisión.
Res. condicional.

6.138. Las operaciones de salto afectan al contenido del _____ _____ (contador de programa, puntero de pila).
Res. contador de programa.

6.139. Acudir a la Figura 6.30. El código de op para la instrucción JZ usada en la Figura 6.61 es _____ (hex).
Res. CAH.

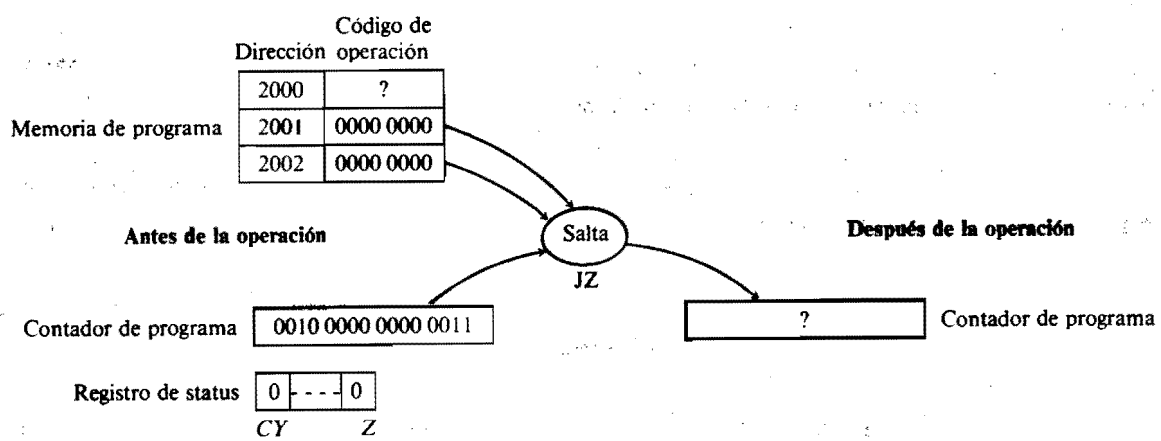


Figura 6.61.

6.140. El contenido del contador de programa después de la operación en la Figura 6.61 será _____ (hex).
Res. 2003H (igual que antes de la operación porque no se realiza ningún salto).

6.141. La instrucción _____ (de llamada, de retorno) del microprocesador genérico realiza la función de una instrucción de introducir y saltar (push y jump).
Res. llamar.

6.142. La operación de _____ (llamar, vuelta) es la última instrucción de una subrutina.

Res. vuelta.

6.143. Acudir a la Figura 6.62. El contenido de la posición de la pila 2109H después de la operación de llamada será _____ (8 bits).

Res. 00100000 (20H).

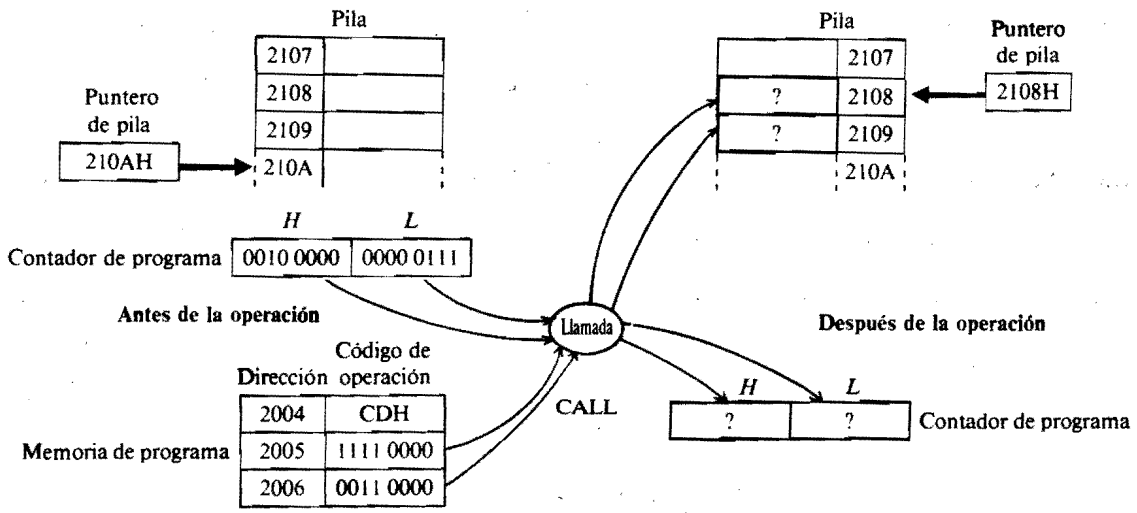


Figura 6.62.

6.144. Acudir a la Figura 6.62. El contenido de la posición 2108H de la pila después de la operación de llamar será _____ (8 bits).

Res. 00000111 (07H).

6.145. Acudir a la Figura 6.62. El contenido del contador de programa después de la operación de llamada será _____ (hex).

Res. 30F0H (0011 0000 1111 0000 en binario).

6.146. Acudir a la Figura 6.62. El contenido del contador de programa después de la operación de llamar (call) apunta a la primera instrucción de la _____ (pila, subrutina).

Res. subrutina.

6.147. Acudir a la Figura 6.40. Si la instrucción «PUSH *PSW*» se hubiese utilizado como primera instrucción de la subrutina, la instrucción _____ (nemotécnico) se utilizaría al final, justo antes de la vuelta.

Res. POP *PSW*.

6.148. Cuando el microprocesador genérico haya sido colocado en el modo de alto por la instrucción «HLT», solamente una petición de interrupción o una entrada de _____ (reloj, reinicialización) a la MPU puede reinicializar la unidad.

Res. reinicialización.

6.149. El primer paso al escribir un programa es _____ (definir el problema, escribir la versión del programa en lenguaje ensamblador).

Res. definir el problema.

6.150. El programador de un sistema microprocesador debe familiarizarse con la arquitectura del sistema, el mapa de memoria, los registros de la CPU y el _____ _____ del microprocesador.

Res. repertorio de instrucciones.

6.151. Después de definir y analizar el problema, el siguiente paso en programación es para dibujar _____ (la documentación, el diagrama de flujo) que muestra la solución del problema.

Res. diagrama de flujo.

6.152. Las operaciones del microprocesador genérico que utilizan modo de direccionamiento inmediato son especificadas por instrucciones de _____ (número) bytes.

Res. 2 ó 3.

6.153. Acudir a la Figura 6.40. La operación «PUSH *H*» es una instrucción de _____ (número) bytes que utiliza el modo de direccionamiento _____.

Res. 1, registro indirecto.

6.154. Acudir a la Figura 6.4. La operación «compara *A* con *L*» es una instrucción de _____ (número) bytes que usa el modo de direccionamiento _____.

Res. 1, registro.

6.155. Acudir a la Figura 6.21. La operación «LDA» es una instrucción de _____ (número) bytes que utiliza el modo de direccionamiento _____.

Res. 3, directo.

6.156. _____ (Se hace una pregunta, Se hace una sentencia de hecho) dentro de un símbolo de decisión en un diagrama de flujo de programación.

Res. Se hace una pregunta.

6.157. Se utiliza una instrucción _____ (de bifurcación condicional, lógica) para realizar la operación representada por el símbolo de decisión de un diagrama de flujo.

Res. de bifurcación condicional.

6.158. Muchas veces instrucciones de _____ (comparación, carga) son utilizadas en un programa justo antes de una operación de bifurcación condicional.

Res. comparación.

6.159. Acudir a la Figura 6.51*b*. ¿Cuál es el propósito de la operación «XRA *A*» en este programa?

Res. poner el acumulador a 00H, inicializando así el registro *A*.

6.160. La técnica de utilizar _____ en un programa se utiliza cuando deben realizarse tareas repetitivas por el microprocesador.

Res. ciclos.

6.161. Acudir a la Figura 6.50. La salida del número más alto en este programa será el _____ (hex) o _____ (decimal).

Res. FEH, 254_{10}.

6.162. Las subrutinas del microprocesador, utilizadas normalmente, son almacenadas a veces, permanentemente, en _____ (RAM, ROM).

Res. ROM.

6.163. La instrucción «JC» examina el señalizador _____ (*CY, Z*) en el microprocesador genérico.

Res. *CY* (señalizador de arrastre).

6.164. Las instrucciones de introducir (push) y _____ (sumar, sacar) deben ser utilizadas en parejas.

Res. pop (sacar).

6.165. Las instrucciones de llamar y _____ (brincar, de vuelta) deben ser utilizadas en parejas.

Res. de vuelta.

Capítulo 7

INTERFACES CON EL MICROPROCESADOR

7.1. INTRODUCCION

La mayoría de los microprocesadores tienen poco valor funcional por sí mismos. Muchos no contienen una memoria sustancial, y pocos tienen puertos de entrada y salida que los conecten directamente a dispositivos periféricos. Los microprocesadores operan como parte de un sistema. La interconexión, o enlace, de las partes en este sistema se denomina *interfaz*. Generalmente, una interfaz es una frontera entre dos o más dispositivos que comparten información; otras consideraciones relativas a las interfaces incluyen la sincronización, dirección de la transmisión de datos y, a veces, el ajuste de niveles o modos de señal.

Considerar el sistema simplificado basado en microprocesador esbozado en la Figura 7.1. La interfaz, o enlace, entre cada uno de los dispositivos mostrados y la unidad microprocesadora serán examinados. Por tanto, en este capítulo se cubrirá la interfaz de la MPU con la ROM y la RAM así como las interfaces entre la MPU y los dispositivos de entrada/salida (E/S). Se dará información poco detallada sobre dispositivos periféricos específicos. Es obvio, según la Figura 7.1, que los buses de direcciones, datos y control juegan un papel importante para enlazar todas las partes del sistema. Las interfaces se encargan de la sincronización y transmisión de datos a y desde la MPU, y por tanto debe estudiarse el software como el hardware.

Generalmente, las transferencias de entrada y salida de datos de la MPU a los buses corresponden a una de estas actividades:

1. Lectura de memoria.
2. Escritura en memoria.
3. Lectura de E/S.
4. Escritura en E/S.
5. Manipulación de interrupciones o reinicialización.

Cuando se dice que el dato es introducido desde otro dispositivo, significa *entrada en relación con la MPU*. De igual forma, una salida será una salida de la unidad microprocesadora. Generalmente, el microprocesador es el foco de todas las operaciones. Sin embargo, algunos controles liberan las MPU de los buses de dirección y datos durante el tiempo que un dispositivo periférico puede acceder directamente a la memoria principal del sistema sin ir a través de la MPU. Esto se denomina *acceso directo de memoria* (DMA).

El diagrama de la Figura 7.1 sugiere que la MPU, ROM, RAM, adaptador de interfaces de entrada y adaptador de interfaces de salida son dispositivos separados. Esto puede ser o no ser cierto dependiendo del sistema específico. Es bastante común para los fabricantes producir adap-

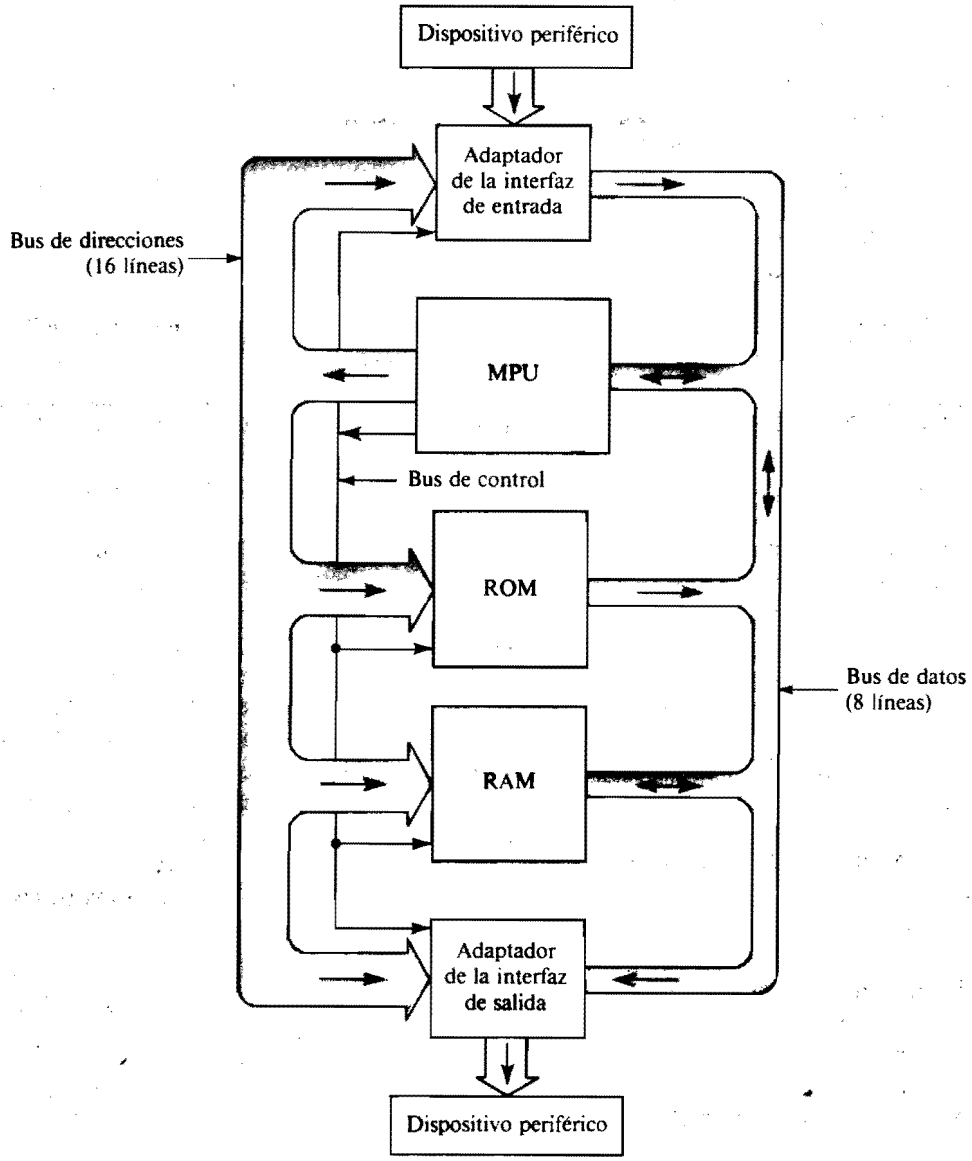


Figura 7.1. Diagrama de bloques de un sistema basado en microprocesador.

tadores de interfaces para periféricos, compatibles con sus unidades microprocesadoras. Estos adaptadores, habitualmente, son de propósito general ya que normalmente pueden ser programados para que funcionen bien como interfaz de entrada o de salida. Algunos fabricantes integran RAM y puertas de E/S o ROM y puertas de E/S en el mismo CI para disminuir el número de componentes del sistema. Los fabricantes también producen componentes especializados de interfaz en forma de CI, entre los que se incluyen interfaces de comunicación programables,

controladores DMA programables, controladores de interrupciones programables, controladores de disquetes, controladores de enlace de datos síncronos, controladores CRT y controladores de pantalla/teclado.

7.2. INTERFAZ CON LA ROM

Considerar la interfaz del microprocesador con una memoria de sólo lectura (ROM) o memoria de sólo lectura programable (PROM). La MPU y sección de ROM de un sistema están representadas en la Figura 7.2. Las 8 líneas del bus de datos están conectadas a los terminales de salida (O_0–O_7) de la ROM, la salida de lectura ($\overline{RD}$) va de la MPU a la entrada de habilitación de salida ($\overline{OE}$) de la ROM.

Los 12 bits menos significativos del bus de direcciones (A_0–A_{11}) están conectados a la ROM de 4K × 8. El decodificador permanente del CI ROM puede acceder cualquiera de 4096 ($2^{12} = 4096$) palabras de memoria de sólo lectura de 8 bits utilizando las 12 entradas de dirección. Las 4 líneas de dirección más significativas (A_{12}–A_{15}) van a un elemento combinacional denominado decodificador de direcciones. Para acceder y leer datos de la ROM, la MPU debe:

1. Inicializar las direcciones en las líneas de dirección A_0 a A_{11}.
2. Poner $\overline{OE}$ en BAJA utilizando la línea de control de lectura.
3. Poner $\overline{CS}$ en BAJA utilizando el decodificador de direcciones y la línea de selección de pastilla.

Suponer que la MPU quiere acceder a la posición de memoria 0000H ($0000\ 0000\ 0000\ 0000_2$). Los 12 bits menos significativos se aplican directamente a la circuitería de decodificación de la

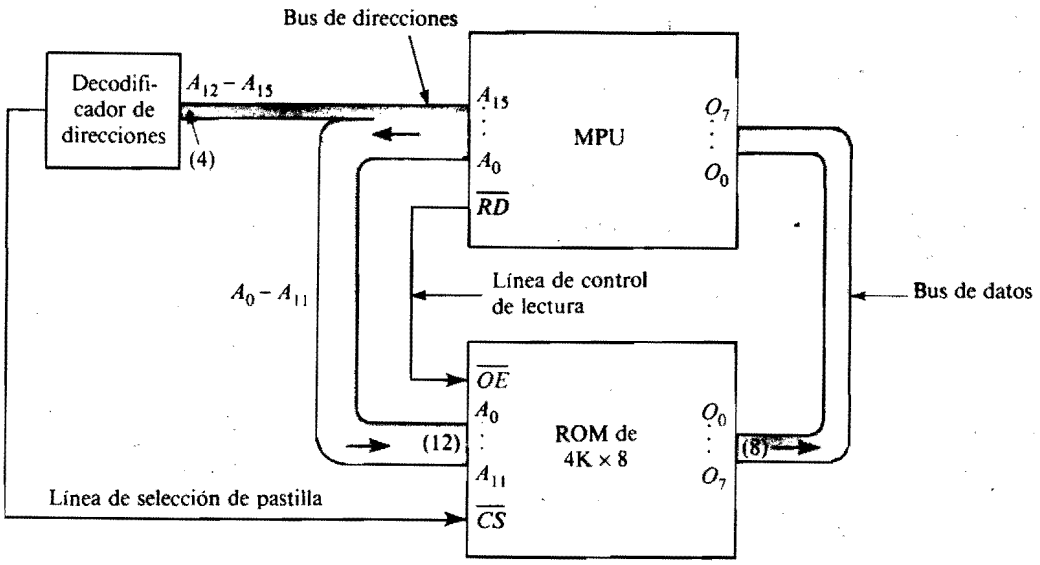


Figura 7.2. Interfaz de la ROM con el microprocesador.

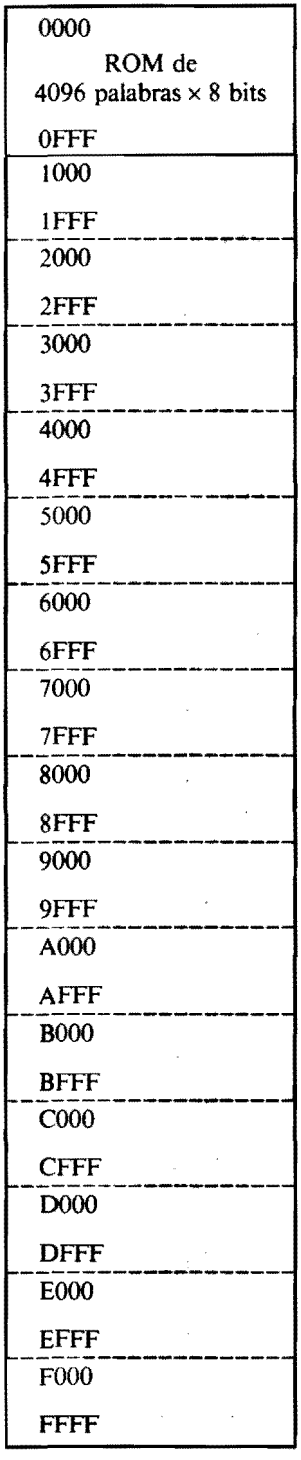


Figura 7.3. Mapa de memoria.

ROM, vía las líneas de dirección A_0 a A_{11}. Los 4 bits más significativos (A_{12}–A_{15}) también forman parte de la dirección, y son decodificados por un *decodificador de direcciones*. Si A_{12} a $A_{15} = 0000_2$, entonces el decodificador de direcciones producirá un nivel BAJO que habilitará la entrada de selección de pastilla ($\overline{CS}$) de la ROM de la Figura 7.2.

El mapa de memoria dibujado en la Figura 7.3 puede ayudar a explicar el trabajo realizado por el decodificador de direcciones. El mapa de memoria simboliza una unidad de 64K (realmente 65.536) posiciones de memoria y, está dividido en dieciséis segmentos de 4K. La tarea del decodificador de direcciones es ayudar a la MPU para que cada vez acceda *solamente uno* de los segmentos de 4K. Si las 4 entradas al decodificador de direcciones fuesen 0000, entonces se accedería al segmento 0 (posiciones de memoria 0000-0FFFH). Sin embargo, si las entradas al decodificador de direcciones fuesen 0001, entonces se accedería al segmento 1 (posiciones de memoria 1000-1FFFH), etc. En resumen, las 4 líneas de dirección más significativas seleccionan un segmento de memoria mientras que los 12 bits menos significativos determinan la posición específica de memoria en ese segmento de 4K.

Cuando se hace la interfaz de la MPU con la ROM, las consideraciones importantes son *direccionamiento* y *temporización*. El direccionamiento se ha explicado ya. El diagrama de la Figura 7.4 muestra las señales de la MPU que afectan la lectura de una palabra de memoria de 8 bits de la ROM. La línea superior del diagrama de tiempos muestra las líneas de dirección (A_0–A_{15}) cambiando a sus respectivos niveles lógicos. De acuerdo con el circuito de la Figura 7.2, las líneas de dirección A_0 a A_{11} activan las entradas de dirección de la ROM mientras que A_{12}–A_{15} son decodificadas por el decodificador de direcciones, activando la entrada de selección de pastilla ($\overline{CS}$) de la ROM. Poco tiempo después la salida de lectura ($\overline{RD}$) de la MPU activa la habilitación de salida de la ROM, y el dato almacenado es colocado en el bus de datos y recogido del bus por la MPU. En la Figura 7.4 se muestra una limitación de tiempos críticos. Una vez que las líneas de dirección han establecido sus respectivos niveles lógicos y la entrada $\overline{CS}$ de la ROM ha sido activada, se tarda cierto tiempo en acceder a la palabra de memoria. Este se denomina *tiempo de acceso de lectura* y es característico de la ROM o PROM particular que se esté utilizando y se define como el tiempo que se necesita para que los decodificadores internos de la ROM localicen el byte correcto en memoria.

Observar el uso de pequeños círculos y flechas en la Figura 7.4. Estos indicadores se utilizan

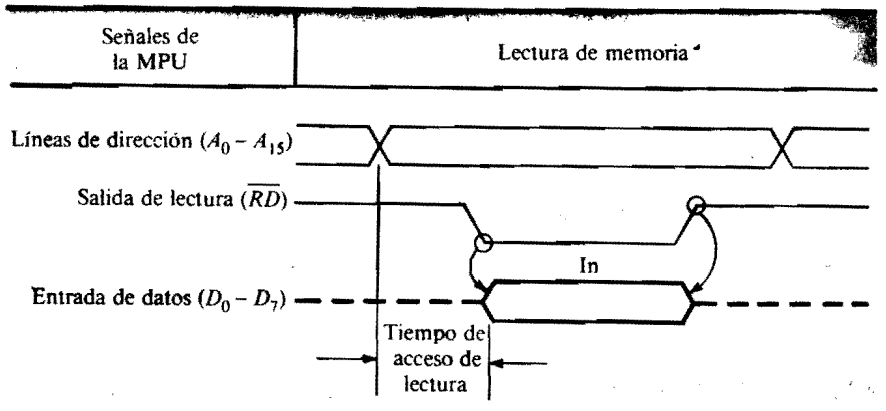


Figura 7.4. Diagrama de tiempos. Señales de la MPU durante la operación de lectura de la ROM.

para simbolizar una *relación causa-efecto* importante en el diagrama de tiempos. A título de ejemplo, en la Figura 7.4 la transición H a L (ALTA a BAJA) de la salida de lectura hace que las patillas del bus de datos de la MPU conmuten de su condición de tres estados para aceptar como entrada el dato del bus de datos. Los trazos de las líneas de entrada de datos significan condición de tres estados o de alta impedancia. De nuevo, en la transición L a H de la salida $\overline{RD}$ las patillas del bus de datos de la MPU vuelven a conmutar a su condición de tres estados y dejan de aceptar datos del bus de datos. El diagrama de tiempos de la Figura 7.4 solamente muestra señales de la MPU y no las entradas y salidas de la ROM.

PROBLEMAS RESUELTOS

7.1. La memoria de sólo lectura de la Figura 7.2 puede almacenar _____ (número) palabras, cada una de _____ (número) bits.

Solución:

La unidad de memoria de la Figura 7.2 se denomina ROM de 4K × 8 bits, lo que significa que puede almacenar 4K (ó 4096) palabras, cada una de 8 bits.

7.2. Una ROM de 4K necesita _____ (números) entradas de dirección para decodificar las 4096 posiciones separadas de memoria.

Solución:

De acuerdo con la Figura 7.2, una ROM de 4K necesitará 12 entradas de dirección para decodificar y acceder a las 4096 posiciones separadas de memoria.

7.3. Acudir a la Figura 7.2. Las 4 líneas de dirección más significativas son decodificadas por el decodificador de _____ (direcciones, la ROM).

Solución:

Las 4 líneas de dirección más significativas (A_{12}–A_{15}) son decodificadas por el decodificador de direcciones.

7.4. ¿Qué segmento del mapa de memoria de la Figura 7.3 seleccionará el decodificador de direcciones si la MPU pone en el bus de direcciones $0010\ 0000\ 0000\ 1111_2$?

Solución:

Los 4 bits más significativos (0010 en este ejemplo) son decodificados por el decodificador de direcciones y determina, en este caso, que se accederá al segmento 2. El segmento 2 de la Figura 7.3 incluye 4096 posiciones de memoria numeradas desde 2000 hasta 2FFFH.

7.5. Acudir a la Figura 7.4. Durante una operación de lectura de memoria, la(s) última(s) salida(s) activada(s) de la MPU era(n) la(s) línea(s) _____ (A_0–A_{15}, $\overline{RD}$).

Solución:

Según la información de la Figura 7.4, la última salida activada de la MPU fue la línea $\overline{RD}$.

7.6. Cuando la salida $\overline{RD}$ de la MPU alcanza el nivel BAJO, habilita la entrada _____ ($\overline{CS}$, $\overline{OE}$) de la ROM, lo que hace que las patillas del bus de datos de la MPU _____ (acepten datos del bus de datos, vayan a la condición de tres estados).

Solución:

De acuerdo con la Figura 7.2 y 7.4, cuando la salida $\overline{RD}$ de la MPU alcanza el nivel BAJO, habilita la entrada $\overline{OE}$ de la ROM para que las patillas del bus de datos de la MPU acepten datos del bus de datos.

7.7. La característica de la ROM relacionada con el tiempo que tarda en ser decodificada una dirección y accedida a una posición específica de memoria se denomina tiempo de _____ _____ .

Solución:

La característica de la ROM que trata del tiempo que tarda en ser decodificada una dirección y accedida una posición específica de memoria se denomina tiempo de acceso de lectura.

7.3. INTERFAZ CON LA RAM

Los dispositivos de almacenamiento de lectura/escritura habitualmente se denominan RAM. Las RAM se clasifican en dinámicas y estáticas. Esta sección tratará de las RAM estáticas, que son más fáciles de conectar mediante interfaces que las de tipo dinámico.

Considerar la MPU y la sección RAM del sistema representado en la Figura 7.5. Observar que la RAM está organizada como una unidad de 4K (realmente 4096 palabras) × 8 bits, es decir 4096 palabras, de 8 bits cada una. Es común que esta gran unidad RAM estática esté formada por muchos CI de RAM. Uno de estos módulos de la memoria de 4K × 8 bits contiene 32 CI de

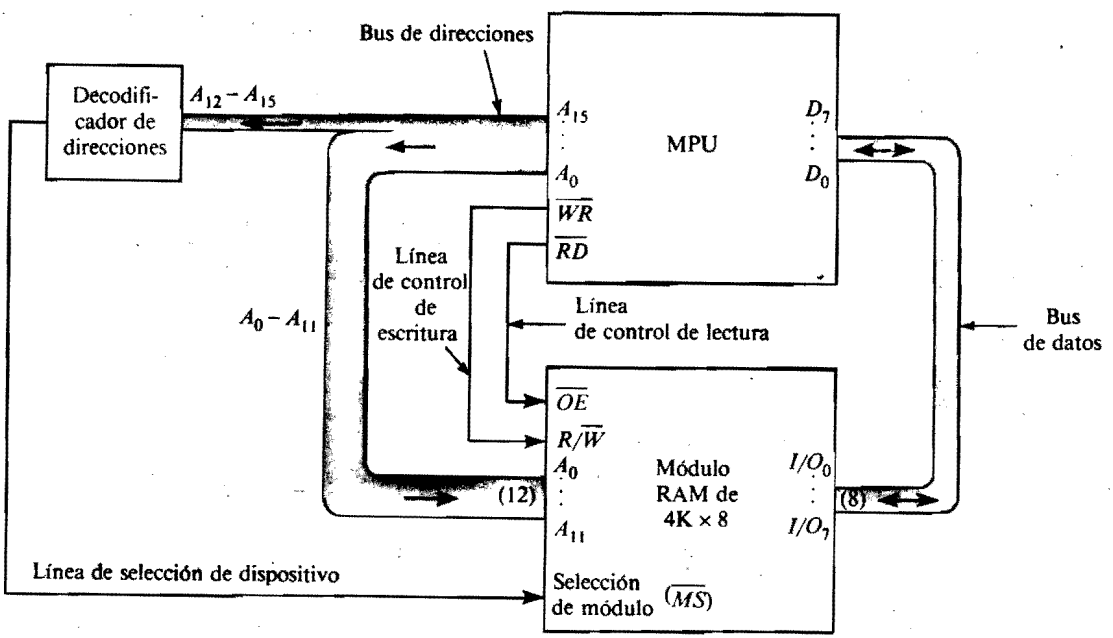


Figura 7.5. Interfaz de la RAM con el microprocesador.

RAM estática de 1024 × 1 bits. El módulo de memoria o tarjeta de memoria contiene también unos 10 CI extra para puertas y buffers. Esta unidad de memoria RAM de 4K × 8 bits representada como un bloque en la Figura 7.5 es por sí misma un sistema complejo.

El mapa de memoria del sistema basado en microprocesador se muestra, de nuevo, en la Figura 7.6. La memoria RAM de 4K se ha añadido, arbitrariamente, en el tercer segmento (llamado segmento 2). Igual que con la ROM, el decodificador de direcciones de la Figura 7.5 generará una señal para habilitar a la línea de selección de dispositivo. El decodificador de direcciones enviará un pulso BAJO a la entrada de selección de módulo ($\overline{MS}$) para habilitar la RAM solamente cuando las 4 líneas de dirección más significativas (A_{12}–A_{15}) sean 0010_2. La decodificación de las 12 líneas de dirección menos significativas (A_0–A_{11}) la realiza la circuitería de decodificación de la RAM.

El bus de datos es un camino de 8 bits y dos direcciones para la lectura o escritura de datos en la RAM de la Figura 7.5. Las 12 líneas de dirección menos significativas de la MPU van directamente a las entradas de dirección del módulo RAM, vía bus de direcciones, las 4 líneas de dirección más significativas de la MPU están conectadas al decodificador de direcciones. La salida de escritura ($\overline{WR}$) de la MPU está conectada, vía la línea de control de escritura, a la

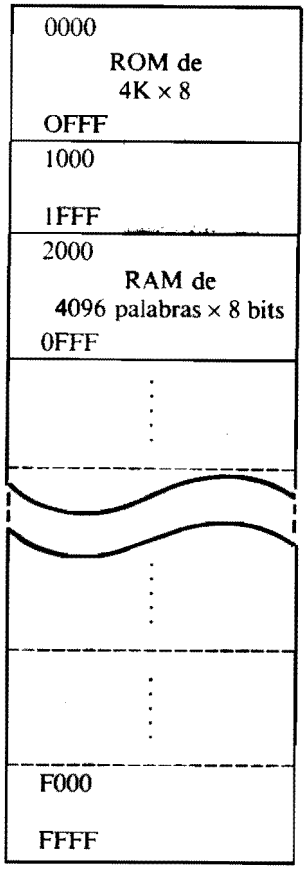


Figura 7.6. Mapa de memoria.

entrada $R/\overline{W}$ de la RAM. Observar que esta entrada es de lectura/escritura, lo que significa que cuando la MPU no habilita la salida de escritura con un nivel BAJO, el terminal $\overline{WR}$ pone un nivel ALTO, que especifica una operación de lectura en la RAM. La salida de lectura ($\overline{RD}$) de la MPU es conectada, vía la línea de control de lectura, a la habilitación de salida ($\overline{OE}$) de la RAM. Un nivel BAJO en la línea de control de lectura habilitará las salidas del módulo RAM.

El diagrama de tiempos de la Figura 7.7 muestra las señales del microprocesador y de la RAM durante la operación de lectura. Las líneas de dirección de la MPU se incializan con una dirección válida y entonces la salida de lectura ($\overline{RD}$) va al nivel BAJO. El bus de datos pasa de la condición de tres estados a una condición de entrada. La MPU está lista para aceptar datos del bus de datos. Las entradas de selección de módulo de RAM ($\overline{MS}$) y de habilitación de salida ($\overline{OE}$) están excitadas en el nivel BAJO, o habilitadas por el decodificador de direcciones y la línea de control de lectura de la MPU. La entrada de lectura/escritura ($R/\overline{W}$) es mantenida en el nivel ALTO, o en el modo de lectura. Inmediatamente después que la habilitación de salida ($\overline{OE}$) alcanza el nivel BAJO, son activadas las salidas de datos. El dato almacenado es colocado en el bus de datos por las salidas de la RAM. Como en una ROM, el *tiempo de acceso de lectura* es una característica importante de la RAM. El tiempo máximo de acceso de lectura puede variar de 250 a 1000 ns para RAM estáticas comunes.

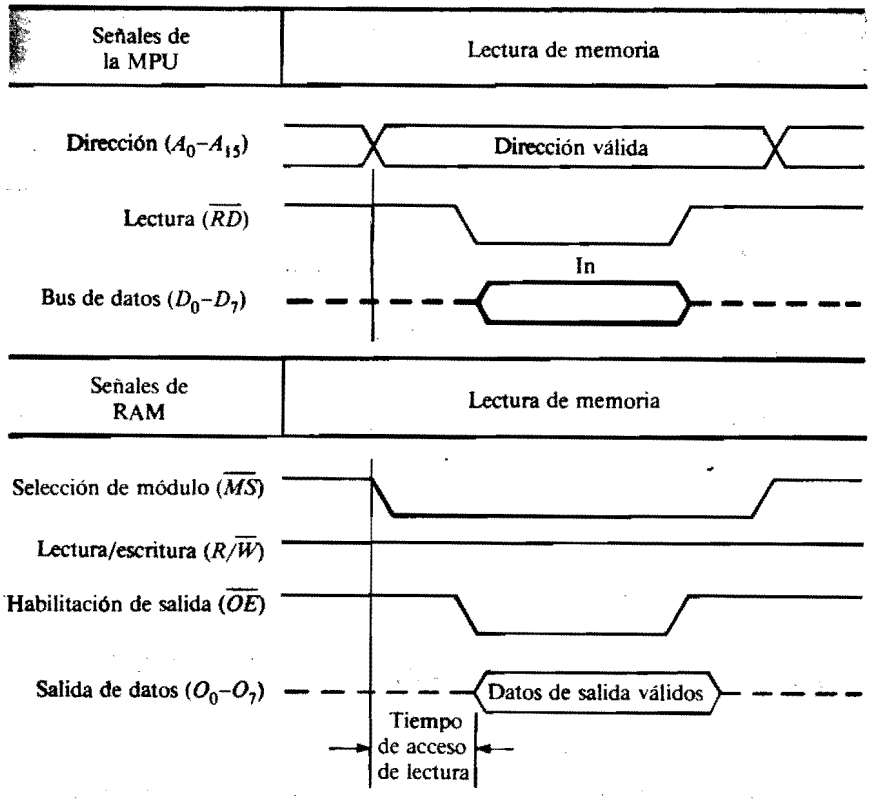


Figura 7.7. Diagrama de tiempos. Señales MPU y RAM durante la operación de lectura de memoria.

El diagrama de tiempos para la operación de escritura en memoria RAM se muestra en la Figura 7.8. La secuencia de eventos durante la operación de escritura implica primero el envío de una dirección a la RAM y al decodificador de direcciones, éste a su vez activa la entrada de selección de módulo ($\overline{MS}$) (que puede ser la entrada de selección de pastilla en un simple CI). Después de un tiempo denominado *tiempo de establecimiento de la dirección*, el pulso de escritura de la MPU ($\overline{WR}$) activa la entrada de lectura/escritura ($R/\overline{W}$) de la RAM y la coloca en el modo de escritura. El pulso de escritura debe ser habilitado como mínimo un cierto tiempo, denominado *tiempo del pulso de escritura* (o anchura del pulso de escritura). Al mismo tiempo que sale el pulso de escritura, el dato que se va a escribir en memoria es colocado en el bus de datos por la MPU. Inmediatamente después, la RAM acepta este dato y lo escribe en la posición direccionada. El tiempo mínimo para un ciclo de escritura se denomina *tiempo de ciclo de escritura* y está identificado cerca la parte superior del diagrama de la Figura 7.8. El ciclo de escritura mínimo típico para una RAM estática común puede variar de 250 a 1000 ns. El mínimo tiempo de preparación de la dirección para el mismo grupo de RAM puede variar de 20 a 200 ns, mientras que el mínimo tiempo de pulso de escritura puede ser de 180 a 750 ns.

El etiquetado y el número de entradas de las RAM comerciales varían. Las señales de salida de los microprocesadores comerciales, también varían considerablemente. Los manuales de datos de los fabricantes deben ser examinados cuidadosamente para estos detalles. Las señales del micro-

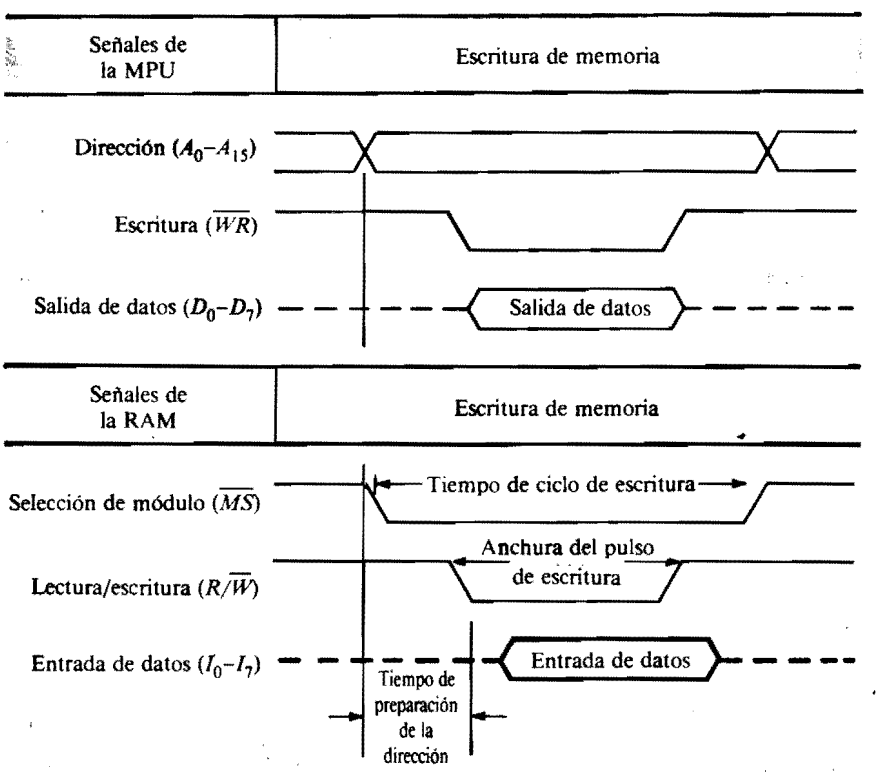


Figura 7.8. Diagrama de tiempos. Señales de la MPU y la RAM durante la operación de escritura en memoria.

procesador y RAM no aparecen en los libros de datos del fabricante, en diagramas combinados como los de las Figuras 7.7 y 7.8. Los diagramas de tiempo han de examinarse separadamente para determinar si son compatibles. Como las celdas de memoria necesitan «refrescarse» cada pocos microsegundos, las RAM dinámicas son más difíciles de conectar mediante interfaces que las unidades estáticas de memoria.

PROBLEMAS RESUELTOS

7.8. La RAM es funcionalmente un dispositivo de almacenamiento de _____ (sólo lectura, lectura/escritura).

Solución:

La RAM es funcionalmente un dispositivo de almacenamiento de lectura/escritura.

7.9. ¿Qué tipo de RAM —estática o dinámica— es más fácil de conectar mediante una interfaz a un microprocesador?

Solución:

La RAM estática.

7.10. Acudir a la Figura 7.6. Si la MPU direcciona a 2030H, el decodificador de direcciones seleccionará el segmento _____ (número) de memoria. Un dispositivo de almacenamiento _____ (RAM, ROM) reside en esta posición en este sistema.

Solución:

Si la MPU de la Figura 7.6 direcciona a 2030H, el decodificador de direcciones seleccionará el segmento 2 (0010_2) de memoria. Un dispositivo de almacenamiento RAM reside en esta posición de acuerdo con el mapa de memoria de la Figura 7.6.

7.11. El bloque de RAM de la Figura 7.5 contiene _____ (número) palabras, cada una de _____ (número) bits de ancho. Una RAM de este tamaño, probablemente, está formada por un _____ (grupo de CI, solo CI).

Solución:

El bloque RAM de la Figura 7.5 contiene 4K (realmente 4096) palabras, cada una de 8 bits. Una RAM de este tamaño, probablemente, está formada por un grupo de CI. Una tarjeta de memoria (circuito impreso con CI) contiene unos 40 CI.

7.12. Para escribir en la RAM de la Figura 7.5, las entradas de dirección A_0 a A_{11} deben ser activadas, por tanto $\overline{MS}$ debe ser habilitado con un nivel _____ (ALTO, BAJO) mientras que la entrada $R/\overline{W}$ es mantenida en un nivel _____ (ALTO, BAJO). Finalmente, cuando la entrada $\overline{OE}$ es excitada con un nivel _____ (ALTO, BAJO), las patillas etiquetadas I/O_0 a I/O_7 son las _____ (entradas, salidas) de la operación de escritura.

Solución:

Para escribir en la RAM de la Figura 7.5, las entradas de dirección A_0 a A_7 deben ser activadas, por tanto $\overline{MS}$ debe ser habilitado con un nivel BAJO mientras que la entrada $R/\overline{W}$ es mantenida en el nivel BAJO. Finalmente, cuando la entrada $\overline{OE}$ es excitada con un nivel BAJO, las patillas etiquetadas I/O_0 a I/O_7 son las entradas de la operación de escritura.

7.13. Acudir a la Figura 7.5. Citar las dos líneas de control que constituyen el «bus de control» de este sistema.

Solución:

El «bus de control» comprende las líneas de control de lectura y de escritura del sistema de la Figura 7.5.

7.14. Acudir a la Figura 7.7. ¿Qué hace que la línea de selección de módulo ($\overline{MS}$) caiga del nivel ALTO al BAJO al comienzo de la operación de lectura de memoria?

Solución:

Los 4 bits más significativos del bus de direcciones son decodificados por el decodificador de direcciones, que activa la línea de selección de módulo ($\overline{MS}$) cuando 0010_2 está presente en sus entradas.

7.15. Acudir a la Figura 7.7. El tiempo de acceso de lectura es el tiempo requerido para que los datos almacenados alcancen las salidas de la RAM, desde el instante en que una dirección válida alcanza las entradas de dirección. El tiempo de acceso de lectura de una memoria RAM típica debe ser de unos 500 _____ (μs, ns).

Solución:

El tiempo de acceso de lectura para una memoria RAM típica debe ser de unos 500 ns.

7.16. Durante la operación de escritura en la Figura 7.5, las patillas D_0 a D_7 del microprocesador son consideradas _____ (entradas, salidas) y están _____ (recibiendo datos del, enviando datos al) bus de datos.

Solución:

Durante la operación de escritura, las patillas D_0 a D_7 del microprocesador se consideran salidas y están enviando datos al bus de datos.

7.17. Acudir a la Figura 7.8. El tiempo del ciclo de escritura es _____ (mayor, menor) que el tiempo del pulso de escritura de una RAM.

Solución:

De acuerdo con la Figura 7.8, el tiempo del ciclo de escritura es mayor que el tiempo del pulso de escritura de una RAM.

7.18. Acudir a la Figura 7.8. La entrada $R/\overline{W}$ a la RAM es puesta en el nivel BAJO por la señal de control _____ del microprocesador.

Solución:

De acuerdo con la Figura 7.5, la señal de control $\overline{WR}$ de la MPU pone la entrada $R/\overline{W}$ de la RAM al nivel BAJO durante una operación de escritura.

7.4. INTERFACES BASICAS DE ENTRADA/SALIDA

Una operación de entrada o salida es el acto de transferir datos a o desde un dispositivo periférico seleccionado. El microprocesador es el foco de todas las operaciones, por tanto una *entrada* significa que el dato fluye hacia la MPU mientras que una *salida* significa que el dato fluye de

la MPU. Aquellas posiciones donde el dato entra o sale, se denominan *puertos* de entrada o salida.

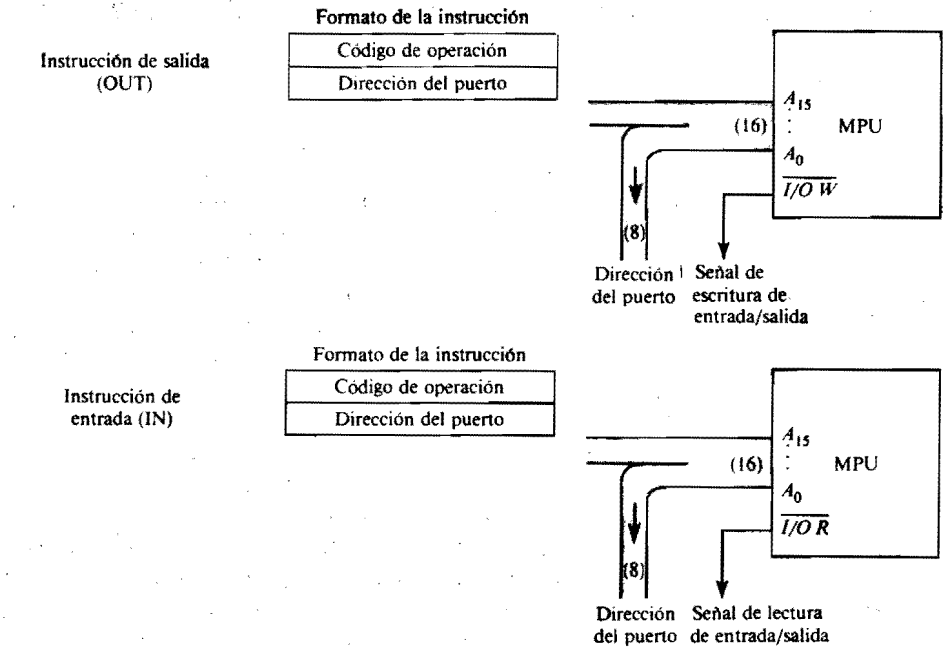
De la tabla de instrucciones de transferencia de datos de la Figura 6.21, se deduce que el microprocesador genérico utiliza las instrucciones IN (Entrar) y OUT (Salir) para transferir datos a y desde los puertos de entrada/salida. Estas instrucciones de transferencia de datos están ilustradas en la Figura 7.9*a*. La instrucción de salida está representada por el nemotécnico OUT en los programas en lenguaje ensamblador, mientras que la instrucción de entrada utiliza el nemotécnico IN. Los formatos de instrucción para estas operaciones se reproducen en la Figura 7.9*a*, donde aparece el código de op seguido de un número de dispositivo o dirección del puerto. Se puede seleccionar uno de 256 (2^8) puertos, ya que la dirección del puerto proviene de las 8 líneas de dirección menos significativas (A_0–A_7). La Figura 7.9*a* también muestra dos señales adicionales de control de salida del microprocesador genérico. Cuando se utiliza la operación OUT, se utiliza una señal especial de *escritura de entrada/salida* ($\overline{I/O\ W}$), la operación IN también requiere el uso de una señal de salida especial llamada señal de *lectura de entrada/salida* ($\overline{I/O\ R}$). Ambas señales de salida son activas en el nivel BAJO y la Figura 7.9*a* muestra que provienen del microprocesador. El uso de señales especiales de control, tales como $\overline{I/O\ W}$ e $\overline{I/O\ R}$ en la Figura 7.9*a*, se conoce como *entrada/salida aislada* o por *acumulador*.

Las transferencias de datos que utilizan las instrucciones IN y OUT se denominan *E/S controladas por programa*. Las instrucciones de programa están controlando las transferencias de datos durante las operaciones IN y OUT. Las transferencias de datos pueden ser iniciadas por el dispositivo periférico cuando dice «Estoy listo para enviar o recibir datos». Los dispositivos periféricos utilizan *interrupciones* para iniciar la acción de la MPU. Recordar que cuando la MPU recibe una petición de interrupción, finaliza la instrucción que está ejecutando y salta a una *rutina de servicio de interrupción* en la memoria de programa. Esta rutina de servicio de interrupción puede incluir operaciones de entrada y salida.

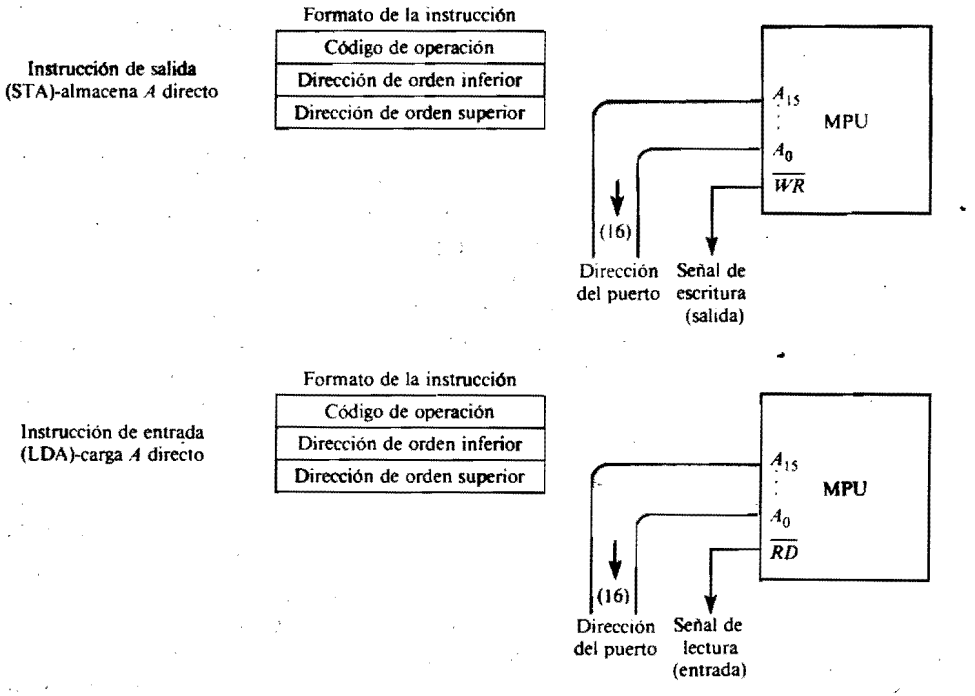
Existen dos técnicas de E/S controladas por programa. La técnica de E/S aisladas se observó en la Figura 7.9*a* utilizando las instrucciones especiales IN y OUT. La otra técnica trata a las posiciones de entrada y salida como direcciones regulares de memoria. Esta técnica se denomina *E/S de mapa de memoria* (o con correspondencia en el espacio de direcciones de memoria). Con esta técnica se utilizan instrucciones regulares de acceso de memoria. En la Figura 7.9*b* la instrucción «almacena *A* directo» se utiliza para sacar datos a un puerto de salida. A continuación, en la misma figura, la instrucción regular «carga *A* directo» se utiliza para introducir datos desde un puerto o dispositivo. En la Figura 7.9*b* se observa que las líneas de dirección deben ser decodificadas y utilizadas para seleccionar la dirección exacta de un puerto de entrada o salida. Las señales habituales de control de escritura ($\overline{WR}$) y lectura ($\overline{RD}$) también se utilizan para entrar o sacar datos. Cualquier instrucción de acceso de memoria puede utilizarse para entrar y sacar datos utilizando la técnica de E/S de mapa de memoria.

La técnica de E/S de mapa de memoria es, probablemente, la más común y puede utilizarse con cualquier microprocesador. La técnica de E/S aisladas puede utilizarse sólo con microprocesadores que tengan instrucciones IN y OUT separadas, entradas/salidas especiales y salidas de control de lectura y escritura.

Es común referenciar una salida como «salida a un dispositivo periférico». En la práctica real, sin embargo, la salida del microprocesador no va directamente a un dispositivo periférico, sino a un dispositivo de memoria que almacena el dato para el periférico, como puede verse en el sistema representado en la Figura 7.1. Los bloques etiquetados «adaptador de interfaz de entrada» y «adaptador de interfaz de salida» son dispositivos de memoria. Es común que los adaptadores de interfaz de E/S tengan características distintas de las de memoria.



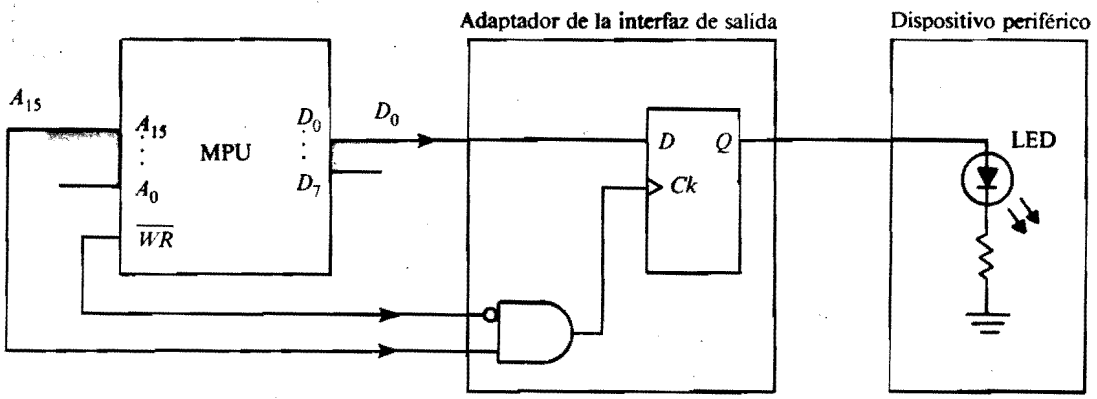
(a) **Instrucciones de E/S aisladas y señales MPU**



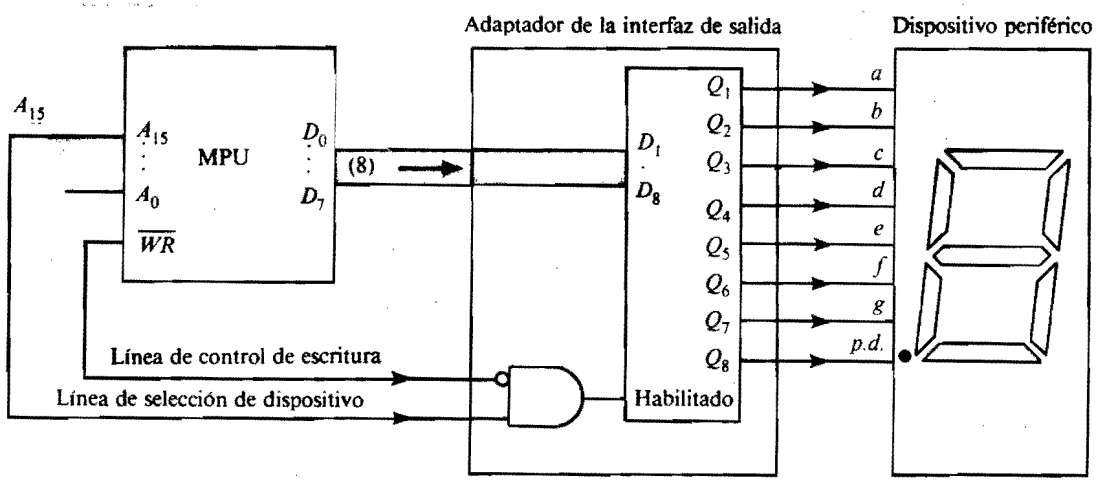
(b) Instrucciones de E/S de mapa de memoria y señales MPU

Figura 7.9.

Un dispositivo periférico y una interfaz elemental de salida y se muestran en la Figura 7.10*a*. Observar que el indicador de salida es un sencillo diodo emisor de luz (LED). El adaptador de la interfaz de salida contiene un flip-flop *D*. Suponer que la MPU está ejecutando la instrucción «almacenar *A* directo» y que el acumulador contiene 00000001. Suponer la dirección 8000H, la línea A_{15} está en el nivel ALTO y activa la entrada inferior de la puerta AND. Inmediatamente después, aparece un nivel ALTO en la línea del bus de datos (D_0). La señal de control ($\overline{WR}$) va al nivel BAJO, activando la puerta AND. Esto hace que el bit 1 se almacene en el flip-flop *D*. El nivel ALTO en la salida *Q* del flip-flop hace que luzca el LED o indicador de bit. La MPU entonces procede con otras actividades, pero el adaptador de interfaz de salida (memoria) hace que el LED siga luciendo. La idea ilustrada para un solo bit de datos en la Figura 7.10*a* es fundamental para todas las salidas.

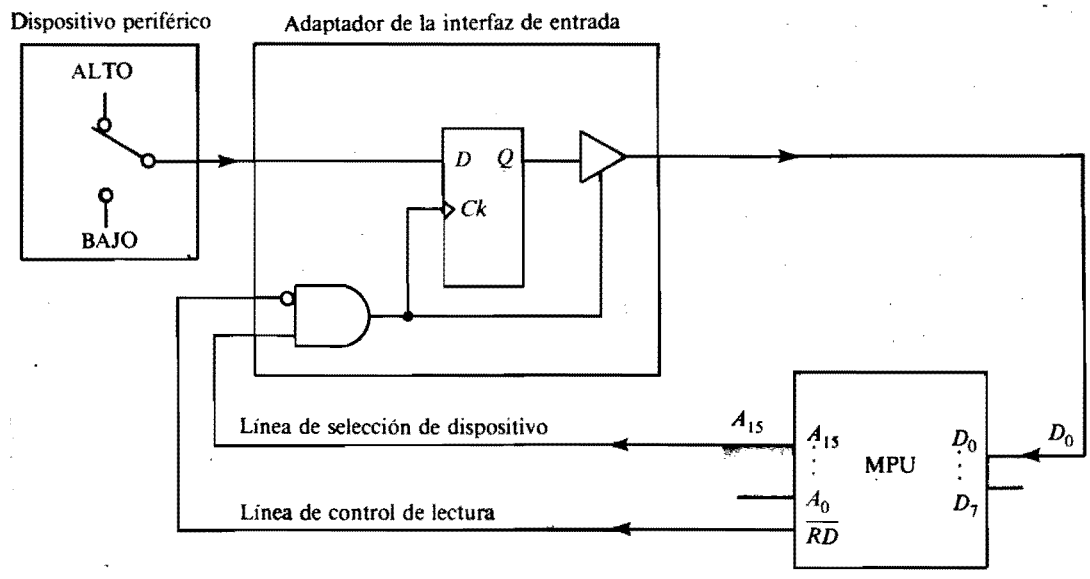


(a) Interfaz de un simple indicador LED del bit de salida

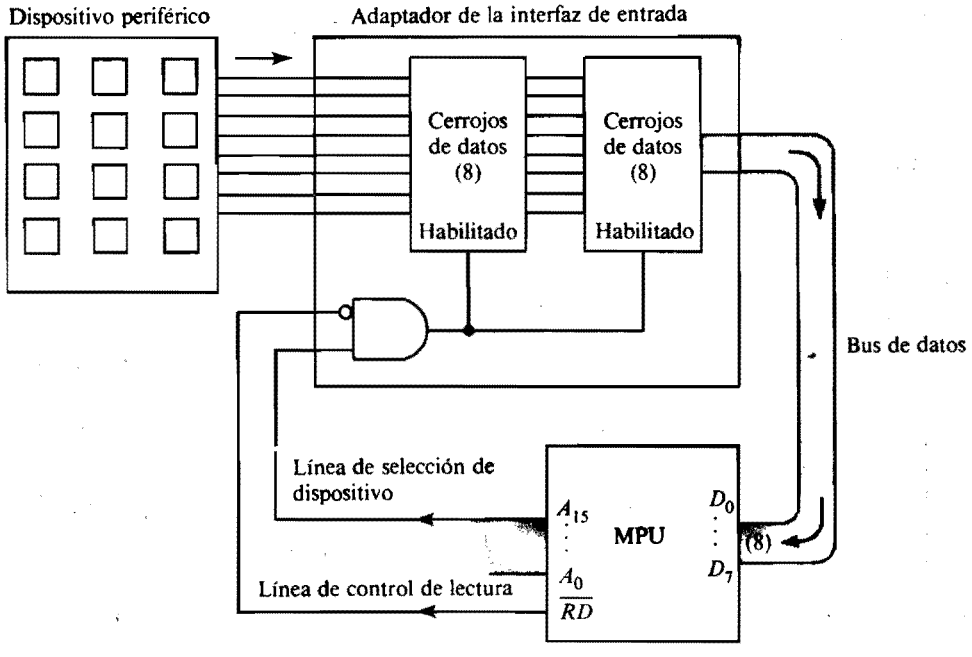


(b) Interfaz del visualizador de siete segmentos

Figura 7.10.



(a) Interfaz de una entrada de conmutador



(b) Interfaz de entrada de teclado

Figura 7.11.

La interfaz de salida de la Figura 7.10*b* es algo más complicada. Transfiere 8 bits de datos de la MPU, vía bus de datos, al adaptador de interfaz de salida, que los almacena en un cerrojo de 8 bits en la forma habitual. Observar que la línea de dirección A_{15} debe estar en ALTA mientras que la salida de control de escritura ($\overline{WR}$) debe ser un pulso BAJO durante el tiempo que esté el dato en el visualizador LED de siete segmentos. Cada segmento del visualizador LED (*a-g* y punto decimal) actúa exactamente como el LED de la Figura 7.10*a*. Un nivel ALTO en cualquier salida Q del cerrojo hará que luzca el segmento.

Otro periférico elemental y dispositivo de interfaz se muestra en la Figura 7.11*a*. El dispositivo periférico es un sencillo conmutador que selecciona un nivel ALTO o un nivel BAJO. Sólo se conecta una línea de entrada a la entrada D del flip-flop de datos dentro del adaptador de interfaz de entrada. Cuando la MPU ejecuta una instrucción como «cargar A directo desde la posición de memoria 8000H», la puerta AND se activa, encerrando el dato de entrada. Se habilita el buffer de salida, permitiendo que el dato pase a través de la línea del bus de datos (D_0). La MPU acepta, del bus de datos, el bit de dato ALTO y lo transfiere al acumulador. Inmediatamente después, se inhabilita la puerta AND, desactivando el buffer del bus de datos. La salida de adaptador de interfaz de entrada está ahora inhabilitada y en su estado de alta impedancia y no tiene efecto sobre otras transferencias del bus de datos.

La técnica de entrada, de mapa de memoria, se utiliza en la Figura 7.11. Normalmente, varias líneas de dirección deben ser decodificadas por un decodificador de direcciones para activar la línea de selección de dispositivo. La Figura 7.11*b* es una extensión de la entrada de un solo bit. Aquí un dato paralelo de 8 bits se introduce desde un teclado. El control de lectura ($\overline{RD}$) y direccionamiento simple son los mismos que se utilizaron en los ejemplos anteriores.

La E/S de mapa de memoria se utiliza con los microprocesadores comunes como 6800, 6809, 68000, 6502 y 65816. Las E/S aisladas (a veces llamadas también E/S de canal de datos) son utilizadas por los microprocesadores comunes como 8080/8085, Z80, 8086, 8088 y Z8000.

PROBLEMAS RESUELTOS

7.19. Una microcomputadora que utiliza las instrucciones especiales IN y OUT se dice que utiliza la técnica de E/S _____ (aisladas, de mapa de memoria).

Solución:

Una microcomputadora que utiliza las instrucciones especiales IN y OUT se dice que utiliza la técnica de E/S aisladas. El uso de esta técnica depende que el microprocesador tenga salidas especiales de control tal como lectura y escritura de E/S.

7.20. La técnica de E/S que trata los puertos de entrada y salida como direcciones regulares de memoria se denomina de E/S _____ (aisladas, de mapa de memoria).

Solución:

La técnica de E/S que trata los puertos de entrada y salida como direcciones regulares de memoria se denomina de E/S de mapa de memoria.

7.21. Si las salidas de control de lectura ($\overline{RD}$) y escritura ($\overline{WR}$) de memoria de la MPU normal están conectadas a adaptadores de interfaces de E/S, se está utilizando la técnica de E/S _____ (aisladas, de mapa de memoria).

Solución:

Si las salidas de control de escritura ($\overline{WR}$) y de lectura ($\overline{RD}$) de memoria de la MPU normal están conectadas a adaptadores de interfaces de E/S, se está utilizando la técnica de E/S de mapa de memoria.

7.22. En la Figura 7.11*b* se transfiere un dato al acumulador de la MPU después de que la línea de dirección A_{15} alcanza el nivel _____ (ALTO, BAJO) y la línea de control de lectura ($\overline{RD}$) alcanza el nivel _____ (ALTO, BAJO), lo que habilita los cerrojos y buffers.

Solución:

El dato es transferido al acumulador de la MPU, en la Figura 7.11*b*, después de que la línea de dirección A_{15} alcance el nivel ALTO y la línea de control de lectura ($\overline{RD}$) alcance el nivel BAJO, lo cual habilita a los cerrojos y buffers.

7.23. Durante la operación de entrada, la MPU de la Figura 7.11*b* está ejecutando, probablemente, la instrucción _____ (IN, LDA).

Solución:

Este sistema está utilizando la técnica de E/S de mapa de memoria. Con esta técnica se utilizan instrucciones de transferencia de datos de referencia regular de memoria. Por tanto, durante la operación de entrada la MPU de la Figura 7.11*b* está ejecutando, probablemente, la instrucción LDA (carga *A* directo).

7.24. Cuando están habilitados, los buffers de datos _____ (bloquean, permiten) el paso de los datos de los cerrojos al bus de datos.

Solución:

Cuando están habilitados, los buffers de datos, permiten el paso de los datos de los cerrojos al bus de datos.

7.25. Acudir a la Figura 7.11*b*. ¿A qué dirección responderá la interfaz de entrada?

Solución:

Siempre que la línea de dirección más significativa (A_{15}) esté en ALTA, se activa la línea de selección de dispositivo. Este no es un método práctico de direccionamiento, porque los 32K (los ocho segmentos inferiores de 4K de la Figura 7.3) serán utilizados por este dispositivo de salida.

7.5. INTERFAZ CON PUERTOS PRACTICOS DE ENTRADA/SALIDA

Los fabricantes producen circuitos de interfaz de E/S en forma de CI. Uno de estos CI es el *puerto de entrada/salida Intel 8212 de 8 bits*. Este CI puede utilizarse como adaptador de puerto de entrada o salida.

Un microprocesador similar a la MPU genérica o al Intel 8080 se muestra en la Figura 7.12, conectado a un visualizador de siete segmentos, utilizando el puerto de E/S 8212. Este puerto sirve como adaptador de interfaz de salida de este sistema. El visualizador de siete segmentos es el dispositivo periférico. La MPU utiliza la técnica de E/S aisladas en este diagrama. Observar que con esta técnica la línea de selección de dispositivo de salida se *decodifica completamente* con las 8 líneas de dirección menos significativas (A_0–A_7), también se utiliza una salida especial de con-

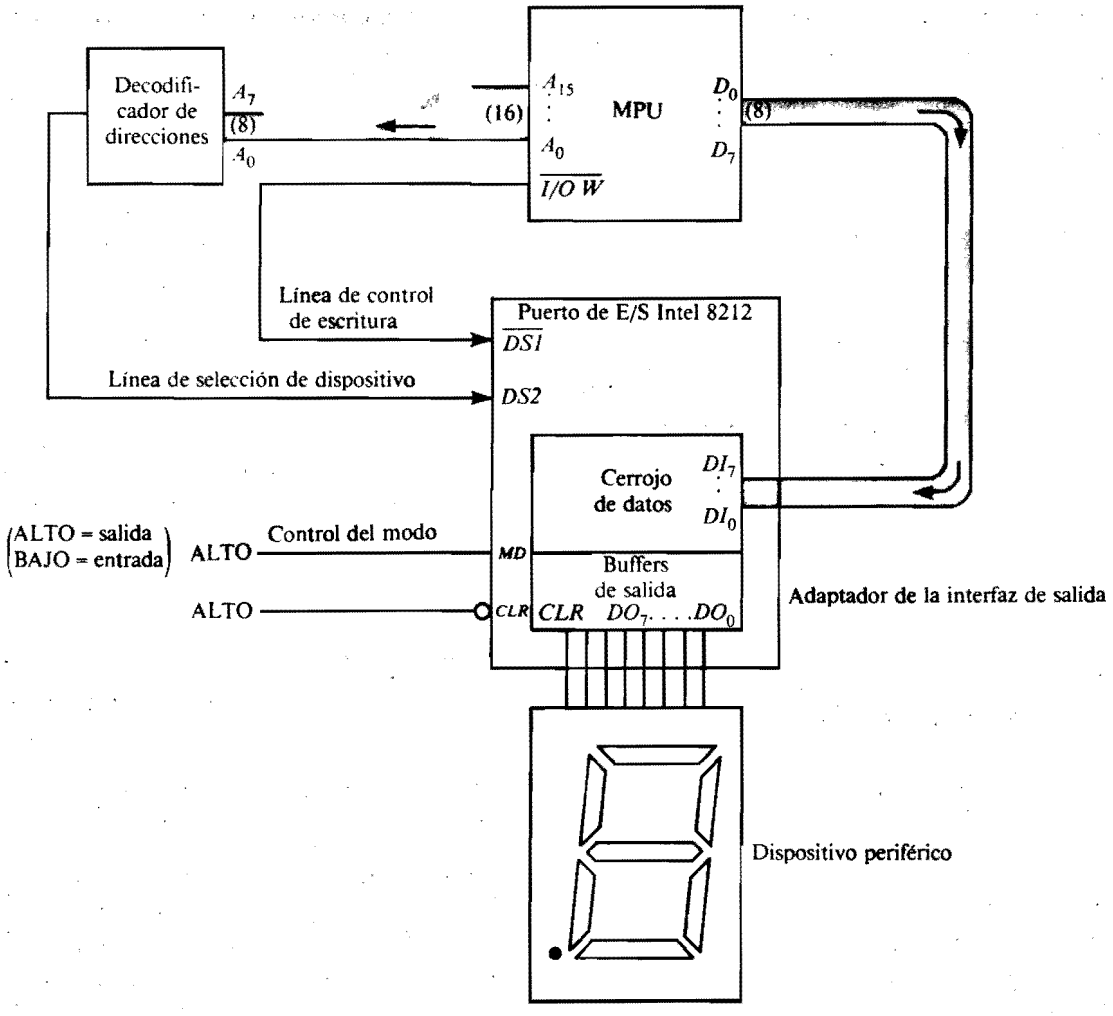


Figura 7.12. Interfaz con visualizador de siete segmentos utilizando el puerto de E/S de Intel 8212.

trol de *escritura de entrada/salida* ($\overline{I/O\ W}$) que se ha añadido a la MPU de la Figura 7.12. Recordar que la técnica de E/S aisladas sólo permite el uso de las instrucciones IN y OUT.

El CI del puerto de E/S 8212 se utiliza como puerto de salida en la Figura 7.12. Esto se determina al observar que la entrada de control de modo (*MD*) al 8212 está en ALTA, y cuando esto ocurre, el 8212 está en el *modo de salida*. Las líneas del bus de datos de la MPU se conectan a las 8 líneas de entrada de datos (DI_0–DI_7) del puerto de E/S 8212. Las 8 líneas de salida de datos (DO_0–DO_7) del 8212 están conectadas a las entradas del visualizador LED de siete segmentos. El puerto de E/S del 8212 tiene una entrada de borrado, pero en este circuito está inhabilitada con un nivel ALTO. Internamente el 8212 tiene ocho cerrojos de datos y buffers de salida. Las 2 entradas de selección de dispositivo ($\overline{DS1}$, *DS2*) son entradas de control para el puerto de E/S

8212 cuando se usa en este modo. Cuando $\overline{DS1}$ está activada por un nivel BAJO y *DS2* por un nivel ALTO, el dato del bus de datos se cargará en los cerrojos y aparecerá también en las patillas de salida (DO_0–DO_7), activando así los segmentos del visualizador LED.

La operación del puerto de E/S 8212, mientras se utiliza en el modo de salida, se detalla en la Figura 7.13. Observar en el diagrama de tiempos que tan pronto como son activadas, las señales de control de entrada ($\overline{DS1}$ y *DS2*), el dato de entrada se almacena en el puerto de E/S y está disponible en las patillas de salida. En el modo de salida, los buffers de salida están permanentemente habilitados, por lo que los datos almacenados aparecen en las patillas de salida aún después que las patillas de control $\overline{DS1}$ y *DS2* vuelvan a sus estados de inhabilitación.

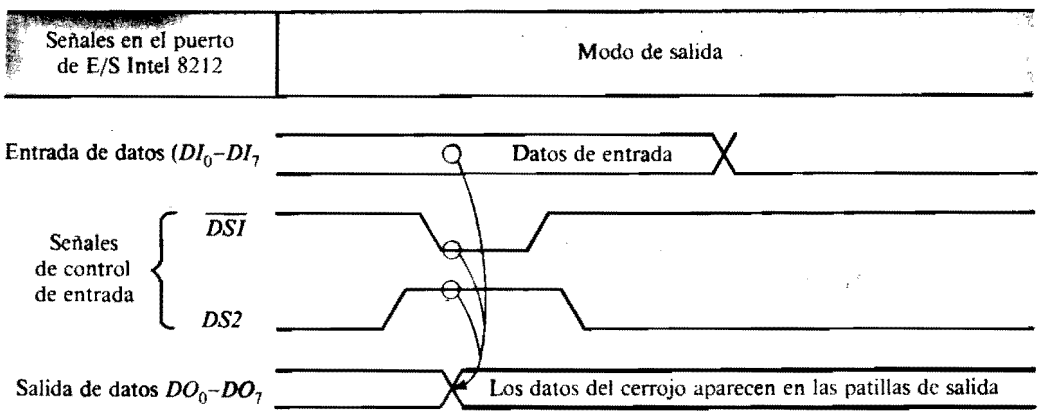


Figura 7.13. Diagrama de tiempos. Señales de E/S del puerto durante el modo de salida.

PROBLEMAS RESUELTOS

7.26. Los datos que llegan a los terminales DI_0 a DI_7 del puerto de E/S 8212 en la Figura 7.12 provienen del bus de _____ (direcciones, datos) del sistema basado en microprocesador.

Solución:

De acuerdo con la Figura 7.12, los datos que llegan a los terminales DI_0 a DI_7 del puerto de E/S del 8212 provienen del bus de datos del sistema basado en microprocesador.

7.27. El puerto de E/S 8212 en la Figura 7.12 se utiliza como puerto de _____ (entrada, salida) paralelo de 8 bits en este diagrama.

Solución:

El puerto de E/S 8212 en la Figura 7.12 se utiliza como puerto de salida paralelo de 8 bits en este diagrama.

7.28. La entrada $\overline{DS1}$ al 8212 de la Figura 7.13 es generada por _____ (el decodificador de direcciones, la señal $\overline{I/O\ W}$ de la MPU).

Solución:

De acuerdo con la Figura 7.12, la entrada $\overline{DS1}$ al puerto de E/S del 8212 es generada por la señal $\overline{I/O\ W}$ de la MPU.

7.29. La entrada *DS2* al 8212 de la Figura 7.13 es generada por _____ (el decodificador de direcciones, el bus de datos de la MPU).

Solución:

De acuerdo con la Figura 7.12, la entrada de control *DS2* del puerto de E/S del 8212 es generada por el decodificador de direcciones. Las 8 líneas de dirección menos significativas son decodificadas completamente en este sistema, lo que significa que están disponibles 256 dispositivos de salida.

7.30. Los buffers de salida del 8212 están permanentemente _____ (inhabilitados, habilitados) mientras la unidad está en su modo de salida. Esto significa que los buffers _____ (bloquean los datos de, pasan los datos a) las salidas de datos (DO_0–DO_7).

Solución:

Los buffers de salida del 8212 están permanentemente habilitados mientras que la unidad está en su modo de salida. Esto significa que los buffers pasan los datos a las salidas de datos (DO_0–DO_7).

7.31. Acudir a la Figura 7.13. El dato está encerrado en el puerto de E/S 8212 cuando están activadas ambas entradas de control _____ y _____.

Solución:

El dato está encerrado en el puerto E/S 8212 en la Figura 7.13 cuando están activadas ambas entradas de control $\overline{DS1}$ y *DS2*.

7.32. El sistema de la Figura 7.12 utiliza la técnica de interfaz de E/S _____ (aisladas, de mapa de memoria).

Solución:

El sistema de la Figura 7.12 utiliza la técnica de interfaz de E/S aisladas.

7.6. SINCRONIZACION DE TRANSFERENCIAS DE ENTRADA/SALIDA DE DATOS MEDIANTE INTERRUPCIONES

Hasta este punto se ha supuesto que cuando el programa autorizaba a la MPU una entrada desde un puerto era porque había un dato válido disponible en esa posición. Sin embargo, éste puede no ser así, ya que los dispositivos periféricos (como por ejemplo un teclado) no actúan a la misma velocidad que la MPU. Por esta razón existen varias técnicas de sincronización de transferencia de datos, éstas son las de *escrutinio (polling)* y las *interrupciones.*

La técnica de escrutinio también se denomina de E/S programada. El escrutinio es el método más simple de sincronización de E/S y se utiliza en las aplicaciones dedicadas pequeñas. La idea del escrutinio es entrar o sacar datos, utilizando un *lazo de escrutinio* en el programa.

Suponer un sistema extremadamente simple, basado en MPU, que utilice un solo conmutador de entrada y un solo indicador LED de salida. Los sistemas mostrados en las Figuras 7.10*a* y 7.11*a* pueden combinarse utilizando una sencilla MPU, un conmutador de entrada con un adaptador de interfaz de entrada y un LED de salida con un adaptador de interfaz de salida. Se puede construir un lazo de escrutinio, extremadamente sencillo, para leer primero el conmutador de entrada y escribir después esta condición en el indicador LED de salida. En este ejemplo, el lazo de lectura-escritura-lectura-escritura- etc., sería continuo. La MPU escruta la entrada y actualiza la salida cada pocos microsegundos.

El lazo de escrutinio más común puede consistir en escrutar los dispositivos de E/S. La MPU pregunta al primer dispositivo si necesita servicio. La necesidad, o no necesidad, de servicio está indicada por el valor verdadero, o falso, de un señalizador. Si el dispositivo necesita servicio, la MPU presta servicio a ese dispositivo, si no lo necesita, la MPU va al segundo dispositivo de E/S, etcétera.

El sistema representado en la Figura 7.14 utiliza una línea de petición de interrupción para notificar al microprocesador cuando está listo un dato para ser transferido a la CPU. Recordar que el microprocesador genérico tenía una entrada de petición de interrupción (INTR). Cuando se activaba con un nivel ALTO, la MPU finalizaba la instrucción actual, almacenaba los contenidos del registro actual y del contador de programa en la pila y saltaba a una *rutina especial de servicio de interrupción.* Después de atender la interrupción, la MPU continuaba la ejecución del programa principal. En este ejemplo, la rutina de servicio de interrupción puede ser tan sencilla como introducir y almacenar entradas del teclado.

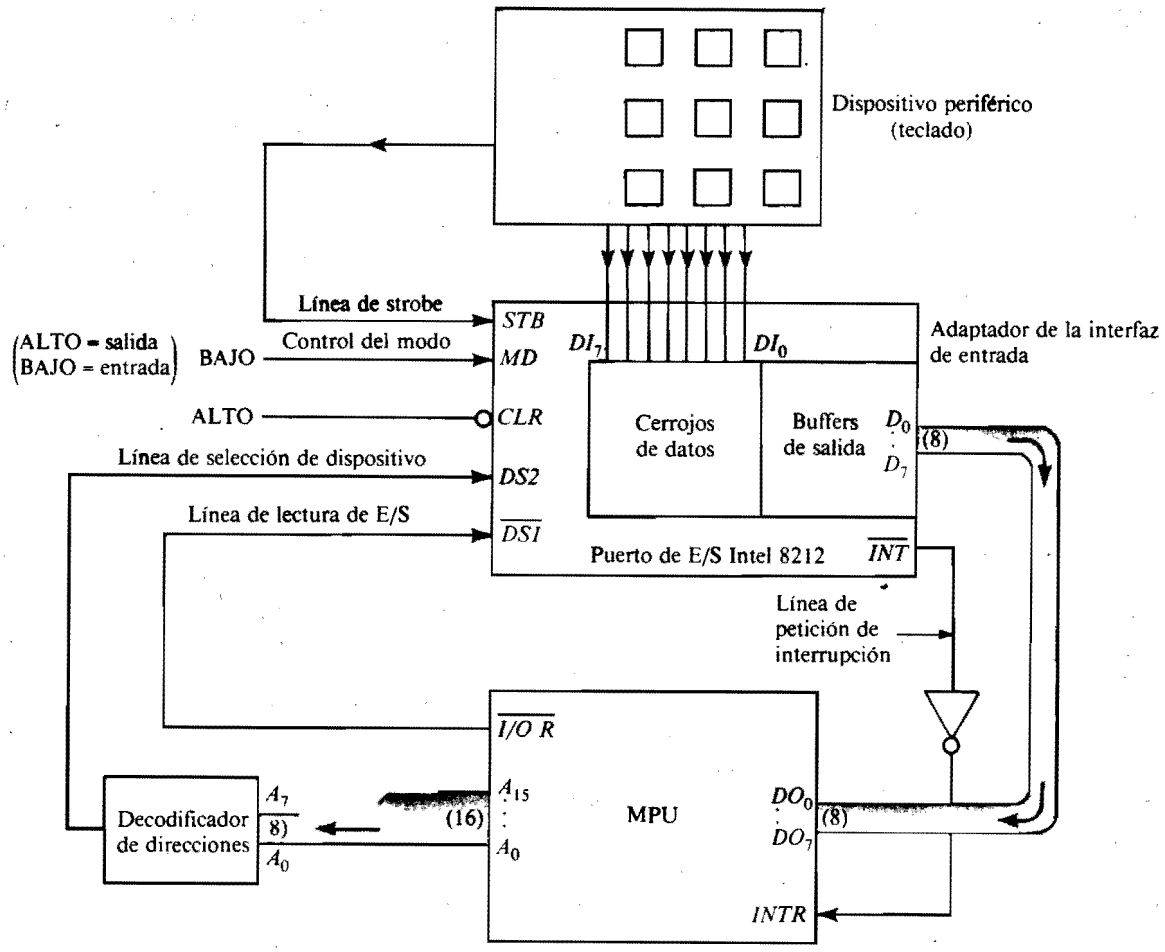


Figura 7.14. Interfaz con el teclado utilizando el puerto de E/S Intel 8212 con una sola interrupción.

Considerar el sistema de la Figura 7.14. El microprocesador está interconectado con un teclado utilizando el CI puerto de E/S Intel 8212 como adaptador de interfaz de entrada. El control de modo (*MD*) del 8212 está en el nivel BAJO, lo que significa que opera como un adaptador de entrada. La entrada de borrado (*CLR*) a la unidad 8212 está inhabilitada con un nivel ALTO. El dato paralelo de 8 bits del teclado llega a las entradas de datos (DI_0–DI_7) del 8212. En este sistema, la *entrada STB controla los cerrojos de datos,* mientras las *entradas de selección de dispositivo* del 8212 ($\overline{DS1}$ y *DS2*) *controlan los buffers de salida.* El teclado debe colocar una palabra de 8 bits en la entrada de datos de la unidad 8212 y también debe producir un pulso ALTO (*strobe*) para almacenar este dato en los cerrojos, de datos, internos del puerto de E/S. Estos eventos están representados en la Figura 7.15 por las tres formas de onda superiores. De nuevo, es el «strobe» ALTO del teclado el que almacena los datos en el puerto de E/S 8212.

De acuerdo con la forma de onda inferior de la Figura 7.15, inmediatamente después que el dato está en los cerrojos de datos, la circuitería de la unidad 8212 produce un pulso de interrupción de salida ($\overline{INT}$). Algún tiempo después de que este pulso llegue a la MPU, vía línea petición de interrupción, son habilitados los controles del buffer de salida de la unidad 8212 ($\overline{DS1}$ y *DS2*). Las formas de onda tercera a sexta de la Figura 7.15 muestran que el dato almacenado es colocado en el bus de datos, durante un corto tiempo, por los controles $\overline{DS1}$ y *DS2* de la unidad 8212. Observar que las salidas del puerto de E/S 8212 vuelven a su condición de tres estados, inmediatamente después de que los controles del buffer ($\overline{DS1}$ y *DS2*) vuelven a su estado de inhabilitación.

Considerar de nuevo la interfaz del teclado de la Figura 7.14. La técnica de E/S aisladas se utiliza en este sistema. La señal de control especial de lectura de entrada/salida ($\overline{I/O\ R}$) activa la

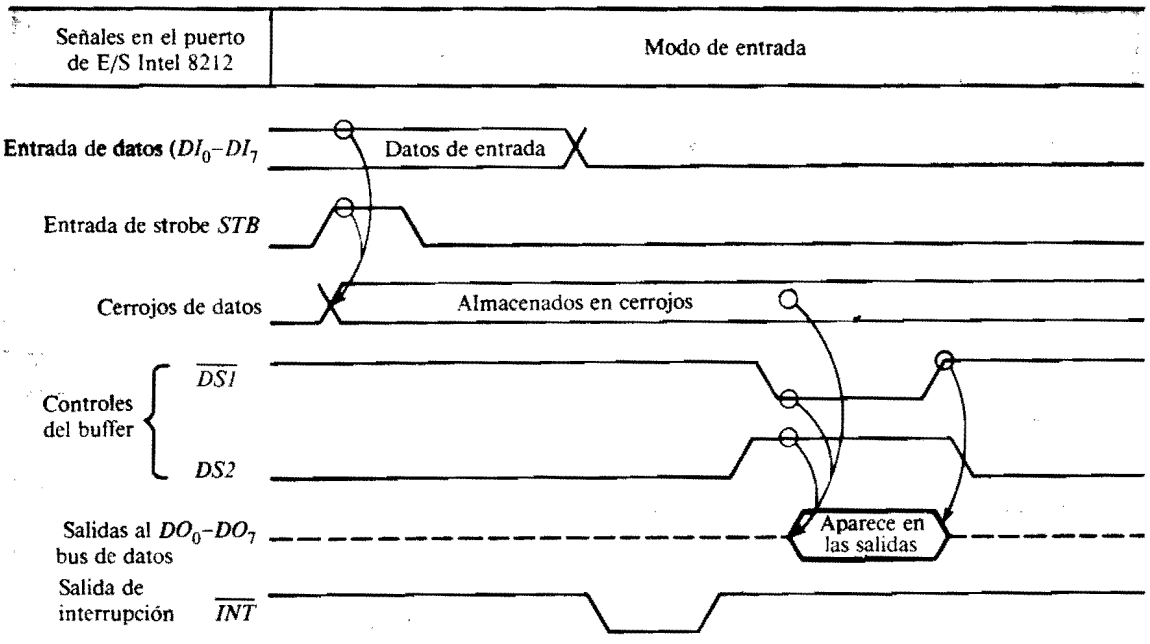


Figura 7.15. Diagrama de tiempos. Señales del puerto de E/S durante el modo de entrada con una sola interrupción.

entrada $\overline{DS1}$ de la unidad 8212. Las 8 líneas de dirección menos significativas (A_0–A_7) son decodificadas completamente por el decodificador de direcciones, cuya salida activa la entrada *DS2* del 8212, vía la línea de selección de dispositivo. Recordar que en este sistema las entradas $\overline{DS1}$ y *DS2* controlan los buffers de salida del puerto de E/S 8212. Observar que se coloca un inversor en la línea de petición de interrupción, este inversor convierte la salida $\overline{INT}$, activa en BAJO, del 8212 en la entrada *INTR*, activa en ALTO, de la MPU.

La interrupción es un método de informar al microprocesador que un dispositivo de E/S está «listo» y de la acción que necesita tomarse. La subrutina especial que realiza esta acción se denomina rutina de servicio de interrupción. Si hay varios dispositivos capaces de interrumpir la MPU, se puede realizar la operación lógica OR con las líneas de petición de interrupción y la CPU tendrá que «encontrar» el dispositivo de E/S que provocó la interrupción. El proceso de «búsqueda» puede involucrar la técnica de escrutinio mencionada anteriormente y se denomina esquema de *escrutinio de interrupciones.* Asociado a cada puerto de E/S en un esquema de escrutinio de interrupción hay un *puerto de status* cuyo código dice a la MPU si el puerto está «listo» para ser introducido o recibir alguna salida.

Algunos microprocesadores tienen varias entradas e instrucciones de interrupción para habilitar e inhabilitar, al menos, algunas de estas entradas de interrupción. Muchos microprocesadores utilizan esquemas *de vectores de interrupción* donde la MPU sabe qué dispositivo de E/S está interrumpiendo y salta a la correspondiente rutina de servicio de interrupción. Muchos microprocesadores tienen también métodos para determinar la prioridad de servicio cuando se reciben a la vez dos interrupciones, lo que se denomina, a veces, *arbitración de la prioridad de interrupciones.* Esta circuitería decide qué dispositivo debe ser atendido primero. Para realizar esta tarea habitualmente se necesita un CI separado, como por ejemplo la unidad de control de prioridad de interrupciones Intel 8259.

Las ventajas del escrutinio sobre las interrupciones son que requiere menos hardware para las interfaces y como está bajo control del programa es síncrona; las desventajas son que requiere excesivo software, gasta tiempo de MPU y, si hay que escrutar muchos dispositivos, se puede tardar bastante tiempo en responder a un dispositivo de E/S que necesite servicio.

Las ventajas de las interrupciones sobre el escrutinio son respuestas más rápidas, mejor uso del tiempo de la MPU y necesidad de menos software. Las desventajas son su operación asíncrona y la necesidad de más hardware para las interfaces.

PROBLEMAS RESUELTOS

7.33. Una petición de interrupción se envía a la MPU cuando un puerto de E/S _____ (no está listo, está listo) para el servicio.

Solución:

Una petición de interrupción es iniciada por un dispositivo de E/S.

7.34. Una petición de interrupción se envía a la MPU cuando un puerto de E/S _____ (no está listo, está listo) para el servicio.

Solución:

Una petición de interrupción se envía a la MPU cuando un puerto de E/S está listo para el servicio.

7.35. Acudir a la Figura 7.14. El puerto de E/S 8212 está operando en su modo de _____ (entrada, salida) en este circuito.

Solución:

El puerto de E/S 8212 de la Figura 7.14 está operando en su modo de entrada en este circuito. El nivel BAJO en la entrada de modo (*MD*) coloca a este puerto de E/S en el modo de entrada.

7.36. Acudir a la Figura 7.14. Para que los datos del teclado sean almacenados en los cerrojos internos de datos de la unidad 8212, la entrada de _____ (borrado, strobe) debe ser un pulso _____ (ALTO, BAJO). Este pulso proviene _____ (del teclado, de la MPU) del sistema.

Solución:

Para que los datos del teclado sean almacenados en los cerrojos internos de datos de la unidad 8212 de la Figura 7.14, la entrada de strobe debe ser un pulso ALTO. Este pulso proviene del teclado del sistema.

7.37. Bajo la ejecución de la instrucción IN (entrar), la MPU de la Figura 7.14 hace que la línea de lectura de E/S vaya al nivel _____ (ALTO, BAJO) y la línea de selección de dispositivo al nivel _____ (ALTO, BAJO). Esto habilita los _____ (cerrojos de datos, buffers de salida) en el puerto de E/S 8212.

Solución:

Bajo la ejecución de la instrucción IN (entrar), la MPU de la Figura 7.14 hace que la línea de lectura de E/S vaya al nivel BAJO y la línea de selección de dispositivo al ALTO. Esto habilita los buffers de salida del puerto de E/S 8212.

7.38. La escrutación a veces se denomina E/S _____ .

Solución:

La escrutación a veces se denomina E/S programadas.

7.39. Listar las ventajas e inconvenientes de las interrupciones sobre el escrutinio como una técnica de sincronización de E/S.

Solución:

Las ventajas de utilizar las interrupciones son que responden rápidamente a las necesidades de los dispositivos de E/S para el servicio, hacen mejor uso del tiempo de la MPU y utilizan menos software. Las desventajas de utilizar interrupciones son su naturaleza asíncrona y la necesidad de más hardware para las interfaces.

7.7. DECODIFICACION DE DIRECCIONES

En la Figura 7.16 se muestra un sencillo sistema basado en microprocesador. Los datos pueden ser introducidos, vía el teclado, y sacados a un simple visualizador de siete segmentos. Se muestran todas las líneas de selección de datos y de control. Los adaptadores de las interfaces de entrada y salida son puertos de E/S Intel 8212. La sincronización de la transferencia de datos desde el adaptador de la interfaz de entrada se realiza utilizando un sencillo interruptor. El sistema utiliza la técnica de E/S de mapa de memoria, tratando así los puertos de E/S como

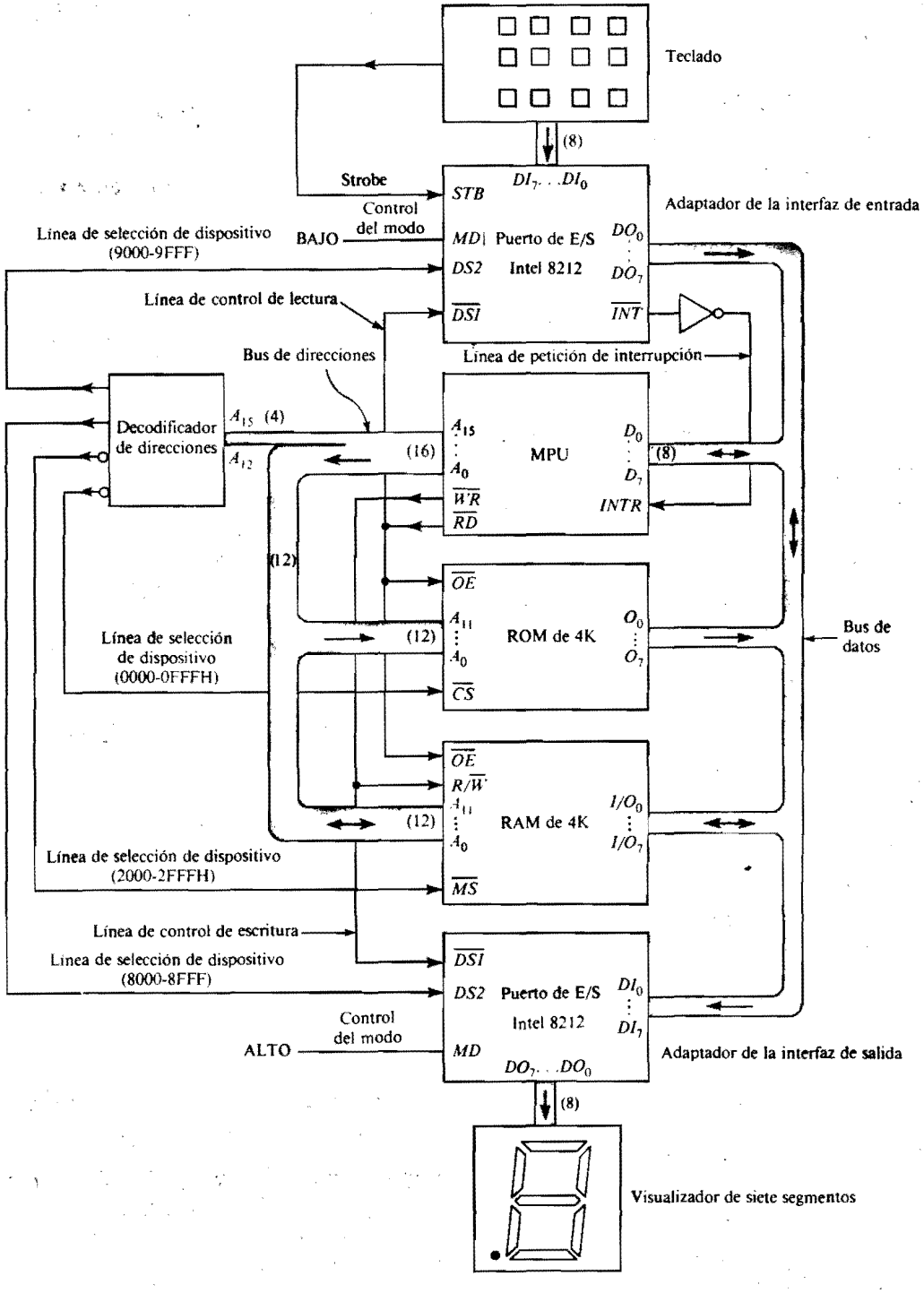


Figura 7.16. Diagrama de un sistema basado en microprocesador.

posiciones de memoria. Cada una de las interfaces individuales ha sido examinada en las secciones anteriores. Esta sección investigará la circuitería de decodificación de direcciones que se debe utilizar en tal sistema.

El diagrama del sistema de la Figura 7.6 muestra las cuatro líneas de dirección más significativas (A_{12}–A_{15}) formando la entrada al decodificador de direcciones. Las tareas del decodificador de direcciones en este sistema son:

1. Generar una salida BAJA para la línea de selección del dispositivo ROM cuando las 4 líneas de dirección más significativas sean 0000_2.
2. Generar una salida BAJA para la línea de selección del dispositivo RAM cuando las 4 líneas de dirección más significativas sean 0010_2.
3. Generar una salida ALTA para la línea de selección del dispositivo del adaptador de la interfaz de salida cuando las 4 líneas de dirección más significativas sean 1000_2.
4. Generar una salida ALTA para la línea de selección del dispositivo del adaptador de la interfaz de entrada cuando las 4 líneas de dirección más significativas sean 1001_2.

Un mapa de memoria del sistema se muestra en la Figura 7.17. Las 4096 posiciones de almacenamiento de la ROM estarán localizadas en el segmento 0 o direcciones 0000 a 0FFFH. El número del segmento en este caso es el equivalente hexadecimal de 0000_2 que genera la señal de selección del dispositivo ROM. En el segmento 2 del mapa de memoria de la Figura 7.17 se localizan otras 4096 direcciones. Estos 4K de memoria son RAM. El puerto de salida está ubicado en las direcciones del segmento 8 del mapa de memoria, mientras que el puerto de entrada utiliza el segmento 9. Observar que cualquiera de las 4096 direcciones desde la 8000 a la 8FFFH activarán el adaptador de la interfaz de salida. Esto es cierto porque sólo se decodifican 4 de las 16 líneas de dirección, lo cual se denomina *decodificación parcial de direcciones* y funciona bien si no se necesita añadir ningún otro dispositivo de salida que tenga direcciones en el segmento 8. De manera análoga, cualquiera de las 4096 direcciones desde la 9000 a la 9FFFH activará el adaptador de la interfaz de entrada de este sistema. De nuevo, esto es cierto porque el bus de direcciones sólo está parcialmente decodificado.

Una solución sencilla al problema de decodificación de direcciones de este sistema se muestra en la Figura 7.18. Para decodificar las 4 líneas de dirección más significativas (A_{12}–A_{15}) se utilizan un CI decodificador 1-de-16 y dos inversores. Recordar que el decodificador (también llamado demultiplexor) actúa como un conmutador rotatorio, activando cada vez sólo una salida (0-15). La salida activada en el decodificador 1-de-16 depende de las líneas de selección que estén habilitadas (S_0–S_3). Además, si las líneas de selección están a 0000, entonces se activa la salida 0 con un nivel BAJO. La Figura 7.18 muestra que la salida 0 del decodificador alimenta la línea de selección del dispositivo de la ROM y selecciona el segmento 0 del mapa de memoria. Si en el decodificador 0010 ($S_3 = 0$, $S_2 = 0$, $S_1 = 1$, $S_0 = 0$), se activa la salida 2 del decodificador 1-de-16 con un nivel BAJO. La salida 2 se utiliza para habilitar la línea de selección de dispositivo de la RAM.

Si las 4 líneas de dirección más significativas de la Figura 7.18 fuesen 1000_2, entonces la salida 8 del decodificador se activaría con un nivel BAJO. Un inversor complementa esta salida para que sea compatible con el pulso ALTO necesario para activar la línea de selección de dispositivo del adaptador de la interfaz de salida. De igual forma un 1001_2 en las entradas de selección del decodificador 1-de-16 habilitarán la línea de selección de dispositivo del adaptador de la interfaz de entrada con un pulso ALTO. Las salidas no utilizadas pueden emplearse para añadir posteriormente más dispositivos periféricos o de memoria.

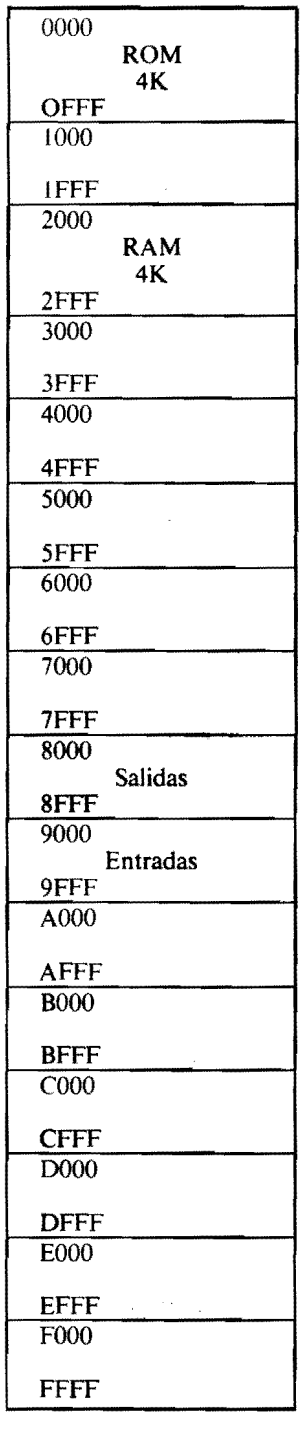


Figura 7.17. Mapa de memoria para el sistema basado en microprocesador.

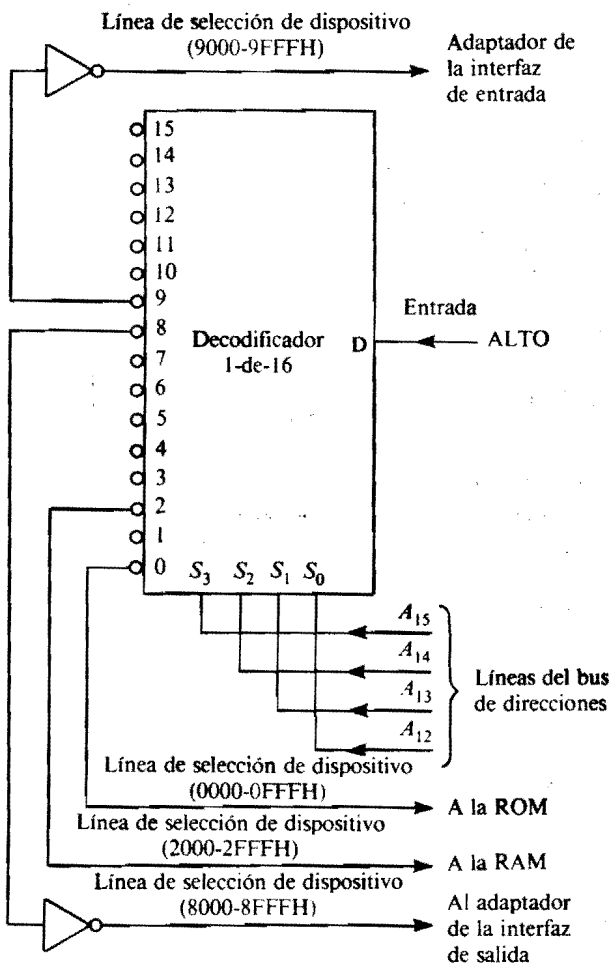
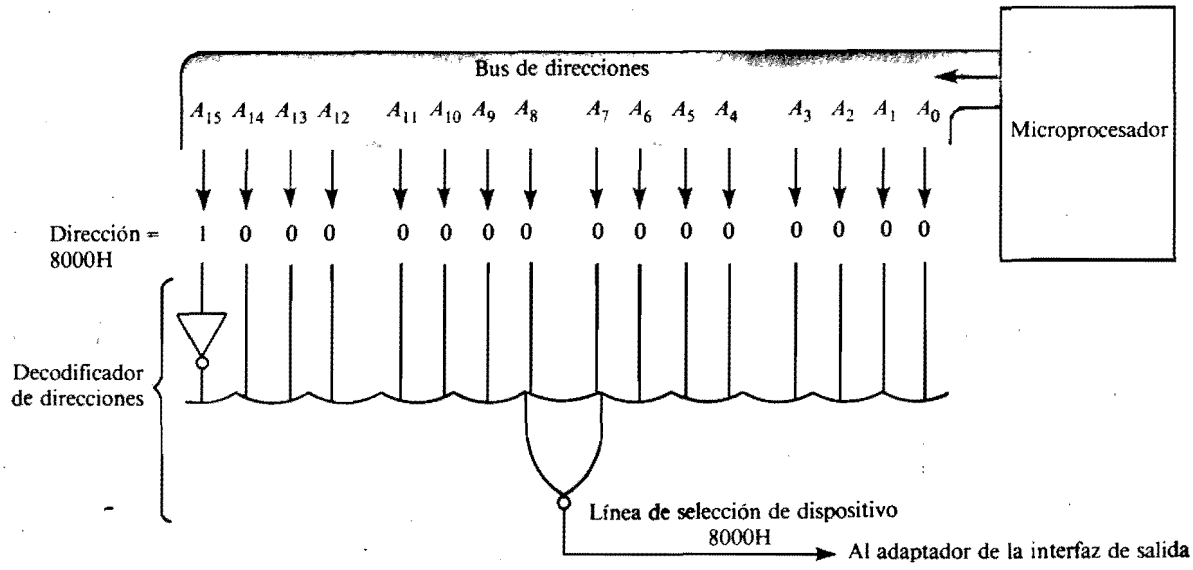


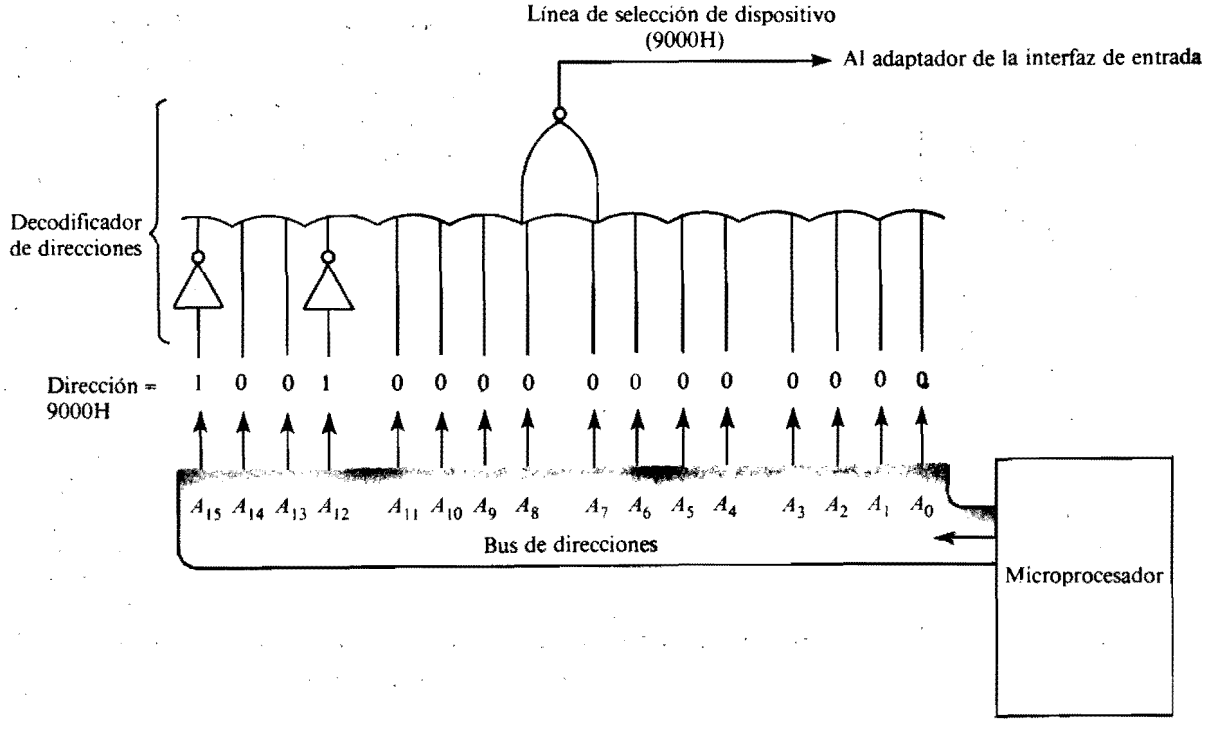
Figura 7.18. Utilización de un decodificador 1 de 16 como decodificador de direcciones.

Las direcciones de la ROM y RAM de la Figura 7.16 están *completamente decodificadas.* La decodificación completa significa que se decodifican las 16 líneas y por tanto se puede acceder a todas las posibles direcciones. Sin embargo, las direcciones del puerto de E/S sólo están parcialmente decodificadas. Por ejemplo la instrucción «LDA, 9000H» debe introducir datos desde el teclado, que está referenciado como la posición de memoria 9000H. Considerar también la instrucción «LDA, 9FFFH». Esta instrucción indica a la CPU que cargue en el acumulador el dato de la posición 9FFFH, que en este caso también será el mismo dato del teclado. Para identificar un *puerto de E/S* con correspondencia en memoria *(mapa de memoria)* con un *único* número hexadecimal, el bus de direcciones de 16 bits debe estar completamente decodificado.

Esquemas completos de decodificación de direcciones para adaptadores de interfaces de entrada y salida se muestran en la Figura 7.19. La Figura 7.19*a* muestra las 16 líneas de dirección que entran al decodificador de direcciones. Solamente cuando la dirección es 8000H la salida de la



(a) Diagrama lógico para decodificar completamente el dispositivo de salida en la dirección 8000H



(b) Diagrama lógico para decodificar completamente el dispositivo de entrada en la dirección 9000H

Figura 7.19.

puerta NOR producirá una señal ALTA de selección de dispositivo al adaptador de interfaz de salida. Todas las demás combinaciones posibles de las 16 entradas pondrán la salida en el nivel BAJO, lo que inhabilita la línea de selección de dispositivo y al adaptador de la interfaz de salida.

La decodificación completa de direcciones para el adaptador de la interfaz de entrada se ilustra en la Figura 7.19*b*. La dirección 9000H es la única que hará que la salida de la puerta NOR vaya al nivel ALTO activando así al adaptador de la interfaz de entrada, vía la línea de selección de dispositivo. Cualquier otra combinación de ceros y unos producirá una salida BAJA en el decodificador de direcciones, lo que inhabilita al adaptador de la interfaz de entrada.

PROBLEMAS RESUELTOS

7.40. Acudir a las Figuras 7.16 y 7.17. Si la MPU coloca 0300H en el bus de direcciones, la (RAM, ROM) se direcciona al activar su línea de selección de dispositivo con un nivel _____ (ALTO, BAJO).

Solución:

Si la MPU en la Figura 7.16 coloca 0300H en el bus de direcciones, la ROM se direcciona al activar su línea de selección de dispositivo con un nivel BAJO.

7.41. Acudir a las Figuras 7.16 y 7.17. Si la MPU coloca 8300H en el bus de direcciones, el adaptador de la interfaz _____ (de entrada, de salida) se direcciona al activar su línea de selección de dispositivo con un nivel _____ (ALTO, BAJO).

Solución:

Si la MPU de la Figura 7.16 coloca 8300H en el bus de direcciones, se direcciona el adaptador de la interfaz de salida al activar su línea de selección de dispositivo con un nivel ALTO.

7.42. Acudir a la Figura 7.16. Las 4096 direcciones de ROM entre 0000H y 0FFFH están _____ (completamente, parcialmente) decodificadas en este sistema.

Solución:

Las 4096 direcciones de ROM entre 0000H y 0FFFH están completamente decodificadas en este sistema. Las 4 líneas de dirección más significativas (A_{12}–A_{15}) son decodificadas por el decodificador de direcciones. Las 12 líneas de dirección menos significativas (A_0–A_{11}) son decodificadas por el decodificador de direcciones en la ROM.

7.43. Acudir a la Figura 7.16. Las 4096 direcciones entre 8000H y 8FFFH están _____ (completamente, parcialmente) decodificadas en este sistema.

Solución:

Las 4096 direcciones en la Figura 7.16 entre 8000H y 8FFFH están parcialmente decodificadas en este sistema. Las 12 líneas de direcciones menos significativas (A_0–A_{11}) no están decodificadas ni por el decodificador de direcciones ni por el dispositivo de salida.

7.44. Acudir a la Figura 7.18. Si las entradas de selección del decodificador 1-de-16 son $S_3 = 1$, $S_2 = 1$ $S_1 = 1$ y $S_0 = 0$, la salida _____ (número) está habilitada y produce un nivel _____ (ALTO, BAJO).

Solución:

Si las entradas de selección del decodificador 1-de-16 son iguales a 1110, la salida 14 está habilitada y produce un nivel BAJO.

7.45. Suponer que al sistema de la Figura 7.16 se le añadiese otra RAM de 4K con las direcciones 1000H a 1FFFH. La salida _____ (número) del decodificador 1-de-16 de la Figura 7.18 podría ser utilizada para excitar la línea de selección de dispositivo de la RAM de 4K añadida.

Solución:

Si se añadiese otra RAM de 4K con las direcciones en el segmento 1 del mapa de memoria, la salida 1 del decodificador de la Figura 7.18 podría utilizarse para excitar la línea de selección de dispositivo del módulo RAM añadido.

PROBLEMAS SUPLEMENTARIOS

7.46. La interconexión, o enlace, de las partes en un sistema basado en microprocesador se denomina _____.

Res. interfaz.

7.47. Una interfaz es una frontera compartida entre dos dispositivos que comparten _____ (información, memoria).

Res. información.

7.48. Otras consideraciones sobre la realización de interfaces incluyen la dirección de la transmisión de datos y la _____ (formación de matrices, sincronización).

Res. sincronización.

7.49. Las tres interconexiones o caminos de datos en un sistema basado en microprocesador incluyen el bus de datos, el bus de _____ y el bus de _____.

Res. direcciones, control.

7.50. Cuando una MPU deja el control de los buses de datos y direcciones para que un dispositivo periférico pueda acceder a la memoria principal del sistema sin ir a través de la CPU, se denomina _____ _____ _____ o DMA.

Res. acceso directo de memoria.

7.51. Acudir a la Figura 7.2. El bus de direcciones está _____ (completamente, parcialmente) decodificado en este sistema.

Res. completamente.

7.52. Acudir a la Figura 7.2. Después de direccionar, un nivel _____ (ALTO, BAJO) en las entradas $\overline{CS}$ y $\overline{OE}$ de la ROM harán que se ejecute una operación de _____ (lectura, escritura) de memoria.

Res. BAJO, lectura.

7.53. Acudir a la Figura 7.4. Los pequeños círculos y flechas significan una relación de causa-_____ en los diagramas de tiempo.

Res. efecto.

7.54. Acudir a la Figura 7.2. Las salidas de la ROM (O_0–O_7) están en el estado _____ (ALTO, de alta impedancia) cuando no se accede a la unidad de almacenamiento.

Res. alta impedancia.

7.55. Si la línea de control de escritura de la Figura 7.5 está en el nivel BAJO, la RAM está en el modo de _____ (lectura, escritura).

Res. escritura.

7.56. La línea de control de la Figura 7.5 habilita la entrada _____ de la RAM con un nivel _____ (ALTO, BAJO).

Res. $\overline{OE}$ (habilitación de salida), BAJO.

7.57. Durante la operación de lectura en la Figura 7.5, las patillas D_0 a D_7 del microprocesador están consideradas como _____ (entradas, salidas) y están _____ (recibiendo datos desde, enviando datos a) el bus de datos.

Res. entradas, recibiendo datos desde.

7.58. Las instrucciones especiales IN (entrar) y OUT (sacar) son utilizadas si se emplea la técnica de E/S _____ (aisladas, de mapa de memoria).

Res. aisladas.

7.59. El adaptador de la interfaz de salida de la Figura 7.10*b* contiene una puerta y un _____ de 8 bits para almacenar los datos de salida.

Res. cerrojo (o cerrojo de datos).

7.60. El dispositivo triangular dentro del adaptador de la interfaz de entrada de la Figura 7.11*a* es un _____ de salida.

Res. buffer (o buffer de tres estados).

7.61. El sistema de la Figura 7.12 utiliza la técnica de E/S _____ (aisladas, de mapa de memoria).

Res. aisladas.

7.62. Los buffers de salida del 8212 están _____ (habilitados por $\overline{DSI}$, permanentemente habilitados) en el sistema de la Figura 7.12.

Res. permanentemente habilitados.

7.63. El sistema de la Figura 7.14 utiliza una _____ para notificar al microprocesador cuándo están listos los datos para ser transferidos a la CPU.

Res. interrupción (o petición de interrupción).

7.64. En la Figura 7.14 se genera una interrupción por _____ (el adaptador de la interfaz de entrada, la MPU) _____ (antes, después) que la entrada del strobe del 8212 almacene los datos en los cerrojos de datos.

Res. adaptador de la interfaz de entrada, después.

7.65. Después de recibir la petición de interrupción de la Figura 7.14, la MPU completa la instrucción actual, almacena los contenidos del contador de programa y del registro actual en la pila y salta a un corto programa llamado _____ _____ _____ .

Res. rutina de servicio de interrupción.

7.66. _____ es una técnica por la cual cada dispositivo de E/S es periódicamente muestreado para determinar si está «listo» y necesita servicio.

Res. escrutinio.

7.67. Acudir a la Figura 7.16. El dispositivo periférico de entrada de este sistema es el _____.

Res. teclado.

7.68. Acudir a la Figura 7.16. _____ (el adaptador de la interfaz de salida, la RAM) almacena los datos de salida y los mantiene en las entradas del visualizador de siete segmentos.

Res. adaptador de la interfaz de salida.

7.69. Acudir a la Figura 7.16. La sincronización de la transferencia de datos del adaptador de la interfaz de entrada en este sistema se realiza utilizando un esquema de _____ (escrutinio de interrupciones, simple interrupción).

Res. simple interrupción.

7.70. Acudir a la Figura 7.16. Las direcciones de la ROM y RAM están _____ (completamente, parcialmente) decodificadas en este sistema.

Res. completamente.

7.71. Acudir a la Figura 7.16. Las direcciones del puerto de entrada y salida están _____ (completamente, parcialmente) decodificadas en este sistema.

Res. parcialmente.

Capítulo 8

EL MICROPROCESADOR INTEL 8080/8085

8.1. INTRODUCCION

El primer microprocesador fue introducido por Intel Corporation en 1971. Intel introdujo ese año el 4004 de 4 bits y el 8008 de 8 bits. En 1974, Intel introdujo el microprocesador 8080, que maneja palabras de datos de 8 bits y tiene 16 líneas de dirección y un puntero de pila de 16 bits. El microprocesador Intel 8085 es una versión mejorada del 8080. Integra el reloj, control del sistema y prioridad de las interrupciones en el CI microprocesador, reduciendo así el número de CI utilizados en la mayoría de los sistemas. El 8085 también opera con una sencilla fuente de alimentación de +5 V. La MPU 8085 utiliza las mismas instrucciones que el 8080, haciendo así los programas compatibles. El microprocesador 8085 tiene dos instrucciones más para aprovechar las características del hardware añadido.

El microprocesador genérico introducido en los capítulos anteriores fue una versión simplificada de la MPU del 8080/8085. Una vez familiarizado con el microprocesador genérico, la mayoría de los estudiantes tendrán poca dificultad para aprender la MPU del 8085. La MPU 8085 tiene más registros internos, muchas más posibles instrucciones, más señales de control y algunas características diferentes cuando se compara con el microprocesador genérico.

Una tendencia en la evolución de los microprocesadores ha sido integrar más funciones en menos CI. Un sencillo sistema basado en microprocesador, hace poco tiempo, habría necesitado más de 20 CI. El sistema mostrado en la Figura 8.1 solamente contiene tres CI. La MPU es el Intel 8085 que controla el bus del sistema y dos CI especiales de interfaz de periféricos.

Las pastillas de interfaz de periféricos, diseñadas especialmente, que se muestran en forma de bloque en la Figura 8.1 son los CI 8155 y 8355. El 8155 contiene 2048 bits de RAM estática, organizada como una memoria de 256 × 8 bits. Esta unidad también contiene tres puertos de E/S y un temporizador. Dos de las E/S son puertos de propósito general de 8 bits. La unidad restante de 6 bits puede ser utilizada como puerto de entrada, como puerto de salida o como camino de las señales de control para los dos puertos de 8 bits. El 8155 es programable y también contiene un registro de status y un contador temporizador de 14 bits.

La otra pastilla mostrada en el diagrama de bloques de la Figura 8.1 es el CI periférico de interfaz 8355. El 8355 contiene 16.384 bits de ROM y está organizado como una memoria de 2048 × 8 bits. El 8355 también contiene dos puertos de E/S de propósito general de 8 bits.

El 8080/8085 y sus parientes de 16 y 32 bits se han convertido en algunos de los microprocesadores de propósito general más ampliamente utilizados en el mundo. Entre los parientes del microprocesador 8080/8085 están los Intel 8086, 8088, 80186, 80188, 80286 y 80386. Algunos de estos circuitos integrados fueron seleccionados para utilizarlos como CPU de las populares microcomputadoras IBM.

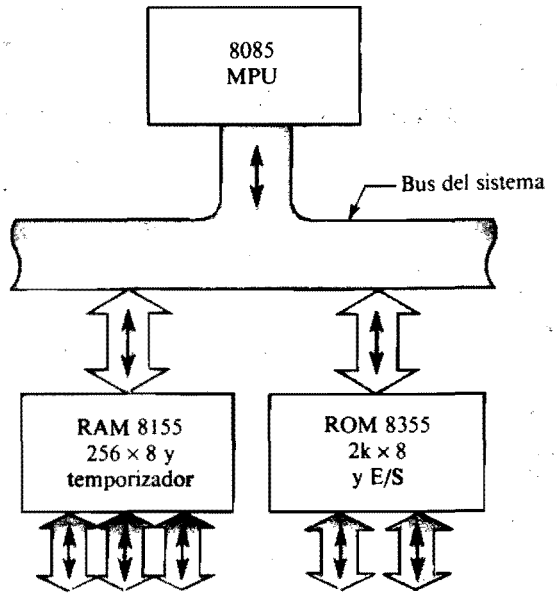


Figura 8.1. Diagrama de bloques del sencillo sistema 8085 que utiliza pastillas especiales de interfaz.

8.2. FUNCIONES Y DIAGRAMA DE PATILLAS DEL 8085

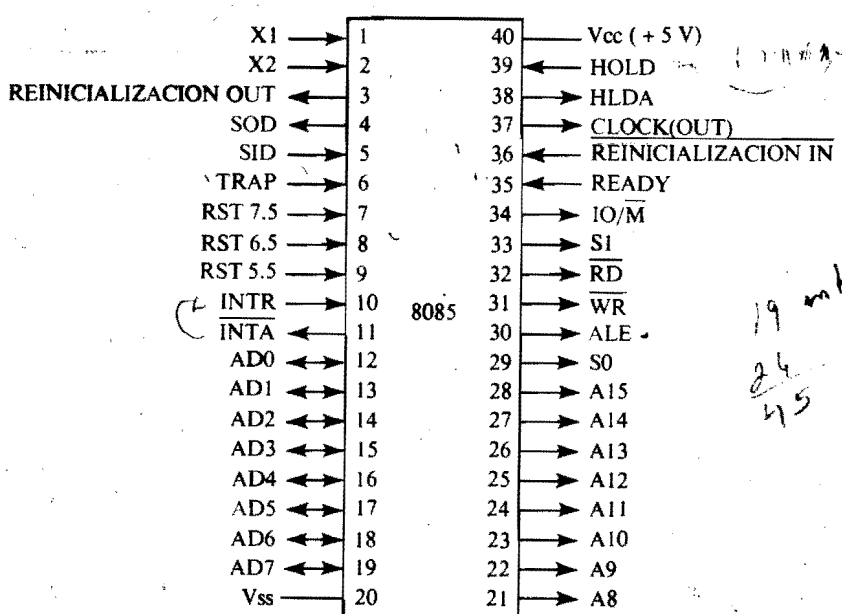
El microprocesador de 8 bits Intel 8085 está ubicado en un encapsulamiento de doble línea (DIP) de 40 patillas. El *diagrama de patillas* de la MPU 8085 se reproduce en la Figura 8.2*a*. En la Figura 8.2*b* hay una tabla donde se resumen los nombres y descripciones de las patillas. La información que da el fabricante puede especificar un microprocesador 8085 u 8085A. La unidad 8085A es una versión ligeramente revisada del microprocesador original 8085. Versiones posteriores incluyen dispositivos de más alta velocidad como el 8085A-2, 8085AH, 8085AH-2 y 8085AH-1.

El microprocesador genérico utilizaba 16 patillas para las líneas de dirección y 8 patillas para las conexiones del bus de datos. Con las funciones añadidas del 8085, el DIP de 40 patillas no tiene patillas suficientes para todas las entradas y salidas. Por esa razón, el fabricante utiliza las patillas 12 a 19 como líneas de bus de propósito doble de direcciones/datos (AD_0–AD_7). Se dice que esta unidad tiene un bus *multiplexado de datos/direcciones* de 8 bits. Las 8 líneas de dirección menos significativas comparten las patillas con las 8 líneas del bus de datos. Multiplexar significa primero seleccionar una y después otra, etc. Por tanto, multiplexar el bus de direcciones/datos significa primero utilizar el bus para enviar una dirección y a continuación enviar o recibir datos vía el mismo bus. El 8085 tiene una señal especial para informar a los periféricos cuando el bus de direcciones/datos envía una dirección y cuando funciona como bus de datos. La señal especial se denomina señal de control *de habilitación del cerrojo de direcciones* (*ALE*). Observar que las patillas del bus de direcciones/datos son bidireccionales o pueden ser de tres estados o alta impedancia (también llamadas triestados). La patilla *ALE* es una salida de control.

El 8085, igual que el microprocesador genérico, tiene un total de 16 líneas de dirección. Las 8 líneas de dirección más significativas están conectadas a las patillas A_8 a A_{15} en la Figura 8.2. Estas se conectan directamente al bus de direcciones como en el microprocesador genérico. Estas patillas son salidas o pueden ser de tres estados. Otras patillas similares a las encontradas en el microprocesador genérico son las de alimentación (V_{cc} y V_{ss}) conectadas a una fuente de alimentación de +5 V. El microprocesador 8085 tiene circuitería de reloj permanente sin más que conectar las patillas de entrada X_1 y X_2 a un cristal. La frecuencia interna de la MPU será la mitad de la frecuencia del cristal.

Algunas de las patillas del 8085 listadas en la Figura 8.2 realizan funciones de control similares a las observadas en el microprocesador genérico. Las salidas de control de lectura ($\overline{RD}$) y escritura ($\overline{WR}$) son utilizadas para informar a la memoria o dispositivos de E/S cuando hay que enviar o recibir datos vía el bus de datos (realmente por el bus de direcciones/datos en esta unidad). La entrada de reinicialización ($\overline{REINICIALIZACION\ IN}$) actúa como la reinicialización de la MPU genérica, ya que el contador de programa se pone, al reinicializar, a 0000H. Los buses de datos y direcciones y líneas de control están en el estado de tres estados durante la reinicialización. El contenido de los registros internos también puede ser alterado durante una reinicialización. La patilla *REINICIALIZACION OUT* está asociada con la operación de reinicialización. Cuando la MPU se está reinicializando, la patilla *REINICIALIZACION OUT* envía una señal a los periféricos que les informa que el sistema se va a reinicializar.

La salida de reloj (*CLK*) del microprocesador 8085 de la Figura 8.2 opera igual que la de la MPU genérica. La entrada de petición de interrupción (*INTR*) al 8085 es una interrupción de



(a) Diagrama de patillas de la MPU 8085

Figura 8.2.

Nombre de patilla	Descripción	Tipo
AD0—AD7	Bus de direcciones/datos	Bidireccional, tres estados
A8—A15	Bus de direcciones	Salida, tres estados
ALE	Habilitación del cerrojo de dirección	Salida*
$\overline{RD}$	Control de lectura	Salida, tres estados
$\overline{WR}$	Control de escritura	Salida, tres estados
IO/$\overline{M}$	E/S o indicador de memoria	Salida, tres estados
S0, S1	Indicadores del estado del bus	Salida
READY	Petición de estado de espera	Entrada
SID	Entrada de datos serie	Entrada
SOD	Salida de datos serie	Salida
HOLD	Petición de mantenimiento	Entrada
HLDA	Conocimiento de mantenimiento	Salida
INTR	Petición de interrupción	Entrada
TRAP	Petición de interrupción no enmascarable	Entrada
RST 5.5 RST 6.5 RST 7.5	Peticiones de interrupción hardware de vectorizadas	Entrada Entrada Entrada
$\overline{INTA}$	Conocimiento de interrupción	Salida
REINICIALIZACION IN	Reinicialización del sistema	Entrada
REINICIALIZACION OUT	Reinicialización de los periféricos	Salida
X1, X2	Conexiones al cristal o RC	Entrada
CLK	Señal de reloj	Salida
Vcc, Vss	Alimentación, tierra	

* Esta salida es de tres estados en el 8085, pero no en el 8085A.

(b) Nombres y funciones de las patillas

Figura 8.2. *(continuación).*

propósito general como en el microprocesador genérico. Sin embargo, de forma distinta a la interrupción del microprocesador genérico, la *INTR* del 8085 puede ser habilitada o inhabilitada por instrucciones software. Además de la petición regular de interrupciones (*INTR*), la MPU 8085 tiene otras cuatro entradas de interrupción. Estas son las patillas de las entradas *TRAP, RST 7.5, RST 6.5* y *RST 5.5*. *TRAP* es la interrupción de más alta prioridad, seguida por *RST 7.5, RST 6.5, RST 5.5* y finalmente por la interrupción de más baja prioridad, *INTR*. *TRAP* o cualquiera de los tres rearranques (*RST 7.5, RST 6.5, RST 5.5*) hacen que la MPU salte a una subrutina en una dirección específica de llamada. Las interrupciones de rearranque pueden ser habilitadas e inhabilitadas por software, mientras que la interrupción *TRAP* no puede ser inhabilitada de esta forma. La interrupción *INTR* salta a una dirección dictada por una instrucción especial recibida, de un dispositivo periférico, cuando es activada la salida de conocimiento de interrupción ($\overline{INTA}$) de la MPU.

El microprocesador 8085 tiene una entrada serie y una salida serie primitivas. Estas son las patillas *SID* (entrada de datos serie) y *SOD* (salida de datos serie) mostradas en la Figura 8.2. El

bit de datos de la patilla de entrada *SID* es cargado en la posición *MSB* (bit 7) del acumulador por la instrucción RIM del 8085. La patilla de salida *SOD* se pone en set (1) o reinicialización (0) según especifique la instrucción SIM del 8085.

Considerar la entrada *READY* al microprocesador 8085 de la Figura 8.2. Esta entrada proveniente de un dispositivo periférico informa al microprocesador que está listo para enviar o recibir datos. Si la entrada *READY* del dispositivo periférico está en el nivel BAJO, durante un ciclo de lectura o escritura, la MPU lo interpreta como una petición para entrar en el *estado de espera,* donde permanecerá hasta que las señales periféricas estén listas para transmitir o recibir datos. La MPU entonces procederá con el ciclo de lectura o escritura. La entrada *READY* es manual cuando se utilizan memorias o periféricos que son demasiado lentos comparados con la velocidad de tratamiento del microprocesador.

Considerar las patillas de entrada *HOLD* (mantenimiento) y de salida *HLDA* (conocimiento de mantenimiento) de la MPU 8085 de la Figura 8.2. La entrada *HOLD* notifica a la MPU que otro dispositivo quiere utilizar los buses de direcciones y datos. Esto puede ocurrir durante operaciones de acceso directo a memoria. Después de recibir una entrada *HOLD*, la CPU completará las transferencias actuales de datos en los buses. Entonces las patillas de direcciones, datos, $\overline{RD}$, $\overline{WR}$ e $IO/\overline{M}$ del 8085 se ponen en alta impedancia para no interferir con las transferencias de datos en los buses. Una salida *HLDA* indica a un periférico que se ha recibido una petición de mantenimiento y que el microprocesador abandonará el control de los buses en el siguiente ciclo de reloj.

Las salidas $IO/\overline{M}$, S_0 y S_1 son señales de control que notifican a los periféricos el tipo de ciclo máquina que está realizando la MPU. Los tipos de ciclo máquina aparecen en la columna derecha de la Figura 8.3. La combinación adecuada de las señales de salida de las patillas $IO/\overline{M}$, S_0 y S_1 se detalla a la izquierda.

Después de esta breve visión general de las funciones de las patillas, se observa que la MPU 8085 tiene algunas características que no aparecen en el microprocesador genérico. La mayoría de estas características y algunos de sus usos se explorarán en las secciones posteriores.

Señales de control del 8085			
$IO/\overline{M}$	S_1	S_0	Status de ciclos de máquina
0	0	1	Escritura en memoria
0	1	0	Lectura de memoria
1	0	1	Escritura de E/S
1	1	0	Lectura de E/S
0	1	1	Búsqueda del código de op
1	1	1	Conocimiento de interrupción
*	0	0	Alto
*	x	x	Mantenimiento
*	x	x	Reinicialización

* = condición de tres estados.
x = no especificado.

Figura 8.3. Ciclos máquina del 8085.

PROBLEMAS RESUELTOS

8.1. El Intel 8085 se denomina un microprocesador de _____ (número) bits porque las palabras del acumulador tienen esta cantidad de bits.

Solución:

El Intel 8085 se denomina un microprocesador de 8 bits porque las palabras del acumulador tienen este número de bits.

8.2. El bus de direcciones/datos se dice que es _____ (decodificado, multiplexado) porque alterna funciones como bus de direcciones y como bus de datos.

Solución:

El bus de direcciones/datos se dice que es multiplexado porque alterna funciones como bus de direcciones y bus de datos.

8.3. Acudir a la Figura 8.2. La salida _____ la utiliza el microprocesador para notificar a los periféricos cuándo se utiliza como bus de direcciones el bus de direcciones/datos.

Solución:

La salida *ALE* (habilitación del cerrojo de direcciones) la utiliza el microprocesador 8085 para notificar a los periféricos cuándo se utiliza como bus de direcciones el bus de direcciones/datos.

8.4. Acudir a la Figura 8.2. El microprocesador 8085 utiliza una fuente de alimentación de _____ V.

Solución:

De acuerdo con la Figura 8.2, el microprocesador 8085 utiliza una fuente de alimentación de +5 V.

8.5. Acudir a la Figura 8.2. Si las dos salidas $IO/\overline{M}$ y $\overline{WR}$ estuviesen en BAJA, el microprocesador estaría _____ (leyendo, escribiendo) en un(a) _____ (puerto de E/S, posición de memoria).

Solución:

Si las dos salidas $IO/\overline{M}$ y $\overline{WR}$ estuviesen en BAJA, la MPU estaría escribiendo en una posición de memoria. Las barras superiores indican que un nivel BAJO activa la M, o memoria, en la señal $IO/\overline{M}$ y una escritura en la señal de salida $\overline{WR}$.

8.6. Acudir a la Figura 8.2. Si el microprocesador estuviese en un ciclo de lectura y la entrada *READY* estuviese en el nivel BAJO, la CPU _____ (terminaría, iría al estado de espera hasta que el dispositivo estuviese listo y entonces terminaría) inmediatamente el ciclo de lectura.

Solución:

Si la MPU estuviese en un ciclo de lectura y la entrada *READY* estuviese en el nivel BAJO, la CPU iría al estado de espera hasta que el dispositivo periférico estuviese listo y entonces finalizaría el ciclo de lectura.

8.7. Acudir a la Figura 8.3. Durante un ciclo de máquina de búsqueda de código de op, las señales de control S_0 = _____ (0, 1), S_1 = _____ (0, 1) e $IO/\overline{M}$ = _____ (0, 1).

Solución:

De acuerdo con la Figura 8.3, durante un ciclo máquina de búsqueda de código de op, las señales de control $S_0 = 1$, $S_1 = 1$ e $IO/\overline{M} = 0$.

8.8. Acudir a la Figura 8.2. Listar las patillas de entrada en el CI microprocesador 8085 que pueden considerarse interrupciones.

Solución:

Las patillas de entradas de la MPU 8085 que se consideran interrupciones son la *INTR, TRAP, RST 7.5, RST 6.5* y *RST 5.5*. Algunas pueden considerar también la entrada de *REINICIALIZACION IN* (reinicialización) como una interrupción porque está provocada por un dispositivo externo y el contador de programa bifurca a un nuevo segmento de programa.

8.9. Acudir a la Figura 8.2. Las patillas de E/S serie están etiquetadas _____ y _____ en el diagrama de patillas del microprocesador 8085.

Solución:

De acuerdo con la Figura 8.2, las patillas de E/S serie están etiquetadas *SID* y *SOD* en el diagrama de patillas del microprocesador 8085.

8.3. ARQUITECTURA DEL 8085

La organización interna o arquitectura del microprocesador Intel 8085 se reproduce en la Figura 8.4. El 8085 tiene un contador de programa y un cerrojo de direcciones de 16 bits que alimenta al bus de direcciones dedicado (A_{15}–A_8) y al bus de direcciones/datos de doble propósito (AD_7–AD_0). Los datos paralelos entran y dejan la MPU vía el bus de direcciones/datos multiplexado (AD_7–AD_0). El bus de direcciones/datos transmite una dirección cuando la línea de control *ALE* está en ALTA y datos cuando la línea *ALE* está en BAJA.

Las entradas o salidas de datos a la unidad se realizan a través del bus de datos interno de 8 bits. Los datos pueden fluir desde el bus de datos interno al acumulador de 8 bits o a los registros temporales, señalizadores, registro de instrucción, unidad de control de interrupción, unidad de control de E/S serie, cualquiera de los registros de propósito general (*B, C, D, E, H* y *L*), puntero de pila de 16 bits, contador de programa de 16 bits, o buffer de datos/direcciones de 8 bits. Las entradas y salidas serie (*SID* y *SOD*) se muestran en la parte superior derecha de la Figura 8.4. Las entradas de interrupción (*INTR, RST 5.5, RST 6.5, RST 7.5* y *TRAP*) y la salida $\overline{INTA}$ (conocimiento de interrupción) se muestran en la parte superior izquierda. La unidad aritmético-lógica (ALU) está alimentada por dos registros de 8 bits (registro acumulador y registro temporal) como en el microprocesador genérico. Los flip-flops señalizadores tienen cinco indicadores de status en lugar de dos como en el microprocesador genérico.

El registro de instrucción alimenta al decodificador de instrucciones, el cual interpreta la instrucción actual y determina el microprograma que debe seguir. El decodificador de instrucciones entonces instruye a las secciones de control y temporización sobre la secuencia de eventos que deben realizarse. La sección de temporización y control coordina las acciones de procesador y los periféricos. Las líneas de control interno no se muestran en la Figura 8.4, pero las salidas de control y de status sí. Las entradas de *REINICIALIZACION, HOLD* y *READY* también se muestran como entradas de la sección de control y temporización del microprocesador 8085.

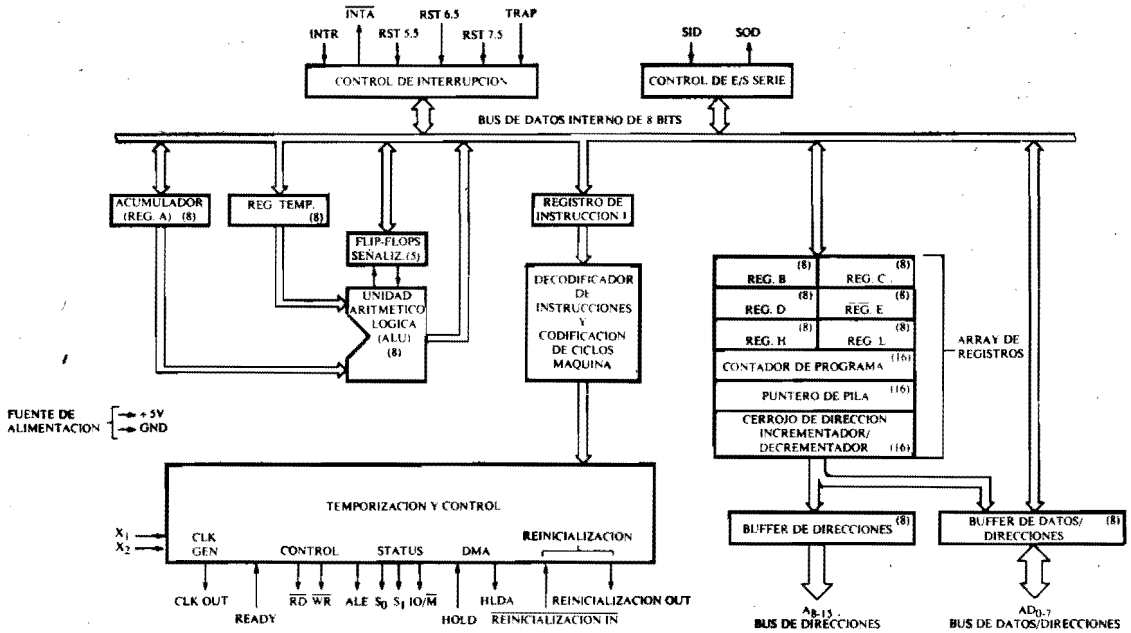


Figura 8.4. Diagrama funcional de bloques de la MPU 8085. (*Cortesía de Intel Corporation.*)

Registros

Igual que el microprocesador genérico, la MPU 8085 utiliza registros de 8 y 16 bits. El 8085 tiene 8 registros direccionables de 8 bits. Seis de estos registros pueden utilizarse como registros de 8 bits o como registros pares de 16 bits. Además, el 8085 contiene dos registros más de 16 bits. Los registros del 8085 son los siguientes:

1. El *acumulador* (registro *A*) es el foco de todas las operaciones del acumulador que incluyen instrucciones aritméticas, lógicas, de carga y almacenamiento y de E/S. Es un registro de 8 bits.
2. Los registros de *propósito general BC, DE* y *HL* pueden ser utilizados como seis registros de 8 bits o como tres registros de 16 bits dependiendo de la instrucción que se vaya a ejecutar. Como en el microprocesador genérico, el registro par *HL* llamado *puntero de datos* por Intel) puede ser utilizado para apuntar a direcciones. Pocas instrucciones utilizan los registros *BC* y *DE* como punteros de dirección, ya que normalmente se utilizan como registros de datos de propósito general.
3. El *contador de programa* (*PC*) siempre señala la posición de memoria de la siguiente instrucción que se va a ejecutar. Siempre contiene una dirección de 16 bits.
4. El *puntero de pila* (*SP*) es un puntero de dirección de propósito especial (o puntero de dato) que siempre apunta a la parte superior de la pila de la RAM. Es un registro de 16 bits.
5. El *registro de señalizadores* contiene cinco señalizadores de 1 bit que contienen información del estado de la CPU. Estos señalizadores son utilizados entonces por instrucciones de bifurcación condicional, llamada y vuelta de subrutina.

Señalizadores

Los cinco señalizadores encontrados en la CPU del 8085 están representados en la Figura 8.5. El *señalizador de arrastre* (*CY*) es puesto a 1 ó 0 por operaciones aritméticas, y su status es examinado por las instrucciones del programa. Como en el microprocesador genérico, un «overflow» (rebose) de una suma de 8 bits hará que el bit *CY* se ponga a 1. En la resta el señalizador *CY* actúa como un señalizador de «préstamo», indicando que el minuendo es menor que el sustraendo si el señalizador está a 1.

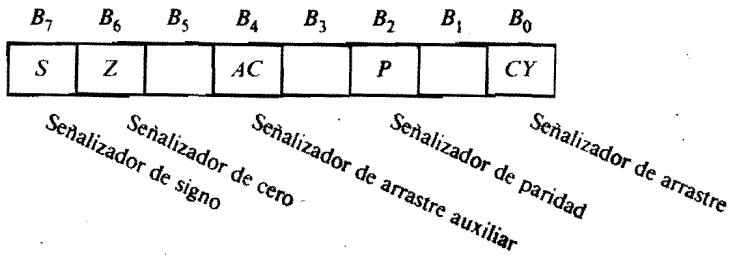


Figura 8.5. Señalizadores del registro de status del 8085.

El *señalizador de cero* (*Z*) se pone a 1 si el resultado de ciertas instrucciones es cero. El señalizador de cero está a 0 si el resultado no es cero. La acción del señalizador de cero se observó en el microprocesador genérico.

El *señalizador de signo* (*S*) indica la condición del bit más significativo del acumulador después de la ejecución de instrucciones aritmético-lógicas. Estas instrucciones utilizan el MSB del dato para representar el signo del número contenido en el acumulador. Un señalizador de signo a 1 representa un número negativo, mientras que a 0 significa un número positivo.

El *señalizador de arrastre auxiliar* (*AC*) indica un «overflow» o arrastre del bit 3 del acumulador de la misma forma que el señalizador de arrastre indica un «overflow» del bit 7. Este señalizador es utilizado comúnmente en aritmética BCD (decimal codificado binario).

El *señalizador de paridad* (*P*) examina el número de bits 1 del acumulador. Si éste contiene un *número par* de 1, se dice que existe paridad par y el señalizador de paridad se pone a 1. Sin embargo, si el acumulador contiene un número impar de 1 (paridad impar), el señalizador de paridad del 8085 se pone a 0. Por ejemplo, si una instrucción ADD da como resultado la suma 00110011_2 en el acumulador, el señalizador de paridad estará a 1 porque hay un número par de 1 (4). Sin embargo, si la suma fuese 10101110_2, el señalizador *P* estaría a 0 porque hay un número impar (5) de 1 en el acumulador.

Puntero de pila

El *puntero de pila* contiene la dirección del último byte introducido en la pila. El puntero de pila puede ser inicializado para utilizar cualquier parte de la RAM como pila. Como en el microprocesador genérico, el puntero de pila se decrementa cada vez que se introduce un dato en la pila y se incrementa cada vez que se saca el dato de la pila.

Unidad aritmético-lógica

La ALU está muy relacionada con el acumulador, registro de señalizadores y algunos registros temporales que son inaccesibles al programador. Las operaciones aritméticas, lógicas y de desplazamiento circular son realizadas en la ALU. Los resultados normalmente se depositan en el acumulador.

Registro de instrucción y decodificador

Durante una búsqueda de instrucción, el primer byte de la instrucción, el código de op, es transferido al *registro de instrucción* de 8 bits. El contenido del registro de instrucción está disponible al *decodificador de instrucciones*. La salida del decodificador, a través de puertas y con señales de temporización, controla los registros, ALU y buffers de datos y direcciones. Las salidas del decodificador de instrucciones y del generador de reloj interno producen las señales de temporización de ciclo y estado de la máquina.

Generador interno de reloj

La MPU 8085 incorpora un generador completo de reloj en su pastilla. Requiere solamente la adición de un cristal de cuarzo para establecer la temporización de su operación. El 8085A puede utilizar un cristal de hasta 6.25 megahertzios (MHz). La versión 8085A-2 del 8085 opera con un cristal de hasta 10 MHz. La patilla de salida *CLK* es una salida de reloj con buffer que tiene una frecuencia una vez y media la frecuencia del cristal. Las entradas de reloj del 8085AH, 8085AH-2 y 8085AH-1 también pueden ser excitadas con cristales de 6, 10 ó 12 MHz.

Interrupciones

Las cinco entradas de *interrupción hardware* están listadas por prioridades en la columna izquierda de la Figura 8.6. La interrupción de más alta prioridad es la de la entrada *TRAP*. Una vez que alcanza el nivel ALTO, la entrada *TRAP* hará que el 8085 guarde en la pila el contenido del contador de programa y salte a la posición de memoria 0024H. La entrada *TRAP* no puede ser inhabilitada y por tanto se denomina *interrupción no enmascarable*.

Las tres siguientes interrupciones hardware listadas en la Figura 8.6 se denominan reinicializadoras (para reinicializar el programa en una nueva posición de memoria). La interrupción *RST 7.5* hace que el 8085 guarde en la pila el contenido del contador de programa y salte a la posición de memoria 003CH. La siguiente interrupción de prioridad más elevada es la *RST 6.5*, que hace que el 8085 guarde en la pila el contenido del contador de programa y salte a la posición de memoria 0034H. La interrupción de prioridad más baja *RST 5.5* hace que el 8085 guarde en la pila el contenido del contador de programa y salte a la posición de memoria 002CH.

La interrupción hardware de más baja prioridad es la entrada *INTR*, que hace que el procesador busque una instrucción desde una fuente externa especial. Las cuatro últimas interrupciones pueden ser habilitadas o inhabilitadas por software.

Entrada y salida serie

Las patillas de *entrada y salida serie* del 8085 ayudan a minimizar la cuenta de pastillas en los pequeños sistemas al proporcionar un puerto de interfaz serie. La instrucción especial RIM del 8085 transfiere datos desde la patilla de entrada serie (*SID*) al bit 7 del acumulador. Esto se ilustra en el ejemplo de la Figura 8.7*a*, donde se transfiere un nivel ALTO desde la entrada *SID* a la posición MSB del acumulador.

Nombre	Prioridad	Dirección (1) a la que se bifurca cuando ocurre la interrupción	Tipo de disparo
TRAP	1	24H	Flanco de subida AND nivel alto hasta que sea muestreada
RST 7.5	2	3CH	Flanco de subida (cerrojo)
RST 6.5	3	34H	Nivel alto hasta que sea muestreada
RST 5.5	4	2CH	Nivel alto hasta que sea muestreada
INTR	5	(2)	Nivel alto hasta que sea muestreada

NOTAS:
(1) En el caso de TRAP y RST 5.5-7.5, el contenido del contador de programa se introduce en la pila antes de que ocurra la bifurcación.
(2) Depende de la instrucción que suministra al 8085A, el 8259 u otro circuito, cuando se tenga conocimiento de la interrupción.

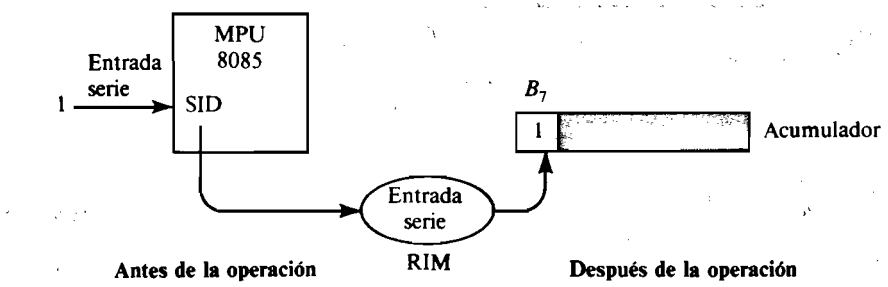
Figura 8.6. Interrupciones hardware de la MPU 8085. (*Cortesía de Intel Corporation.*)

Un solo bit serie puede sacarse vía la patilla de salida serie (*SOD*) utilizando la instrucción especial SIM del 8085. La operación de salida serie está explicada en la Figura 8.7*b*. Un nivel BAJO sale a la patilla *SOD* de la Figura 8.7*b* vía el cerrojo de salida serie. Observar que la fuente del dato era el bit 7 del acumulador. Para que se produzca la salida serie hay que poner a 1 el bit 6 (B_6) del acumulador.

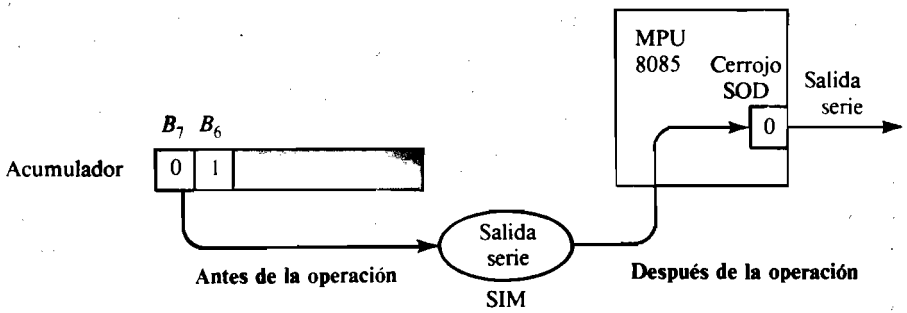
La entrada serie (*SID*) también puede utilizarse como una entrada de *TEST* de propósito general, mientras que las patillas de salida serie (*SOD*) pueden servir como salida de control de 1 bit. En una sección posterior sobre el repertorio de instrucciones del 8085 las instrucciones RIM y SIM se explicarán con más detalle. Son instrucciones que tienen varios propósitos. El nemotécnico RIM realmente significa máscara de lectura de interrupción (*read* interrupt *mask*), mientras que el nemotécnico SIM significa máscara de inicialización de interrupción (*set* interrupt *mask*).

Los registros programables de la unidad microprocesadora 8085 están resumidos en la Figura 8.8. Estos son los registros de principal interés para el programador. Este tipo de diagrama a veces se denomina modelo de programación para la MPU 8085.

El *acumulador principal* de 8 bits se etiqueta con *A* en el modelo de programación de la Figura 8.8. Los otros registros de propósito general de 8 bits (etiquetados *B, C, D, E, H* y *L*) también se denominan, comúnmente, *acumuladores secundarios/contadores de datos.* Este nom-



(a) Instrucción RIM del 8085 para entrada serie



(b) Instrucción SIM del 8085 para salida serie

Figura 8.7.

bre sugiere su doble uso como acumuladores auxiliares o contadores de datos (o punteros de dirección). El puntero de pila (*SP*) de 16 bits y el contador de programa (*PC*) también se muestran en el modelo de programación de la Figura 8.8.

Palabra de status del programa (señalizadores)		Acumulador principal		
PSW	(8)	*A*	(8)	
B	(8)	*C*	(8)	Acumuladores secundarios/contadores de datos
D	(8)	*E*	(8)	
H	(8)	*L*	(8)	
SP			(16)	Puntero de pila
PC			(16)	Contador de programa

Figura 8.8. Modelo de programación del microprocesador 8085.

PROBLEMAS RESUELTOS

8.10. Acudir a la Figura 8.4. El microprocesador 8085 tiene un contador de programa de _____ (número) - bits que alimenta el cerrojo de direcciones. En el cerrojo de direcciones la dirección es dividida en dos partes y enviada a los periféricos vía el bus de _____ y el bus de _____ / _____ .

Solución:

De acuerdo con el diagrama de la Figura 8.4, el microprocesador 8085 tiene un contador de programa de 16 bits que alimenta el cerrojo de direcciones. En el cerrojo de direcciones la dirección es dividida en dos y enviada a los periféricos vía el bus de direcciones y el bus de direcciones/datos.

8.11. Acudir a la Figura 8.4. La línea de control de salida *ALE* estará en el nivel _____ (ALTO, BAJO) cuando la MPU envíe una dirección al bus de direcciones/datos.

Solución:

La línea de control de salida estará en ALTA cuando la MPU envíe una dirección al bus de direcciones/datos. Cuando la línea de control *ALE* alcance el nivel BAJO, el bus de direcciones/datos se convierte en un bus de datos bidireccional.

8.12. Acudir a la Figura 8.4. Las patillas *SID* y *SOD* del microprocesador 8085 están relacionadas con la sección de control de _____ (interrupciones, E/S serie).

Solución:

De acuerdo con la Figura 8.4, las patillas *SID* y *SOD* del microprocesador 8085 están muy relacionadas con la sección de control de E/S serie.

8.13. Además del acumulador, citar seis registros de propósito general de 8 bits del microprocesador 8085.

Solución:

Además del acumulador, los seis registros de propósito general de 8 bits son el *B, C, D, E, H* y *L*. Estos pueden utilizarse como registros pares (registros de 16 bits) durante ciertas operaciones. Los registros pares son *BC, DE* y *HL*.

8.14. Este registro de 16 bits siempre contiene una dirección y apunta al tope de la pila de la RAM.

Solución:

El puntero de pila siempre contiene una dirección y apunta al tope de la pila de la RAM.

8.15. Citar cinco indicadores de status del registro de señalizadores del microprocesador 8085.

Solución:

De acuerdo con la Figura 8.5, los cinco señalizadores, o indicadores de status, del microprocesador 8085 son el señalizador de signo, el de cero, el de arrastre auxiliar, el de paridad y el de arrastre.

8.16. El contenido del acumulador (la suma) de la Figura 8.9 después de la operación de suma será _____ (8 bits).

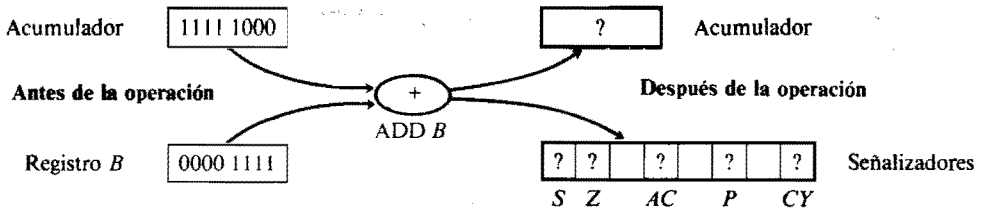


Figura 8.9.

Solución:

Sumar los binarios 11110000 + 00001111 = 1 000 0111. El contenido del acumulador de la Figura 8.9 después de la operación de suma será 00000111 (los 8 bits menos significativos de la suma).

8.17. El señalizador de signo (*S*) estará a _____ (0, 1) después de la operación de suma de la Figura 8.9.

Solución:

El bit más significativo (bit 7) del acumulador es el bit de signo. Después de la operación de suma el acumulador contiene 00000111 en la Figura 8.9. Por tanto el bit de signo estará a 0.

8.18. El señalizador de cero (*Z*) estará a _____ (0, 1) después de la operación de suma de la Figura 8.9.

Solución:

El contenido del acumulador después de la operación de suma en la Figura 8.9 es 00000111; como no es cero, el señalizador de cero estará a 0.

8.19. El señalizador de arrastre auxiliar (*AC*) estará a _____ (0, 1) después de la operación de suma de la Figura 8.9.

Solución:

Las posiciones B_3 del acumulador y del registro *B* tienen un 1 en la Figura 8.9. En binario $1 + 1 = 10_2$. Esto significa que habrá una salida de arrastre de la posición B_3 a la posición B_4. Debido a este arrastre el señalizador de arrastre auxiliar (*AC*) se pondrá a 1, después de la operación de suma de la Figura 8.9.

8.20. El señalizador de paridad (*P*) estará a _____ (0, 1) después de la operación de suma de la Figura 8.9.

Solución:

Después de la operación de la Figura 8.9, el acumulador contendrá el binario 00000111. Esta suma contiene tres 1, que *no es paridad par*, y por tanto el señalizador de paridad estará a 0.

8.21. El señalizador de arrastre (*CY*) estará a _____ (0, 1) después de la operación de suma de la Figura 8.9.

Solución:

Sumar los binarios 11111000 + 00001111 = 1 0000 0111. Debido a que el «overflow» (rebose) sobrepasa el límite de 8 bits del acumulador, hay un arrastre en la posición B_7. El señalizador de arrastre (*CY*) se pondrá a 1 después de la operación de suma en la Figura 8.9.

8.22. La interrupción hardware de prioridad más elevada en el microprocesador 8085 que no puede ser inhabilitada es la _____ (*INTR, TRAP*).

Solución:

La interrupción hardware de más alta prioridad en el microprocesador 8085 que no puede ser inhabilitada se llama *TRAP*.

8.23. Acudir a la Figura 8.6. Cuando se activa la interrupción hardware *RST 7.5*, la CPU guarda el contenido de _____ _____ en la pila y bifurca a la dirección de memoria _____ (hex).

Solución:

De acuerdo con la tabla de la Figura 8.6, cuando se activa la interrupción hardware *RST 7.5*, la CPU guarda el contenido del contador de programa en la pila y bifurca a la dirección de memoria 3CH.

8.24. Mientras el bus de datos transfiere 8 bits de datos en paralelo, la patilla *SID* maneja _____ (número) bit(s) cada vez.

Solución:

De acuerdo con la Figura 8.7*a*, la patilla *SID* (entrada de datos serie) introduce 1 bit cada vez.

8.4. MODOS DE DIRECCIONAMIENTO

Como en el microprocesador genérico, la MPU 8085 utiliza cinco modos de direccionamiento. Estos son:

1. Direccionamiento implicado.
2. Direccionamiento de registro.
3. Direccionamiento inmediato.
4. Direccionamiento directo.
5. Direccionamiento indirecto de registro.

Direccionamiento implicado

El modo de direccionamiento de ciertas instrucciones de la MPU 8085 es *implicado* por (o inherente a) la función de la instrucción. Por ejemplo, la instrucción STC (poner a 1 el señalizador de arrastre) está relacionada con el señalizador de arrastre solamente y no con otros registros o posiciones de memoria.

Direccionamiento de registro

Muchas instrucciones de la MPU 8085 utilizan *direccionamiento de registro*. Cuando se utilizan estas instrucciones, se especifican la fuente del operando y la operación. Considerar la ejecución de la instrucción ADD *C* de la Figura 8.10. En este ejemplo, el operando del registro fuente (registro *C* en este caso) se suma al operando localizado en el acumulador. Una vez que se ejecuta la instrucción ADD *C*, la suma (00001111_2 en este ejemplo) se deposita en el acumulador. Como es habitual, la operación de suma también afecta a los señalizadores del registro de status.

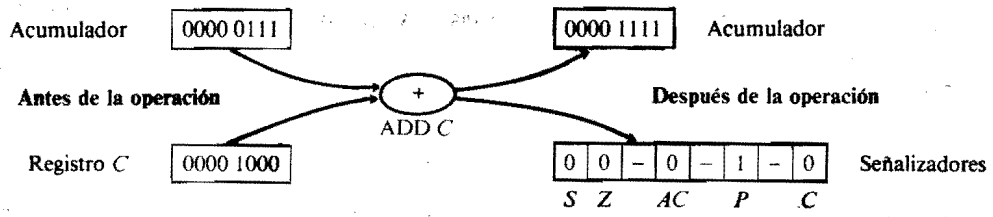


Figura 8.10. La instrucción ADD *C* (modo direccionamiento de registro).

La instrucción ADD *C* ilustrada en la Figura 8.10 utilizaba el direccionamiento de registros. Ambos operandos estaban localizados en registros internos de la MPU (registros *A* y *C*). Las instrucciones que utilizan el direccionamiento de registro son muy eficientes ya que solamente utilizan un espacio de la memoria de programa de 1 byte. También se ejecutan rápidamente porque no tienen que buscar operandos en memoria. Buscar en memoria es una operación relativamente lenta.

Direccionamiento inmediato

Las instrucciones que utilizan *direccionamiento inmediato* tienen los datos inmediatamente a continuación del código de op en la memoria del programa. A título de ejemplo se ilustra la instrucción ADI (suma inmediata) del 8085 en la Figura 8.11. El microprocesador busca el código de op (C6H en este ejemplo) en la memoria de programa. Después de decodificar, la MPU determina que ésta es una instrucción de direccionamiento inmediato. Por tanto encuentra el dato inmediato en la siguiente posición consecutiva de la memoria de programa después del código de op. Este dato inmediato (00001000_2) se suma al contenido del acumulador (00001100_2). Después de la operación, la suma (00010100_2) se coloca en el acumulador.

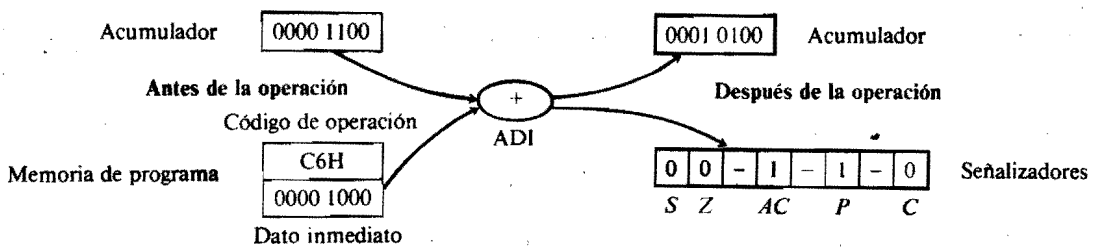


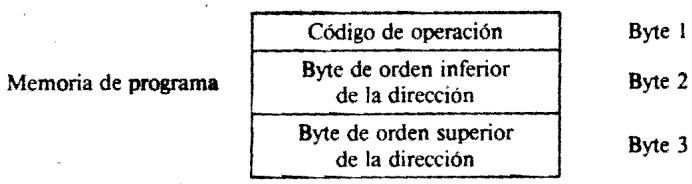
Figura 8.11. La instrucción ADI (modo de direccionamiento inmediato).

Las instrucciones inmediatas del microprocesador 8085 utilizan el acumulador como operando implicado, como en la operación ADI mostrada en la Figura 8.11. La instrucción MVI (transferir inmediato) del 8085 puede transferir datos inmediatos a cualquier registro o posición de memoria. La instrucción LXI (carga registro par inmediato) del 8085 carga un registro par con un valor de 16 bits y por tanto no afecta al acumulador.

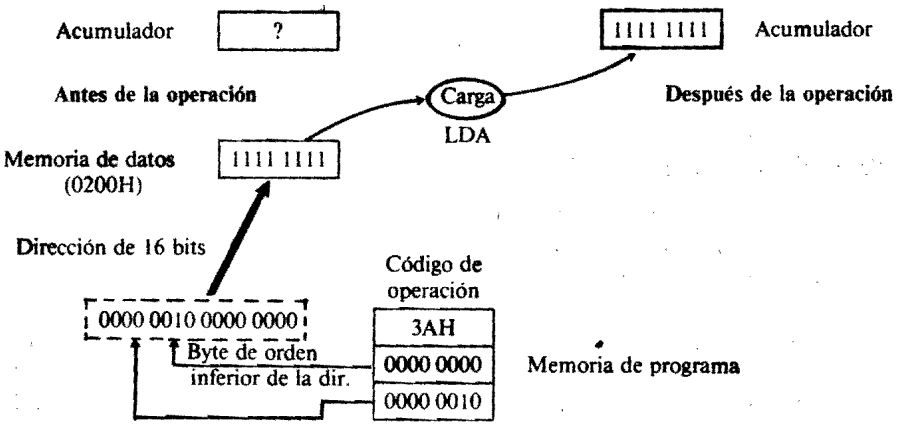
Direccionamiento directo

Las operaciones que utilizan *direccionamiento directo* en la MPU del Intel 8085 se especifican utilizando formatos de instrucción de 3 bytes. Este formato está representado en la Figura 8.12*a*. El byte 1 contiene el código de op para la instrucción de direccionamiento directo. El byte 2 contiene el byte de orden inferior de la dirección del operando. El byte 3 de la instrucción contiene el byte de orden superior de la dirección del operando.

La instrucción LDA (cargar *A* directo) del 8085 se está ejecutando en la Figura 8.12*b*. La instrucción de 3 bytes se muestra en la parte inferior del diagrama. El código de op de la instrucción LDA es 3AH. Los 2 siguientes bytes de la memoria de programa son ensamblados por el microprocesador en una dirección de 16 bits (0200H en este ejemplo). Esta dirección de la memoria de datos (0200H) es accedida entonces por la MPU y su contenido (11111111_2) es cargado entonces en el acumulador.



(a) Formato de instrucción para direccionamiento directo



(b) La instrucción de cargar *A* directo

Figura 8.12.

Las instrucciones que utilizan direccionamiento directo a veces son evitadas por los diseñadores de microprocesadores y por los programadores. El direccionamiento directo requiere mucho espacio de la memoria de programa (3 bytes de memoria de programa por una instrucción de la MPU 8085). Las operaciones de direccionamiento directo también requieren un tiempo de ejecución relativamente largo, debido a los numerosos accesos a memoria que se necesitan.

Direccionamiento indirecto de registro

Las instrucciones de *registro indirecto* referencian la memoria utilizando el contenido de un registro par para señalar la dirección del operando. Un ejemplo de una operación de registro indirecto se ilustra en la Figura 8.13. La instrucción ADD *M* (sumar memoria), de la Figura 8.13, suma el contenido del acumulador con el contenido de la posición de memoria indicado por la dirección del registro par *HL* de la MPU. En este ejemplo, el registro par *HL* señala la posición de memoria 2050H. El operando en la posición 2050H (00000001 + 11111111 = 1 0000 0000. Los 8 bits menos significativos de la suma se depositan en el acumulador después de la operación ADD *M*. Los señalizadores apropiados también se ponen a 1 y 0 según el resultado de la operación de suma y se muestran en la Figura 8.13.

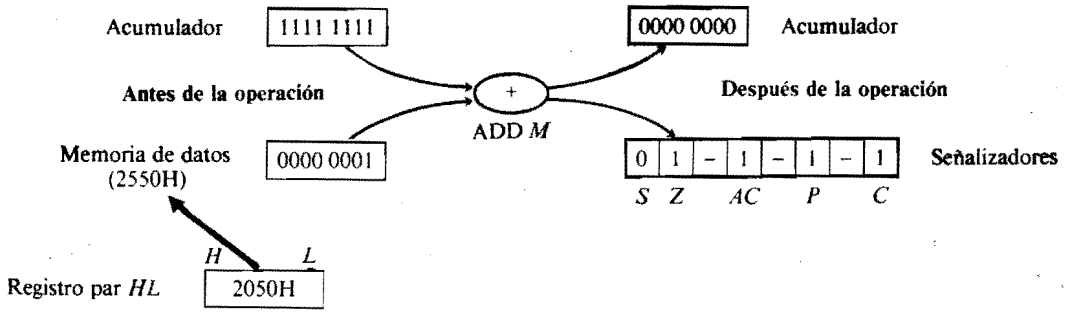


Figura 8.13. Una instrucción de suma indirecta de registro.

Modos de direccionamiento combinados

Algunas instrucciones de la MPU 8085 utilizan una combinación de modos de direccionamiento. Una instrucción de CALL (LLAMADA) a subrutina, por ejemplo, combina el direccionamiento directo e indirecto de registro. El direccionamiento directo en una instrucción CALL especifica la dirección de la subrutina deseada. El direccionamiento indirecto de registro es el del puntero de pila. La instrucción CALL introduce primero el contenido actual del contador de programa en la posición de memoria especificada por el puntero de pila. A continuación, el procesador carga la dirección directa en el contador de programa. Finalmente, la MPU salta a la subrutina cuya dirección se encuentra ahora en el contador de programa.

PROBLEMAS RESUELTOS

8.25. Listar los cinco modos de direccionamiento utilizados por el microprocesador Intel 8085.

Solución:

El microprocesador Intel 8085 utiliza los modos de direccionamiento implicado, de registro, inmediato, directo e indirecto de registro.

8.26. La instrucción CMC (arrastre complementario) del 8085 afecta solamente el señalizador de arrastre y no a otros registros o posiciones de memoria. Por tanto utiliza el modo de direccionamiento _____ (directo, implicado).

Solución:

La instrucción CMC (arrastre complementario) utiliza el modo de direccionamiento implicado porque afecta solamente el señalizador de arrastre y no utiliza otros registros o posiciones de memoria.

8.27. La instrucción MOV *B,A* del 8085 (transfiere *A* a *B*) transfiere el contenido de un registro a otro. La instrucción MOV *B,A* por tanto utiliza el modo de direccionamiento _____ (inmediato, de registro).

Solución:

La instrucción MOV *B,A* del 8085 (transfiere *A* a *B*) ejecuta una operación registro a registro y por tanto utiliza el modo de direccionamiento de registro.

8.28. La instrucción MVI *A* del 8085 mostrada en la Figura 8.14 utiliza el modo de direccionamiento _____ (implicado, inmediato).

Solución:

En la Figura 8.14 se determina que el dato es transferido desde el segundo byte de la memoria del programa al acumulador. Esto significa que la instrucción MVI *A* (transfiere inmediato) del 8085 utiliza el modo de direccionamiento inmediato.

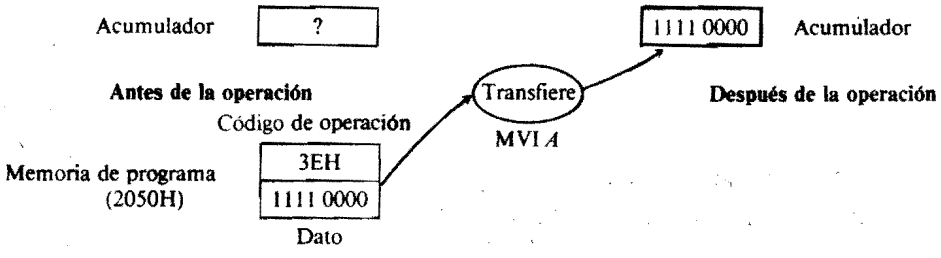


Figura 8.14. Una instrucción de transferencia inmediata.

8.29. La instrucción SUB *M* del 8085 de la Figura 8.15 utiliza el modo de direccionamiento de _____ (registro, indirecto de registro).

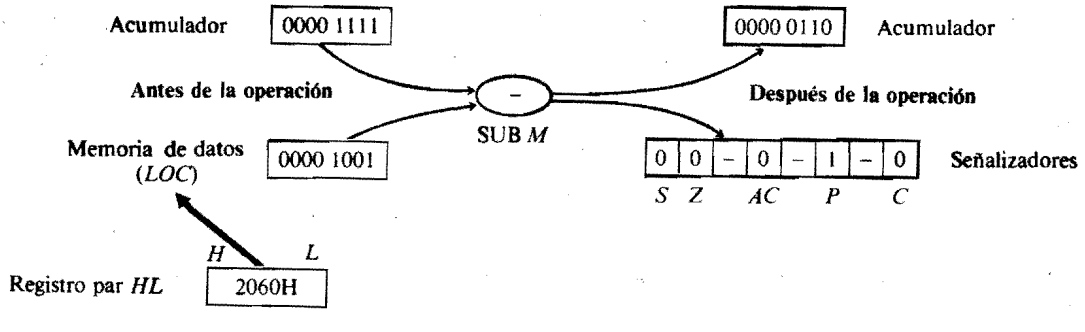


Figura 8.15. La instrucción SUB *M*.

Solución:

A partir de la Figura 8.15 se determina que el sustraendo proviene de la posición de la memoria de datos señalada por el registro par *HL*. Esto significa que la instrucción SUB *M* del 8085 utiliza el modo de direccionamiento indirecto de registro.

8.30. En la Figura 8.15, el minuendo está localizado en el acumulador mientras que el sustraendo se encuentra en la posición _____ (hex) de la memoria de datos.

Solución:

En la Figura 8.15, el minuendo está localizado en el acumulador mientras que el sustraendo se encuentra en la posición 2060H de la memoria de datos como señala el registro par *HL*.

8.5. REPERTORIO DE INSTRUCCIONES DEL 8080/8085

Los microprocesadores Intel 8080 y 8085 son utilizados para formar computadoras de programa-almacenado. Las instrucciones del programa son almacenadas como bytes de 8 bits en un área de memoria denominada memoria de programa. El microprocesador Intel 8080/8085 utiliza instrucciones de 1, 2 y 3 bytes. El primer byte de la instrucción es siempre el código de op (código de operación), que especifica cuál de las 200 instrucciones del microprocesador debe ejecutarse. La MPU reconoce estos códigos de op cuando están en forma binaria de 8 bits. El conjunto de instrucciones a las cuales responderá la MPU 8080 y 8085 está fijado, permanentemente, en el diseño de las pastillas. El 8085 utiliza los mismos códigos de op que la MPU del 8080. El repertorio de instrucciones del 8085 también contiene dos instrucciones extra, además de las utilizadas por la MPU 8080.

Intel Corporation agrupa las instrucciones del microprocesador 8080/8085 en las siguientes categorías funcionales:

1. Grupo de transferencia de datos.
2. Grupo aritmético.
3. Grupo lógico.
4. Grupo de bifurcación.
5. Grupo de pila, E/S y de control máquina.

El *grupo* de instrucciones de *transferencia de datos* transfiere datos entre registros o entre posiciones de memoria y registros. Entre estas instrucciones se encuentran transferencias, cargas, almacenamientos e intercambios.

El *grupo aritmético* de instrucciones realiza sumas, restas, incrementos o decrementos en datos de registros o memoria. El *grupo lógico* de instrucciones realiza operaciones AND, OR y XOR, comparaciones, desplazamientos circulares o complementaciones de datos en registros o entre memoria y un registro.

El *grupo* de instrucciones de *bifurcación* indica saltos condicionales o incondicionales, llamadas, vueltas y reinicializaciones. El *grupo de pila, E/S y de control de máquina* incluye instrucciones para mantener la pila, leer los puertos de entrada, escribir en los puertos de salida, inicializar y leer máscaras de interrupción e inicializar y borrar señalizadores.

Un resumen del repertorio de *instrucciones* del microprocesador 8080/8085 se detalla en la Figura 8.16. Este es un listado alfabético por nemotécnicos. Se utilizan los nemotécnicos con derecho de copia de Intel junto con los códigos de op únicos del 8080/8085. Los códigos de op

	Nemotécnico	Código de op (hex)	Descripción
A	ADD *A*	87	Suma *A* a *A* (doble *A*)
	ADD *B*	80	Suma *B* a *A*
	ADD *C*	81	Suma *C* a *A*
	ADD *D*	82	Suma *D* a *A*
	ADD *E*	83	Suma *E* a *A*
	ADD *H*	84	Suma *H* a *A*
	ADD *L*	85	Suma *L* a *A*
	ADD *M*	86	Suma LOC memoria (*H* & *L*) a *A*
	ADI *v*	C6	Suma dato inmediato *v* a *A*
	ADC *A*	8F	Suma *A* a *A* con arrastre (doble *A* con arrastre)
	ADC *B*	88	Suma *B* a *A* con arrastre
	ADC *C*	89	Suma *C* a *A* con arrastre
	ADC *D*	8A	Suma *D* a *A* con arrastre
	ADC *E*	8B	Suma *E* a *A* con arrastre
	ADC *H*	8C	Suma *H* a *A* con arrastre
	ADC *L*	8D	Suma *L* a *A* con arrastre
	ADC *M*	8E	Suma LOC memoria (*H* & *L*) a *A* con arrastre
	ACI *v*	CE	Suma dato inmediato *v* a *A* con arrastre
	ANA *A*	A7	Examina *A* y borra arrastre
	ANA *B*	A0	AND *B* con *A*
	ANA *C*	A1	AND *C* con *A*
	ANA *D*	A2	AND *D* con *A*
	ANA *E*	A3	AND *E* con *A*
	ANA *H*	A4	AND *H* con *A*
	ANA *L*	A5	AND *L* con *A*
	ANA *M*	A6	AND LOC memoria (*H* & *L*) con *A*
	ANI *v*	E6	AND dato inmediato *v* con *A*
C	CALL *aa*	CD	Call subrutina en dirección *aa*
	CZ *aa*	CC	Si cero, CALL en dirección *aa*
	CNZ *aa*	C4	Si no cero, CALL en dirección *aa*
	CP *aa*	F4	Si más, CALL en dirección *aa*
	CM *aa*	FC	Si menos, CALL en dirección *aa*
	CC *aa*	DD	Si arrastre, CALL en dirección *aa*
	CNC *aa*	D4	Si no arrastre, CALL en dirección *aa*
	CPE *aa*	EC	Si paridad par, CALL en dirección *aa*
	CPO *aa*	E4	Si paridad impar, CALL en dirección *aa*
	CMA	2F	Complementa *A*
	CMC	3F	Complementa arrastre
	CMP *A*	BF	Pone a 1 el señalizador de cero
	CMP *B*	B8	Compara *A* con *B*
	CMP *C*	B9	Compara *A* con *C*
	CMP *D*	BA	Compara *A* con *D*
	CMP *E*	BB	Compara *A* con *E*
	CMP *H*	BC	Compara *A* con *H*
	CMP *L*	BD	Compara *A* con *L*
	CMP *M*	BE	Compara *A* con LOC memoria (*H* & *L*)
	CPI *v*	FE	Compara *A* con dato inmediato *v*
D	DAA	27	Ajuste decimal *A*
	DAD *B*	09	Suma *B* & *C* a *H* & *L*
	DAD *D*	19	Suma *D* & *E* a *H* & *L*
	DAD *H*	29	Suma *H* & *L* a *H* & *L* (doble *H* & *L*)
	DAD *SP*	39	Suma *SP* a *H* & *L*

Figura 8.16. Resumen del repertorio de instrucciones del microprocesador 8080/8085. (Todos los nemotécnicos tienen derecho de copia © Intel Corporation, 1976.)

	Nemotécnico	Código de op (hex)	Descripción
D	DCR *A*	3D	Decrementa *A*
	DCR *B*	05	Decrementa *B*
	DCR *C*	0D	Decrementa *C*
	DCR *D*	15	Decrementa *D*
	DCR *E*	1D	Decrementa *E*
	DCR *H*	25	Decrementa *H*
	DCR *L*	2D	Decrementa *L*
	DCR *M*	35	Decrementa LOC memoria (*H* & *L*)
	DCX *B*	0B	Decrementa *B* & *C*
	DCX *D*	1B	Decrementa *D* & *E*
	DCX *H*	2B	Decrementa *H* & *L*
	DCX *SP*	3B	Decrementa *SP*
	DI	F3	Inhabilita interrupciones
E	EI	FB	Habilita interrupciones
H	HLT	76	Alto hasta interrupción
I	IN *v*	DB	Entrada del dispositivo *v*
	INR *A*	3C	Incrementa *A*
	INR *B*	04	Incrementa *B*
	INR *C*	0C	Incrementa *C*
	INR *D*	14	Incrementa *D*
	INR *E*	1C	Incrementa *E*
	INR *H*	24	Incrementa *H*
	INR *L*	2C	Incrementa *L*
	INR *M*	34	Incrementa LOC memoria (*H* & *L*)
	INX *B*	03	Incrementa *B* & *C*
	INX *D*	13	Incrementa *D* & *E*
	INX *H*	23	Incrementa *H* & *L*
	INX *SP*	33	Incrementa *SP*
J	JMP *aa*	C3	Salta a dirección *aa*
	JZ *aa*	CA	Si cero, JMP en dirección *aa*
	JNZ *aa*	C2	Si no cero, JMP en dirección *aa*
	JP *aa*	F2	Si más, JMP en dirección *aa*
	JM *aa*	FA	Si menos, JMP en dirección *aa*
	JC *aa*	DA	Si arrastre, JMP en dirección *aa*
	JNC *aa*	D2	Si no arrastre, JMP en dirección *aa*
	JPE *aa*	EA	Si paridad par, JMP en dirección *aa*
	JPO *aa*	E2	Si paridad impar, JMP en dirección *aa*
L	LDA *aa*	3A	Carga *A* de dirección *aa*
	LDAX *B*	OA	Carga *A* de LOC memoria (*B* & *C*)
	LDAX *D*	1A	Carga *A* de LOC memoria (*D* & *E*)
	LHLD *aa*	2A	Carga *H* & *L* de dirección *aa*
	LXI *B,vv*	01	Carga *B* & *C* con dato inmediato *vv*
	LXI *D,vv*	11	Carga *D* & *E* con dato inmediato *vv*
	LXI *H,vv*	21	Carga *H* & *L* con dato inmediato *vv*
	LXI *SP,vv*	31	Carga *SP* con dato inmediato *vv*
M	MOV *A,B*	78	Transfiere *B* a *A*
	MOV *A,C*	79	Transfiere *C* a *A*
	MOV *A,D*	7A	Transfiere *D* a *A*
	MOV *A,E*	7B	Transfiere *E* a *A*
	MOV *A,H*	7C	Transfiere *H* a *A*
	MOV *A,L*	7D	Transfiere *L* a *A*
	MOV *A,M*	7E	Transfiere LOC memoria (*H* & *L*) a *A*

Figura 8.16. *(continuación).*

	Nemotécnico	Código de op (hex)	Descripción
M	MOV *B,A*	47	Transfiere *A* a *B*
	MOV *B,C*	41	Transfiere *C* a *B*
	MOV *B,D*	42	Transfiere *D* a *B*
	MOV *B,E*	43	Transfiere *E* a *B*
	MOV *B,H*	44	Transfiere *H* a *B*
	MOV *B,L*	45	Transfiere *L* a *B*
	MOV *B,M*	46	Transfiere LOC memoria (*H* & *L*) a *B*
	MOV *C,A*	4F	Transfiere *A* a *C*
	MOV *C,B*	48	Transfiere *B* a *C*
	MOV *C,D*	4A	Transfiere *D* a *C*
	MOV *C,E*	4B	Transfiere *E* a *C*
	MOV *C,H*	4C	Transfiere *H* a *C*
	MOV *C,L*	4D	Transfiere *L* a *C*
	MOV *C,M*	4E	Transfiere LOC memoria (*H* & *L*) a *C*
	MOV *D,A*	57	Transfiere *A* a *D*
	MOV *D,B*	50	Transfiere *B* a *D*
	MOV *D,C*	51	Transfiere *C* a *D*
	MOV *D,E*	53	Transfiere *E* a *D*
	MOV *D,H*	54	Transfiere *H* a *D*
	MOV *D,L*	55	Transfiere *L* a *D*
	MOV *D,M*	56	Transfiere LOC memoria (*H* & *L*) a *D*
	MOV *E,A*	5F	Transfiere *A* a *E*
	MOV *E,B*	58	Transfiere *B* a *E*
	MOV *E,C*	59	Transfiere *C* a *E*
	MOV *E,D*	5A	Transfiere *D* a *E*
	MOV *E,H*	5C	Transfiere *H* a *E*
	MOV *E,L*	5D	Transfiere *L* a *E*
	MOV *E,M*	5E	Transfiere LOC memoria (*H* & *L*) a *E*
	MOV *H,A*	67	Transfiere *A* a *H*
	MOV *H,B*	60	Transfiere *B* a *H*
	MOV *H,C*	61	Transfiere *C* a *H*
	MOV *H,D*	62	Transfiere *D* a *H*
	MOV *H,E*	63	Transfiere *E* a *H*
	MOV *H,L*	65	Transfiere *L* a *H*
	MOV *H,M*	66	Transfiere LOC memoria (*H* & *L*) a *H*
	MOV *L,A*	6F	Transfiere *A* a *L*
	MOV *L,B*	68	Transfiere *B* a *L*
	MOV *L,C*	69	Transfiere *C* a *L*
	MOV *L,D*	6A	Transfiere *D* a *L*
	MOV *L,E*	6B	Transfiere *E* a *L*
	MOV *L,H*	6C	Transfiere *H* a *L*
	MOV *L,M*	6E	Transfiere LOC memoria (*H* & *L*) a *L*
	MOV *M,A*	77	Transfiere *A* a LOC memoria (*H* & *L*)
	MOV *M,B*	70	Transfiere *B* a LOC memoria (*H* & *L*)
	MOV *M,C*	71	Transfiere *C* a LOC memoria (*H* & *L*)
	MOV *M,D*	72	Transfiere *D* a LOC memoria (*H* & *L*)
	MOV *M,E*	73	Transfiere *E* a LOC memoria (*H* & *L*)
	MOV *M,H*	74	Transfiere *H* a LOC memoria (*H* & *L*)
	MOV *M,L*	75	Transfiere *L* a LOC memoria (*H* & *L*)
	MVI *A,v*	3E	Transfiere dato inmediato *v* a *A*
	MVI *B,v*	06	Transfiere dato inmediato *v* a *B*
	MVI *C,v*	0E	Transfiere dato inmediato *v* a *C*
	MVI *D,v*	16	Transfiere dato inmediato *v* a *D*
	MVI *E,v*	1E	Transfiere dato inmediato *v* a *E*
	MVI *H,v*	26	Transfiere dato inmediato *v* a *H*
	MVI *L,v*	2E	Transfiere dato inmediato *v* a *L*
	MVI *M,v*	36	Transfiere dato inmediato *v* a memoria LOC (*H* & *L*)

Figura 8.16. *(continuación).*

	Nemotécnico	Código de op (hex)	Descripción
N	NOP	00	No operación
O	ORA *A*	B7	Examina *A* y borra arrastre
	ORA *B*	B0	OR *B* con *A*
	ORA *C*	B1	OR *C* con *A*
	ORA *D*	B2	OR *D* con *A*
	ORA *E*	B3	OR *E* con *A*
	ORA *H*	B4	OR *H* con *A*
	ORA *L*	B5	OR *L* con *A*
	ORA *M*	B6	OR LOC memoria (*H* & *L*) con *A*
	ORI *v*	F6	OR dato inmediato *v* con *A*
	OUT *v*	D3	Saca *A* a dispositivo *v*
P	PCHL	E9	Salta a LOC memoria contenida en (*H* & *L*)
	POP *B*	C1	Saca *B* & *C* de pila
	POP *D*	D1	Saca *D* & *E* de pila
	POP *H*	E1	Saca *H* & *L* de pila
	POP *PSW*	F1	Saca *A* y los señalizadores de pila
	PUSH *B*	C5	Introduce *B* & *C* en pila
	PUSH *D*	D5	Introduce *D* & *E* en pila
	PUSH *H*	E5	Introduce *H* & *L* en pila
	PUSH *PSW*	F5	Introduce *A* y los señalizadores en pila
R	RAL	17	Desplazamiento circular *CY* + *A* a la izquierda
	RAR	1F	Desplazamiento circular *CY* + *A* a la derecha
	RLC	07	Desplazamiento circular *A* a la izquierda y arrastre
	RRC	0F	Desplazamiento circular *A* a la derecha y arrastre
	RIM	20	Lee máscara de interrupción (8085 sólo)
	RET	C9	Vuelve de subrutina
	RZ	C8	Si cero, vuelve de subrutina
	RNZ	C0	Si no cero, vuelve de subrutina
	RP	F0	Si más, vuelve de subrutina
	RM	F8	Si menos, vuelve de subrutina
	RC	D8	Si arrastre, vuelve de subrutina
	RNC	D0	Si no arrastre, vuelve de subrutina
	RPE	E8	Si paridad par, vuelve de subrutina
	RPO	E0	Si paridad impar, vuelve de subrutina
	RST 0	C7	Rearrancar subrutina en dirección 00H
	RST 1	CF	Rearrancar subrutina en dirección 08H
	RST 2	D7	Rearrancar subrutina en dirección 10H
	RST 3	DF	Rearrancar subrutina en dirección 18H
	RST 4	E7	Rearrancar subrutina en dirección 20H
	RST 5	EF	Rearrancar subrutina en dirección 28H
	RST 6	F7	Rearrancar subrutina en dirección 30H
	RST 7	FF	Rearrancar subrutina en dirección 38H
S	SIM	30	Pone a 1 máscara de interrupción (8085 sólo)
	SPHL	F9	Carga *SP* de *H* & *L*
	SHLD *aa*	22	Almacena *H* & *L* en LOC memoria *aa*
	STA *aa*	32	Almacena *A* en LOC memoria *aa*
	STAX *B*	02	Almacena *A* en LOC memoria (*B* & *C*)
	STAX *D*	12	Almacena *A* en LOC memoria (*D* & *E*)
	STC	37	Pone a 1 señalizador de arrastre

Figura 8.16. *(continuación).*

	Nemotécnico	Código de op (hex)	Descripción
S	SUB *A*	97	Borra *A*
	SUB *B*	90	Resta *B* de *A*
	SUB *C*	91	Resta *C* de *A*
	SUB *D*	92	Resta *D* de *A*
	SUB *E*	93	Resta *E* de *A*
	SUB *H*	94	Resta *H* de *A*
	SUB *L*	95	Resta *L* de *A*
	SUB *M*	96	Resta contenido de LOC memoria (*H* & *L*) de *A*
	SUI *v*	D6	Resta dato inmediato *v* de *A*
	SBB *A*	9F	Pone *A* con arrastre menos
	SBB *B*	98	Resta *B* de *A* con préstamo
	SBB *C*	99	Resta *C* de *A* con préstamo
	SBB *D*	9A	Resta *D* de *A* con préstamo
	SBB *E*	9B	Resta *E* de *A* con préstamo
	SBB *H*	9C	Resta *H* de *A* con préstamo
	SBB *L*	9D	Resta *L* de *A* con préstamo
	SBB *M*	9E	Resta LOC memoria (*H* & *L*) de *A* con préstamo
	SBI *v*	DE	Resta dato inmediato *v* de *A* con préstamo
X	XCHG	EB	Cambia *D* & *E* con *H* & *L*
	XTHL	E3	Cambia tope de pila con *H* & *L*
	XRA *A*	AF	Borra *A*
	XRA *B*	A8	OR exclusiva de *B* con *A*
	XRA *C*	A9	OR exclusiva de *C* con *A*
	XRA *D*	AA	OR exclusiva de *D* con *A*
	XRA *E*	AB	OR exclusiva de *E* con *A*
	XRA *H*	AC	OR exclusiva de *H* con *A*
	XRA *L*	AD	OR exclusiva de *L* con *A*
	XRA *M*	AE	OR exclusiva de LOC memoria (*H* & *L*) con *A*
	XRI *v*	EE	OR exclusiva de dato inmediato *v* con *A*

Figura 8.16. *(continuación).*

están representados en notación hexadecimal en la Figura 8.16. También una breve descripción para cada una de las 239 instrucciones del microprocesador. En la Figura 8.16 se utilizan dos instrucciones exclusivas de la MPU 8085. Estas son las instrucciones RIM y SIM.

El repertorio de instrucciones del microprocesador genérico (Cap. 6) es un subconjunto del 8080/8085 y por tanto utiliza los mismos nemotécnicos y códigos de operación. Como la MPU 8080/8085 tiene más registros y funciones que el microprocesador genérico, ésta necesitará más instrucciones. Por ejemplo, el repertorio de instrucciones del microprocesador genérico tiene cuatro instrucciones de suma (ver Fig. 6.4), mientras que las 18 primeras instrucciones listadas en la Figura 8.16 para la MPU del 8080/8085 son instrucciones de suma. El 8080/8085 también tiene cuatro instrucciones más de suma doble (DAD *B*, DAD *D*, etc.) para un total de 22 instrucciones de suma. Después de utilizar el repertorio de instrucciones del microprocesador genérico, la mayoría de los estudiantes tendrá un conocimiento razonable de la función que realizan aproximadamente la mitad de las instrucciones del 8080/8085 listadas en la Figura 8.16. Un estudio más detallado del repertorio de instrucciones de la MPU 8080/8085 se dará a continuación.

PROBLEMAS RESUELTOS

8.31. Acudir a la Figura 8.16. ¿Cuántas instrucciones lógicas AND hay listadas para el microprocesador 8080/8085?

Solución:

El resumen del repertorio de instrucciones del 8080/8085 de la Figura 8.16 listaba nueve operaciones AND diferentes comenzando con ANA *A*.

8.32. Acudir a la Figura 8.16. El repertorio de instrucciones del microprocesador genérico contiene una instrucción de llamada a subrutina, mientras que la MPU 8080/8085 puede realizar _____ (número) de operaciones de llamada diferentes.

Solución:

La MPU 8080/8085 puede realizar nueve operaciones de llamada diferentes empezando con la instrucción CALL *aa* en la Figura 8.16.

8.33. Acudir a la Figura 8.16. El repertorio de instrucciones del 8080/8085 lista _____ (número) de operaciones de resta diferentes.

Solución:

El repertorio de instrucciones del 8080/8085 lista 18 operaciones diferentes de resta empezando con SUB *A* en la Figura 8.16. No están incluidas operaciones de comparación que utilizan restas internas cuando se comparan datos.

8.34. Acudir a la Figura 8.16. Algunos nemotécnicos del 8080/8085 tienen una X adicional como, por ejemplo, CDX *B*, INX *B*, LXI *B* y STAX *B*. Cada vez que aparece la X en estas instrucciones, un _____ _____ (registro par, registro simple) es afectado por la operación.

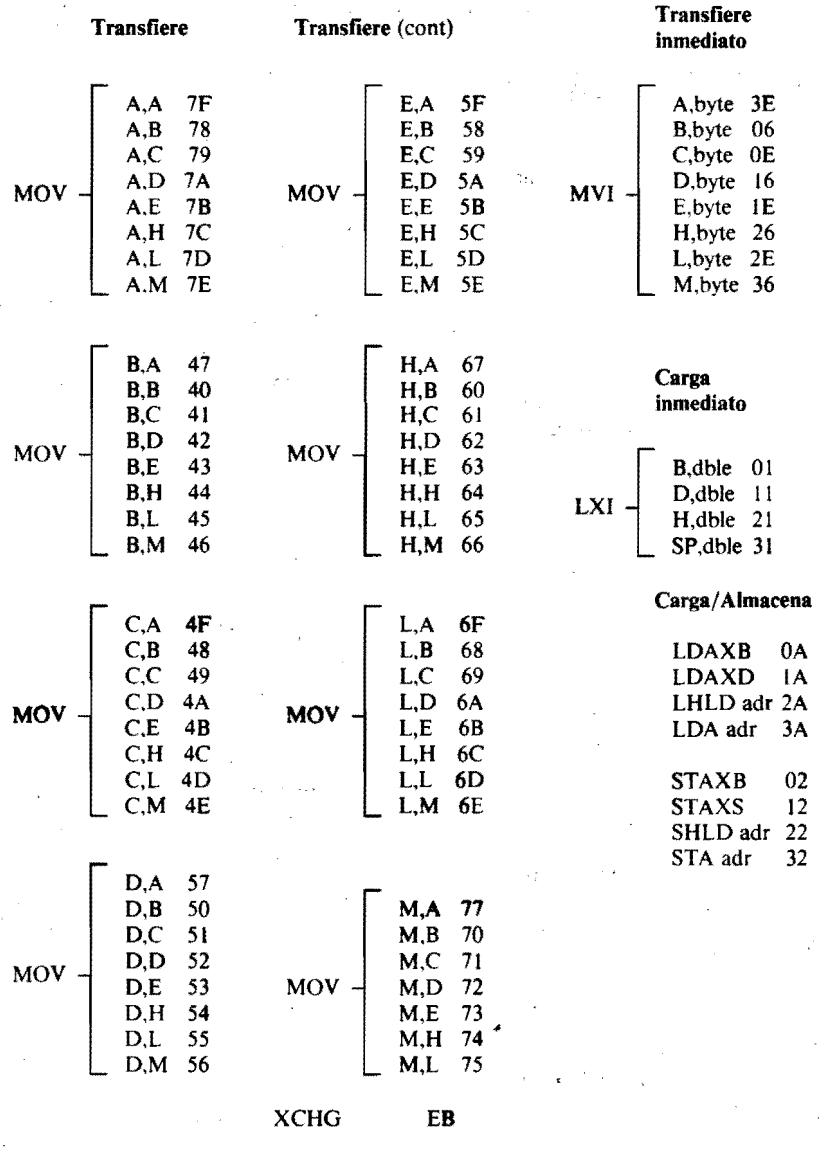
Solución:

De acuerdo con las descripciones de la Figura 8.16, cada vez que aparece una X adicional en las instrucciones (como CDX *B*, INX *B*, LXI *B* y STAX *B*), un registro par es afectado por la operación.

8.6. INSTRUCCIONES DE TRANSFERENCIA DE DATOS DEL 8080/8085

Un resumen de las instrucciones de transferencia de datos del microprocesador 8080/8085 se muestra en la Figura 8.17. Estas instrucciones incluyen operaciones de transferir, intercambiar, cargar y almacenar. La Figura 8.17 da sólo los nemotécnicos y los códigos de op hexadecimales para estas 84 instrucciones. Las instrucciones de transferencia de datos son de las más utilizadas del microprocesador.

El *grupo de instrucciones de transferencia de datos* se detallará en las páginas 275-277 utilizando el formato propio de Intel Corporation. Este grupo de instrucciones transfiere datos a y desde registros y memoria. Los señalizadores de condición no *son afectados* por cualquier instrucción de este grupo. Acudir a la Figura 8.18 para el significado de las *abreviaturas* (como por ejemplo *r*1, DDD, SSS, etc.) utilizadas en las siguientes descripciones del *Manual del Usuario* de Intel.



adr = dirección de 16 bits (segundo y tercer bytes de instrucciones de 3 bytes)

Figura 8.17. Resumen de las instrucciones de transferencia de datos del 8080/8085. (Todos los nemotécnicos tienen derecho de copia © de Intel Corporation, 1976.)

Considerar la instrucción MOV *A,B* (código de op = 78H) del 8080/8085. MOV *A,B* puede representar una instrucción que transfiere datos del registro *B* al acumulador (registro *A*). Este tipo de instrucción está detallada en la página 275 en la sección «transfiere registro» (MOV *r*1, *r*2). Observar que el primer registro (*A* en este ejemplo) es el destino de la transferencia, mientras que

SIMBOLOS	SIGNIFICADO
acumulador	Registro A
addr	Dirección de 16 bits
dato	Cantidad de 8 bits
16 datos	Datos de 16 bits
byte 2	El segundo byte de la instrucción
byte 3	El tercer byte de la instrucción
puerto	Dirección de 8 bits de un dispositivo de E/S
r,r1,r2	Uno de los registros A,B,C,D,E,H,L
DDD,SSS	El patrón de bits designa uno de los registros A,B,C,D,E,H,L (DDD = destino, SSS = fuente):

DDD o SSS	REGISTRO NOMBRE
111	A
000	B
001	C
010	D
011	E
100	H
101	L

rp	Uno de los registros pares: B representa el par B,C siendo B el registro de orden superior y C el de orden inferior; D representa el par D,E siendo D el registro de orden superior y E el de orden inferior; H representa el par H,L siendo H el registro de orden superior y L el de orden inferior; SP representa el registro puntero de pila de 16 bits.
RP	El patrón de bits designa uno de los registros pares B,D,H,SP:

RP	REGISTRO PAR
00	B-C
01	D-E
10	H-L
11	SP

rh	El primer registro (orden superior) de un registro par designado.
rl	El segundo registro (orden inferior) de un registro par designado.
PC	El contador de programa de 16 bits (PCH y PCL son utilizados para referenciar los 8 bits de orden superior e inferior respectivamente).
SP	Registro puntero de pila de 16 bits (SPH y SPL son utilizados para referenciar los 8 bits de orden superior e inferior respectivamente).
r_m	Bit m del registro r (los bits están numerados de 7 a 0 de izquierda a derecha).
ETIQUETA	Dirección de 16 bits de subrutina.
	Los señalizadores de condición:
Z	Cero
S	Signo
P	Paridad
CY	Arrastre
AC	Arrastre auxiliar
( )	Contenido de la posición de memoria o registro encerrado en los paréntesis.
←	«Es transferido a»
∧	AND lógica
∀	OR exclusiva
∨	OR inclusiva
+	Suma
−	Resta en complemento a dos
*	Multiplicación
↔	«Se intercambia con»
____	El complemento a uno (e.j., $\overline{(A)}$)
n	Número de 0 a 7
NNN	Representación binaria de 000 a 111 para los núms. 0 a 7 respectivamente.

Figura 8.18. Abreviaturas utilizadas con las descripciones de Intel Corporation. (*Cortesía de Intel Corporation.*)

el segundo registro (*B* en este ejemplo) es la fuente del dato. El patrón del código de op general para este tipo de instrucción es 01DDDSSS y puede completarse utilizando la información de la Figura 8.18. El destino es el acumulador que utiliza el patrón de bits 111, mientras que la fuente es el registro *B*, que utiliza el patrón de bits 000. El código de op es entonces $01\ 111\ 000_2$, ó 78 en la notación hexadecimal. De acuerdo con la información previa de Intel la instrucción MOV *A,B* utiliza un ciclo máquina para ejecutar la transferencia de datos. El ciclo máquina del 8085 utiliza

cuatro estados *T*. La instrucción MOV *A,B* utiliza direccionamiento de registro y ningún señalizador es afectado durante la ejecución de esta instrucción. Cada instrucción de transferencia de datos de la Figura 8.17 encaja una de las categorías más generales y puede analizarse como funciona.

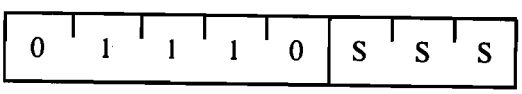
MOV r1, r2 (Transfiere registro)
(r1) ← (r2)
El contenido del registro r2 es transferido al registro r1.

Ciclos: 1
Estados: 4 (8085), 5 (8080)
Direccionamiento: registro
Señalizadores: ninguno

MOV r, M (Transfiere de memoria)
(r) ← ((H) (L))
El contenido de la posición de memoria, cuya dirección está en los registros H y L es transferido al registro r.

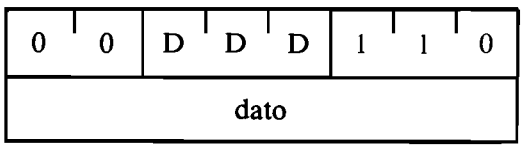
Ciclos: 2
Estados: 7
Direccionamiento: indirecto de reg.
Señalizadores: ninguno

MOV M, r (Transfiere a memoria)
((H) (L)) ← (r)
El contenido del registro r es transferido a la posición de memoria cuya dirección está en los registros H y L.



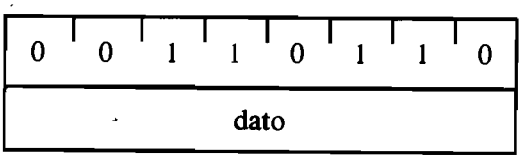
Ciclos: 2
Estados: 7
Direccionamiento: indirecto de reg.
Señalizadores: ninguno

MVI r, dato (Transferencia inmediata)
(r) ← (byte 2)
El contenido del byte 2 de la instrucción es transferido al registro r.



Ciclos: 2
Estados: 7
Direccionamiento: inmediato
Señalizadores: ninguno

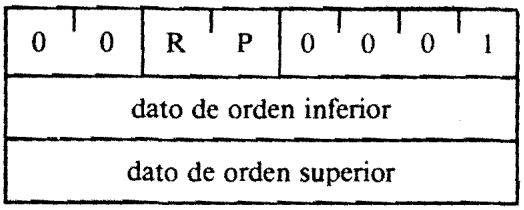
MVI M, dato (Transfiere a memoria inmediata)
((H) (L)) ← (byte 2)
El contenido del byte 2 de la instrucción es transferido a la posición de memoria cuya dirección está en los registros H y L.



Ciclos: 3
Estados: 10
Direccionamiento: inmed./indirecto de reg.
Señalizadores: ninguno

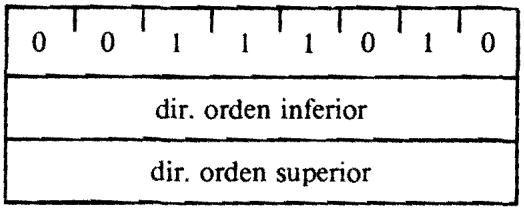
LXI rp, dato 16 (Carga registro par inmediato)
(rh) ← (byte 3),
(rl) ← (byte 2)
El byte 3 de la instrucción es transferido al registro de orden superior (rh) del registro par rp. El byte 2 de la instrucción es

transferido al registro de orden inferior (rl) del registro par rp.



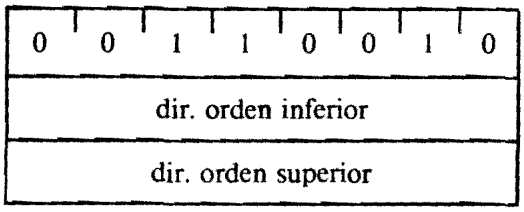
Ciclos: 3
Estados: 10
Direccionamiento: inmediato
Señalizadores: ninguno

LDA addr (Carga acumulador directo)
(A) ← ((byte 3)(byte 2))
El contenido de la posición de memoria, cuya dirección está especificada en los bytes 2 y 3 de la instrucción, es transferido al registro A.



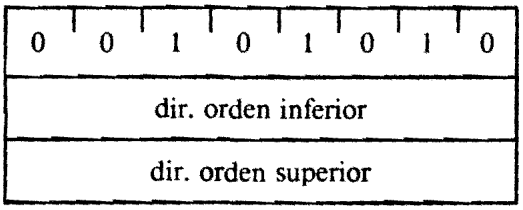
Ciclos: 4
Estados: 13
Direccionamiento: directo
Señalizadores: ninguno

STA addr (Almacena acumulador directo)
((byte 3)(byte 2)) ← (A)
El contenido del acumulador es transferido a la posición de memoria cuya dirección está especificada en los bytes 2 y 3 de la instrucción.



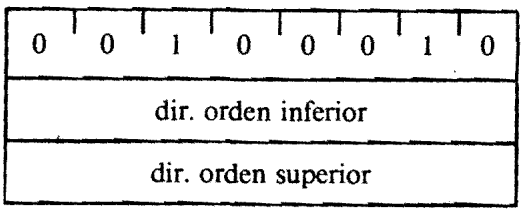
Ciclos: 4
Estados: 13
Direccionamiento: directo
Señalizadores: ninguno

LHLD addr (Carga H y L directo)
(L) ← ((byte 3)(byte 2))
(H) ← ((byte 3)(byte 2) + 1)
El contenido de la posición de memoria, cuya dirección está especificada en los bytes 2 y 3 de la instrucción, es transferido al registro L. El contenido de la posición de memoria en la dirección siguiente es transferido al registro H.



Ciclos: 5
Estados: 16
Direccionamiento: directo
Señalizadores: ninguno

SHLD addr (Almacena H y L directo)
((byte 3)(byte 2)) ← (L)
((byte 3)(byte 2) + 1) ← (H)
El contenido del registro L es transferido a la posición de memoria, cuya dirección está especificada en los bytes 2 y 3. El contenido del registro H es transferido a la posición de memoria siguiente.



Ciclos: 5
Estados: 16
Direccionamiento: directo
Señalizadores: ninguno

LDAX rp (Carga indirecto el acumulador)
(A) ← ((rp))
El contenido de la posición de memoria, cuya dirección está en el registro par rp, es transferido al registro A. Nota: solamente pueden ser especificados los registros rp = B (registros B y C) o rp = D (registros D y E).

0	0	R	P	1	0	1	0

Ciclos: 2
Estados: 7
Direccionamiento: indirecto de reg.
Señalizadores: ninguno

STAX rp (Almacena indirecto el acumulador)
((rp)) ← (A)
El contenido del registro A es transferido a la posición de memoria cuya dirección está en el registro par rp. Nota: solamente pueden ser especificados los registros rp = B (registros B y C) o rp = D (registros D y E).

0	0	R	P	0	0	1	0

Ciclos: 2
Estados: 7
Direccionamiento: reg. indirecto
Señalizadores: ninguno

XCHG (Intercambia H y L con D y E)
(H) ↔ (D)
(L) ↔ (E)
Los contenidos de los registros H y L son intercambiados con los contenidos de los registros D y E.

1	1	1	0	1	0	1	1

Ciclos: 1
Estados: 4
Direccionamiento: registro
Señalizadores: ninguno

PROBLEMAS RESUELTOS

8.35. El código de op para la instrucción MOV *A,M* es _____ (hex). Esta operación transfiere datos _____ (de una posición de memoria, del registro *A*) _____ (a una posición de memoria, al registro *A*). La posición de memoria se identifica por el contenido del _____ (registro par *HL*, segundo y tercer bytes de la instrucción). La instrucción MOV *A,M* utiliza _____ (número) ciclos máquina y _____ (número) estados *T*. La instrucción MOV *A,M* utiliza direccionamiento _____ (directo, indirecto de registro).

Solución:

De acuerdo con la Figura 8.17 y la descripción de Intel «transfiere de memoria» el código de op para la instrucción MOV *A,M* es 7EH. Esta operación transfiere datos de una posición de memoria al registro *A*. La posición de memoria está identificada por el contenido del registro par *HL*. La instrucción MOV *A,M* utiliza dos ciclos máquina y siete estados *T* para su ejecución. La instrucción MOV *A,M* utiliza direccionamiento indirecto de registro.

8.36. El código de op para la instrucción MOV *M,C* es _____ (binario). Esta operación transfiere datos desde _____ (una posición de memoria, el registro *C*). La posición de memoria está identificada por el contenido del _____ (registro par *HL*, segundo y tercer bytes de la instrucción). La instrucción MOV *M,C* utiliza _____ (número) ciclos máquina y _____ (número) estados *T*. La instrucción MOV *M,C* utiliza direccionamiento _____ (inmediato, indirecto de registro).

Solución:

De acuerdo con la Figura 8.17 y la descripción de Intel «transfiere a memoria», el código de op para la instrucción MOV *M,C* es 01110001_2. Esta operación transfiere datos desde el registro *C* a una posición de memoria. La posición de memoria está identificada por el contenido del registro par *HL*. La instrucción MOV *M,C* utiliza dos ciclos máquina y siete estados *T* durante la ejecución. La instrucción MOV *M,C* utiliza direccionamiento indirecto de registro.

8.37. El código de op del 8085 para la instrucción MVI *H* esbozada en la Figura 8.19 es _____ (binario).

Solución:

De acuerdo con la Figura 8.17, el código de op del 8085 para la instrucción MVI *H* esbozada en la Figura 8.19 es 00100110_2 ó 26H.

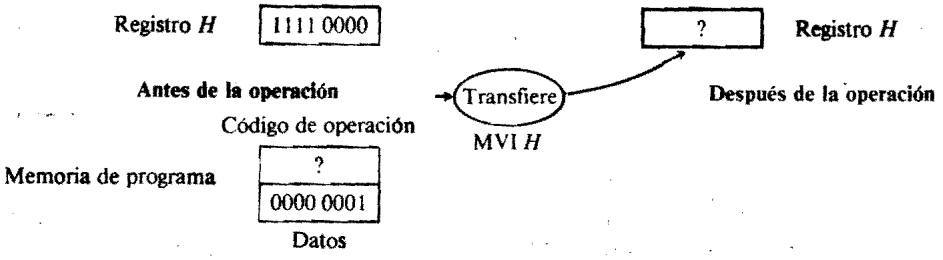


Figura 8.19.

8.38. El contenido del registro *H* después de la operación MVI *H* mostrada en la Figura 8.19 es _____ (8 bits).

Solución:

La instrucción MVI *H* utiliza el modo de direccionamiento inmediato. El dato inmediato 00000001 de la memoria del programa es transferido al registro *H* durante esta operación de transferencia.

8.39. La instrucción MVI *H* mostrada en la Figura 8.19 utiliza _____ (número) ciclos máquina y _____ (número) estados *T* para ejecutar la transferencia de datos.

Solución:

De acuerdo con la descripción de Intel «transferencia inmediata», la instrucción MVI *H* mostrada en la Figura 8.19 utiliza dos ciclos máquina y siete estados *T* para ejecutar la transferencia de datos.

8.40. La instrucción SHLD mostrada en la Figura 8.20 utiliza el modo de direccionamiento _____ (directo, inmediato).

Solución:

De acuerdo con la descripción de Intel, la instrucción SHLD mostrada en la Figura 8.20 utiliza direccionamiento directo.

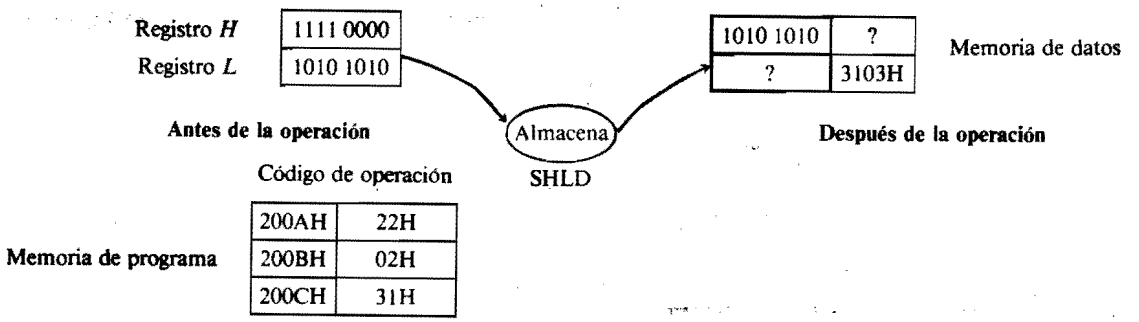


Figura 8.20.

8.41. Después de la operación de almacenar ilustrada en la Figura 8.20, la posición de la memoria de datos _____ (hex) contiene 10101010_2, mientras que la posición 3103H contiene _____ (8 bits).

Solución:

La instrucción SHLD utiliza direccionamiento indirecto, así que la dirección de 16 bits es ensamblada con el segundo y tercer bytes de la memoria de programa. Por tanto, después de la operación de almacenar ilustrada en la Figura 8.20, la posición 3103H de la memoria de datos contiene 11110000_2, mientras que la posición 3102H contiene 10101010_2.

8.7. INSTRUCCIONES ARITMETICAS DEL 8080/8085

Un resumen de las instrucciones aritméticas del microprocesador 8080/8085 se muestra en la Figura 8.21. Estas instrucciones incluyen las operaciones de suma, suma con arrastre, resta, resta con préstamo, incrementación, decrementación y ajuste decimal. Sólo los nemotécnicos y códigos de operación de las instrucciones aritméticas se dan en la Figura 8.21 en la página 283.

Como en la sección anterior, el *grupo aritmético* de instrucciones se detallará utilizando el propio formato de Intel Corporation. Este grupo de instrucciones realiza operaciones aritméticas sobre datos en registros y memoria. A menos que se indique otra cosa, todas las instrucciones de este grupo afectan a los señalizadores de cero (*Z*), de signo (*S*), de paridad (*P*), de arrastre (*CY*) y de arrastre auxiliar (*AC*). Todas las operaciones de resta son realizadas vía aritmética de complemento a 2 y ponen el señalizador de arrastre a 1 para indicar un préstamo y a 0 para indicar no préstamo. Acudir a la Figura 8.18 para los significados de las abreviaturas utilizadas en las siguientes descripciones del *Manual del Usuario* de Intel.

ADD r (Suma registros)

(A) ← (A) + (r)

El contenido del registro r es sumado al contenido del acumulador. El resultado se coloca en el acumulador.

1	0	0	0	0	S	S	S

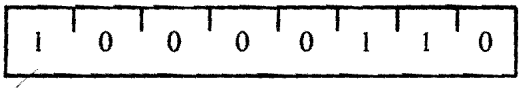
Ciclos: 1
Estados: 4
Direccionamiento: registro
Señalizadores: Z,S,P,CY,AC

ADD M (Suma memoria)

(A) ← (A) + ((H) (L))

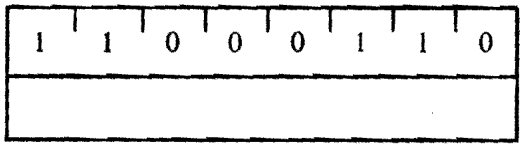
El contenido de la posición de memoria

cuya dirección está contenida en los registros H y L es sumado al contenido del acumulador. El resultado se coloca en el acumulador



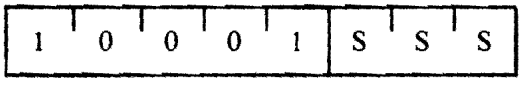
Ciclos: 2
Estados: 7
Direccionamiento: indirecto de reg.
Señalizadores: Z,S,P,CY,AC

ADI dato (Suma inmediata)
(A) ← (A) + (byte 2)
El contenido del segundo byte de la instrucción es sumado al contenido del acumulador. El resultado se coloca en el acumulador.



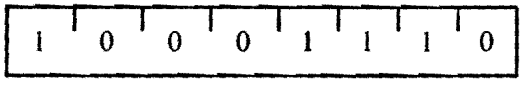
Ciclos: 2
Estados: 7
Direccionamiento: inmediato
Señalizadores: Z,S,P,CY,AC

ADC r (Suma registro con arrastre)
(A) ← (A) + (r) + (CY)
El contenido del registro r y el contenido del bit de arrastre son sumados al contenido del acumulador. El resultado se coloca en el acumulador.



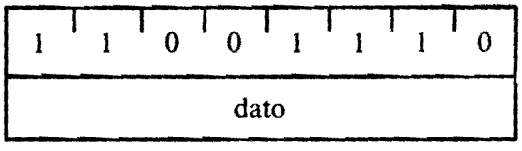
Ciclos: 1
Estados: 4
Direccionamiento: registro
Señalizadores: Z,S,P,CY,AC

ADC M (Suma memoria con arrastre)
(A) ← (A) + ((H) (L)) + (CY)
El contenido de la posición de memoria cuya dirección está contenida en los registros H y L y el contenido del señalizador CY son sumados al acumulador. El resultado se coloca en el acumulador



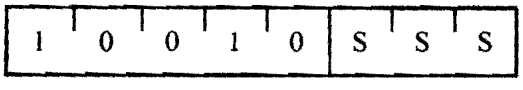
Ciclos: 2
Estados: 7
Direccionamiento: indirecto de reg.
Señalizadores: Z,S,P,CY,AC

ACI dato (Suma inmediata con arrastre)
(A) ← (A) + (byte 2) + (CY)
El contenido del segundo byte de la instrucción y el contenido del señalizador CY son sumados al contenido del acumulador. El resultado se coloca en el acumulador.



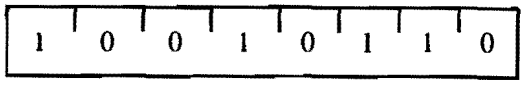
Ciclos: 2
Estados: 7
Direccionamiento: inmediato
Señalizadores: Z,S,P,CY,AC

SUB r (Resta registro)
(A) ← (A) − (r)
El contenido del registro r es restado del contenido del acumulador. El resultado se coloca en el acumulador.



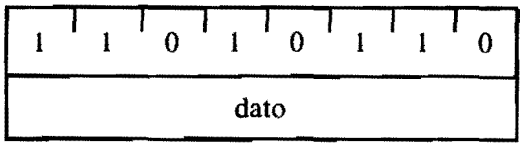
Ciclos: 1
Estados: 4
Direccionamiento: registro
Señalizadores: Z,S,P,CY,AC

SUB M (Resta memoria)
(A) ← (A) – ((H) (L))
El contenido de la **posición de memoria** cuya dirección está contenida en los registros H y L es restado del contenido del acumulador. El resultado se coloca en el acumulador



Ciclos: 2
Estados: 7
Direccionamiento: indirecto de reg.
Señalizadores: Z,S,P,CY,AC

SUI dato (Resta inmediata)
(A) ← (A) – (byte 2)
El contenido del segundo byte de la instrucción es restado del contenido del acumulador. El resultado se coloca en el acumulador.

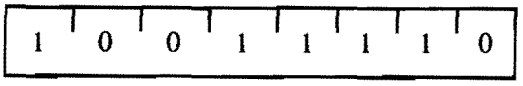


Ciclos: 2
Estados: 7
Direccionamiento: inmediato
Señalizadores: Z,S,P,CY,AC

SBB r (Resta registro con préstamo)
(A) ← (A) – (r) – (CY)
El contenido del registro r y el contenido del señalizador CY son ambos restados del acumulador. El resultado se coloca en el acumulador.

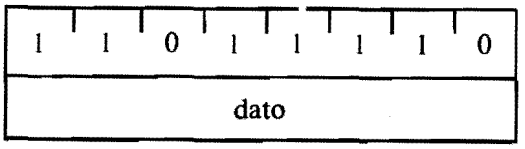
Ciclos: 1
Estados: 4
Direccionamiento: registro
Señalizadores: Z,S,P,CY,AC

SBB M (Resta memoria con préstamo)
(A) ← (A) – ((H) (L)) – (CY)
El contenido de la posición de memoria cuya dirección está contenida en los registros H y L y el contenido del señalizador CY son ambos restados del acumulador. El resultado se coloca en el acumulador



Ciclos: 2
Estados: 7
Direccionamiento: indirecto de reg.
Señalizadores: Z,S,P,CY,AC

SBI dato (Resta inmediata con préstamo)
(A) ← (A) – (byte 2) – (CY)
El contenido del segundo byte de la instrucción y el contenido del señalizador CY son ambos restados del acumulador. El resultado se coloca en el acumulador.



Ciclos: 2
Estados: 7
Direccionamiento: inmediato
Señalizadores: Z,S,P,CY,AC

INR r (Incrementa registro)
(r) ← (r) + 1
El contenido del registro r es incrementado en uno. Nota: Todos los señalizadores de condición **excepto CY** están afectados.

Ciclos: 1
Estados: 4 (8085), 5 (8080)
Direccionamiento: registro
Señalizadores: Z,S,P,AC

INR M (Incrementa memoria)
((H) (L) ← ((H) (L)) + 1
El contenido de la posición de memoria cuya dirección está contenida en los registros H y L es incrementado en uno. Nota: Todos los señalizadores de condición **excepto CY** son afectados.

0	0	1	1	0	1	0	0

Ciclos: 3
Estados: 10
Direccionamiento: indirecto de reg.
Señalizadores: Z,S,P,AC

DCR r (Decrementa registro)
(r) ← (r) − 1
El contenido del registro r es decrementado en uno. Nota: Todos los señalizadores de condición **excepto CY** son afectados.

0	0	D	D	D	1	0	1

Ciclos: 1
Estados: 4 (8085), 5 (8080)
Direccionamiento: registro
Señalizadores: Z,S,P,AC

DCR M (Decrementa memoria)
((H) (L)) ← ((H) (L)) − 1
El contenido de la posición de memoria cuya dirección está contenida en los registros H y L es decrementado en uno. Nota: Todos los señalizadores de condición **excepto CY** son afectados.

0	0	1	1	0	1	0	1

Ciclos: 3
Estados: 10
Direccionamiento: indirecto de reg.
Señalizadores: Z,S,P,AC

INX rp (Incrementa registro par)
(rh) (rl) ← (rh) (rl) + 1
El contenido del registro par rp es incrementado en uno. Nota: **Ningún señalizador de condición es afectado.**

0	0	R	P	0	0	1	1

Ciclos: 1
Estados: 6 (8085), 5 (8080)
Direccionamiento: registro
Señalizadores: ninguno

DCX rp (Decrementa registro par)
(rh) (rl) ← (rh) (rl) − 1
El contenido del registro par rp es decrementado en uno. Nota: **Ningún señalizador de condición es afectado.**

0	0	R	P	1	0	1	1

Ciclos: 1
Estados: 6 (8085), 5 (8080)
Direccionamiento: registro
Señalizadores: ninguno

DAD rp (Suma registro par a H y L)
(H) (L) ← (H) (L) + (rh) (rl)
El contenido del registro par rp es sumado al contenido del registro par H y L. El resultado es colocado en el registro par H y L. Nota: **Sólo es afectado el señalizador CY.** Se pone a 1 si hay un arrastre en la suma en doble precisión; en cualquier otro caso está en reinicialización.

0	0	R	P	1	0	0	1

Ciclos: 3
Estados: 10
Direccionamiento: registro
Señalizadores: CY

DAA (Ajuste decimal acumulador)
El número de ocho bits del acumulador es ajustado para formar dos dígitos Decimales Codificados en Binario de cuatro bits, por el siguiente proceso:

Suma*

ADD	A	87
	B	80
	C	81
	D	82
	E	83
	H	84
	L	85
	M	86

ADC	A	8F
	B	88
	C	89
	D	8A
	E	8B
	H	8C
	L	8D
	M	8E

Resta*

SUB	A	97
	B	90
	C	91
	D	92
	E	93
	H	94
	L	95
	M	96

SUB	A	9F
	B	98
	C	99
	D	9A
	E	9B
	H	9C
	L	9D
	M	9E

Doble suma †

DAD	B	09
	D	19
	H	29
	SP	39

Incrementa**

INR	A	3C
	B	04
	C	0C
	D	14
	E	1C
	H	24
	L	2C
	M	34

INX	B	03
	O	13
	H	23
	SP	33

Decrementa**

DCR	A	3D
	B	05
	C	0D
	D	15
	E	1D
	H	25
	L	2D
	M	35

INX	B	0B
	O	1B
	H	2B
	SP	3B

Especial

DAA* 27

Inmediato

ADI byte	C6
ACI byte	CE
SUI byte	D6
SBI byte	DE

byte = expresión constante o lógica/aritmética que evalúa un dato de 8 bits (segundo byte de instrucciones de 2 bytes)
* = todos los señalizadores (C, Z, S, P, AC) afectados
** = todos los señalizadores excepto el de ARRASTRE son afectados (excepción: INX y DXC no afectan a los señalizadores)
† = sólo CY (ARRASTRE) afectado
Todos los nemotécnicos tienen derecho de copia de Intel Corporation 1976

Figura 8.21. Resumen de las instrucciones aritméticas del 8080/8085.

1. Si el valor de los 4 bits menos significativos del acumulador es mayor que 9 o si el señalizador AC está a 1, se suma 6 al acumulador.
2. Si el valor de los 4 bits más significativos del acumulador es ahora mayor que 9, se suma 6 a los 4 bits más significativos del acumulador.

NOTA: Todos los señalizadores son afectados.

0	0	1	0	0	1	1	1

Ciclos: 1
Estados: 4
Señalizadores: Z,S,P,CY,AC

Considerar la instrucción ADC *B* (sumar registro *B* con arrastre) listada en la Figura 8.21. Su funcionamiento se detalla en la sección ADC *r* (sumar registro con arrastre). Intel Corporation describe esta instrucción en forma simbólica como

$$(A) \leftarrow (A) + (r) + (CY)$$

Esto significa que el contenido de *r* (registro *B* en este ejemplo) y el contenido del bit de arrastre (*CY*) son sumados al contenido del acumulador (registro *A*). La suma es depositada en el acumulador. La descripción de Intel muestra que esta operación utiliza una instrucción de 1 byte. La descripción también muestra que la instrucción ADC *B* es ejecutada en un ciclo máquina y cuatro estados *T*. Se utiliza el modo de direccionamiento de registro, y todos los señalizadores (*Z, S, P, CY* y *AC*) son afectados por la ejecución de esta instrucción.

La instrucción ADC *B* (suma registro *B* con arrastre) está ejecutándose en la Figura 8.22. El contenido del registro *A* (10000000 en este ejemplo) se suma al del registro *B* (10000000) y al del bit de arrastre (puesto a 1 en este ejemplo). El problema puede ser el siguiente:

```
  1000 0000   registro A
  1000 0000   registro B
+        1    bit de arrastre
-------------
1 0000 0001   suma
```

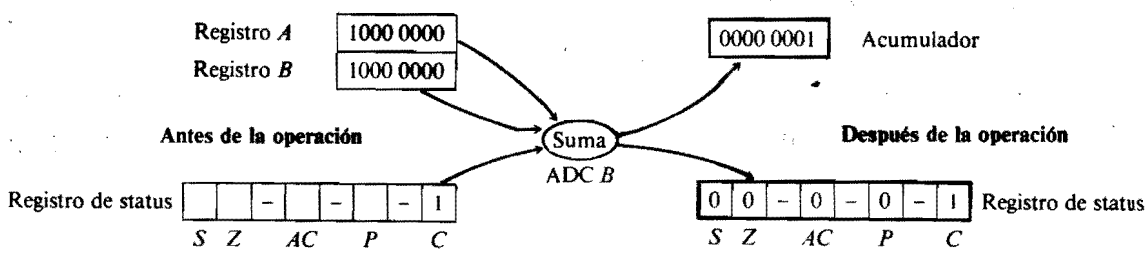


Figura 8.22. La instrucción suma con arrastre (ADC *B*).

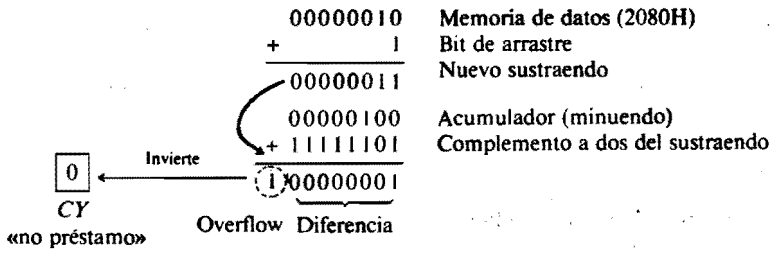
La suma es igual a 1 0000 0001. Los 8 bits menos significativos (0000 0001) son depositados en el acumulador. Esto se muestra en la Figura 8.22. El overflow (rebose) del acumulador (el bit más significativo de la suma) pone el señalizador de arrastre (*CY*) a 1 después de la operación. Los bits de signo, cero, arrastre auxiliar y de paridad par son puestos a 0 después que se ejecuta la instrucción ADC *B* en la Figura 8.22.

A continuación considerar la utilización de la instrucción del 8085 SBB *M* (resta memoria con préstamo) del listado de la Figura 8.21. Su funcionamiento se detalla en la sección SBB *M*. Intel describe esta instrucción en forma simbólica como

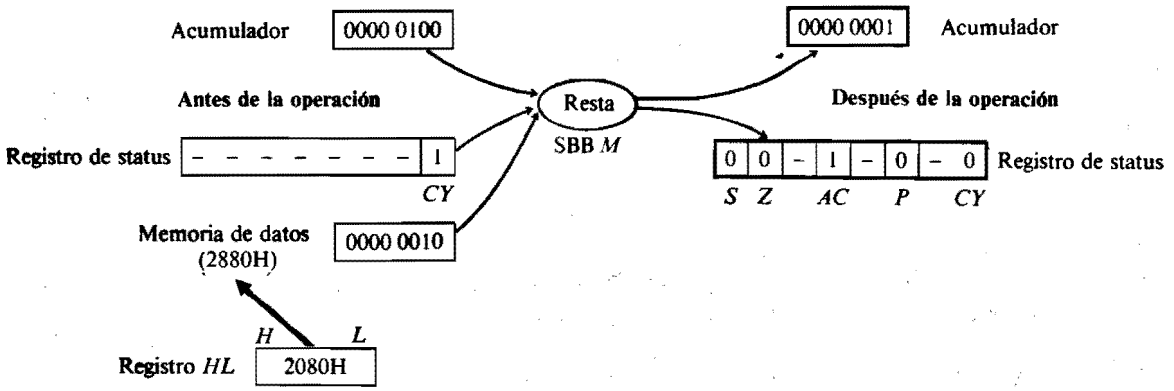
$$(A) \leftarrow (A) - ((H)(L)) - (CY)$$

Esto significa que el contenido de la posición de memoria señalada por el registro *HL* es restado del contenido del acumulador. El bit de arrastre (*CY*) del registro de status es también restado del contenido del acumulador. La diferencia es depositada en el acumulador después de que se ejecuta la instrucción. Intel Corporation también nota que SBB *M* es una instrucción de 1 byte que utiliza direccionamiento indirecto de registro. Para su ejecución necesita dos ciclos máquina y siete estados *T*. Todos los señalizadores son afectados por la instrucción SBB *M*.

La instrucción SBB *M* (resta memoria con préstamo) se está ejecutando en la Figura 8.23*b*. El acumulador contiene 00000100, el bit de arrastre es 1, y la posición de la memoria de datos señalada por el registro *HL* contiene 00000010. La operación de la instrucción SBB *M* con los datos está detallada en la Figura 8.23*a*. Los dos sustraendos son sumados primero (00000010 + 1 = 00000011). El nuevo sustraendo combinado de 00000011 se convierte entonces a su forma en complemento a 2 (11111101). El minuendo (00000100) se suma entonces al



(a) Solución para el problema de resta con préstamo



(b) Instrucción de resta con préstamo (SBB *M*)

Figura 8.23.

complemento a 2 del sustraendo (11111101), dando 1 0000 0001. Los 8 bits menos significativos forman la diferencia (00000001 en este ejemplo), mientras que el «overflow» de 1 es invertido por el microprocesador poniendo a 0 el señalizador de arrastre (*CY*). El señalizador de reinicialización significa «no préstamo» o que el minuendo era mayor que los sustraendos combinados en este problema.

Las instrucciones de suma **con arrastre** y resta con préstamo no son utilizadas con la suma o resta regular de un solo byte. Estas son instrucciones especiales utilizadas cuando se suman o restan 2 o más bytes de datos. La forma en que se utilizan estas instrucciones en la programación real se demostrará en el capítulo sobre programación.

PROBLEMAS RESUELTOS

8.42. El código de op para la instrucción ADD *C* es _____ (hex). Esta operación suma el contenido del registro _____ (letra) al contenido del acumulador. La instrucción ADD *C* utiliza direccionamiento _____ (de registro, indirecto de registro).

Solución:

De acuerdo con la Figura 8.21 y la descripción de Intel de «suma registro», el código de op para la instrucción ADD *C* es 81H. Esta operación suma el contenido del registro *C* al contenido del acumulador. La instrucción ADD *C* utiliza direccionamiento de registro.

8.43. El código de op para la instrucción INX *H* es _____ (hex). Esta operación _____ (decrementa, incrementa) el registro par _____ (*BC*, *HL*). La instrucción INX *H* afecta a _____ (todos, ninguno de los) señalizadores.

Solución:

De acuerdo con la Figura 8.21 y la descripción de Intel de «incrementa registro par», el código de op para la instrucción INX *H* es 23H. Esta operación incrementa el registro par *HL*. La instrucción INX *H* no afecta a los señalizadores.

8.44. El código de op del 8085 para la instrucción DAD *B*, esbozada en la Figura 8.24, es _____ (hex).

Solución:

De acuerdo con la Figura 8.21, el código de op para la instrucción DAD *B* esbozada en la Figura 8.24 es 09H.

8.45. La instrucción DAD *B* de la Figura 8.24 suma el contenido de _____ (8, 16) bits del registro par *HL* al contenido del registro par _____ (dos letras).

Solución:

De acuerdo con la descripción de Intel, la instrucción DAD *B* utilizada en la Figura 8.24 suma el contenido de 16 bits del registro par *HL* al contenido del registro par *BC*.

8.46. Después de la operación DAD *B* de la Figura 8.24, el registro par *HL* contendrá _____ (hex).

Solución:

Sumar el hexadecimal FFFF + 0001 = 1 0000H. Por tanto después de la operación DAD *B* de la Figura 8.24, el registro par *HL* contendrá 0000H.

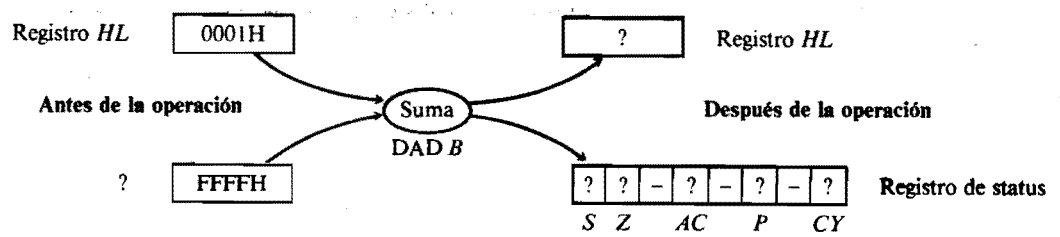


Figura 8.24.

8.47. Listar la condición de los señalizadores del registro de status después que se haya ejecutado la instrucción DAD *B* en la Figura 8.24.

Solución:

Sólo es afectado el señalizador de arrastre por la operación DAD *B*. Por tanto, las condiciones de los señalizadores de signo, cero, arrastre auxiliar y paridad par de la Figura 8.24 no son predecibles. El señalizador de arrastre (*CY*) se pondrá a 1 a causa del «overflow» o arrastre durante la suma (FFFF + 0001 = 1 0000H).

8.48. Acudir a la Figura 8.25. ¿Es el número del acumulador antes de la operación DAA un número «legal» BCD (decimal codificado binario 8421)?

Solución:

El número 0000 1110 no es un número BCD «legal» porque el 1110 no tiene equivalente BCD.

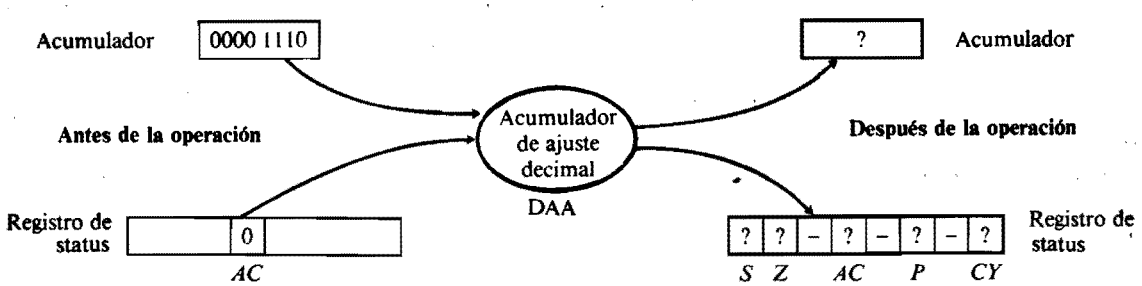


Figura 8.25.

8.49. El contenido del acumulador de la Figura 8.25 después de la operación DAA será _____ (BCD), que es igual a _____ en decimal.

Solución:

De acuerdo con la descripción Intel de la instrucción DAA (ajuste decimal del acumulador), el contenido del acumulador de la Figura 8.25 después de la operación DAA será $0001\ 0100_{BCD}$, que es igual a 14 en decimal. Como 1110 es mayor que 9, 0110 (6_{10}) se sumó a este número BCD «ilegal», dando $0001\ 0100_{BCD}$ (14_{10}).

8.50. Listar la condición de los señalizadores del registro de status después de la operación DAA mostrada en la Figura 8.25.

Solución:

En la Figura 8.25 la operación DAA hacía que 0110 se sumase a 0000 1110, dando 0001 0100. Durante esta operación los señalizadores se afectaron de la siguiente manera:
Señalizador de signo (S) = 0
Señalizador de arrastre auxiliar (AC) = 1
Señalizador de arrastre (CY) = 0
Señalizador de cero (Z) = 0
Señalizador de paridad par (P) = 1

8.51. La instrucción SUB E de la Figura 8.26 resta el contenido del registro _____ (letra) del contenido del acumulador.

Solución:

La instrucción SUB E de la Figura 8.26 resta el contenido del registro E del contenido del acumulador.

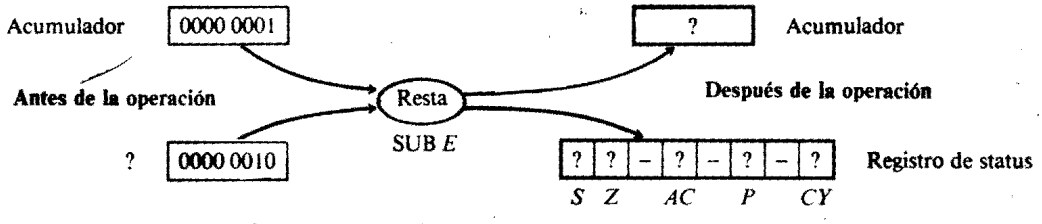


Figura 8.26.

8.52. Después de que se ha ejecutado la instrucción SUB E en la Figura 8.26, el contenido del acumulador será _____ (8 bits).

Solución:

La MPU cambia el sustraendo (00000010 en el registro E) a su forma en complemento a 2 (11111110) y suma como sigue:

0000 0001	minuendo
+ 1111 1110	sustraendo en complemento a 2
0 1111 1111	diferencia

Los 8 bits menos significativos de la diferencia (1111 1111) se depositan en el acumulador. En forma decimal, el problema es $1 - 2 = -1$. El -1_{10} es igual a 1111 1111 en notación de complemento a 2.

8.53. Acudir a la Figura 8.26. Listar la condición de los señalizadores del registro de status después que se haya ejecutado la instrucción SUB E.

Solución:

Después de la operación SUB E de la Figura 8.26, los señalizadores están en las siguientes condiciones:
Señalizador de signo (S) = 1
Señalizador de arrastre auxiliar (AC) = 1

Señalizador de arrastre (CY) = 1
Señalizador de cero (Z) = 0
Señalizador de paridad par (P) = 1
La MPU cambia el sustraendo (000000100 en el registro E) a su forma en complemento a 2 (11111110) y suma como sigue:

```
                0000 0001   minuendo
              + 1111 1110   sustraendo en complemento a 2
              -------------
[1] <-invierte- 0 1111 1111 diferencia
 CY
```

Notar que el bit de overflow (bit más significativo) en la diferencia es un 0 que está invertido y colocado en el señalizador de arrastre (CY) del registro de status. Un señalizador de arrastre puesto a 1 después de la operación de resta significa que la diferencia es un número en complemento a 2 y que el sustraendo era mayor que el minuendo.

8.8. INSTRUCCIONES LOGICAS DEL 8080/8085

Un resumen de las instrucciones lógicas del microprocesador 8080/8085 se muestra en la Figura 8.27. Estas incluyen las instrucciones AND, OR, XOR, comparar, desplazamiento circular y complementación. Solamente los códigos de op y nemotécnicos de Intel están listados para cada instrucción en la Figura 8.27.

Como en la sección anterior, el *grupo* de instrucciones *lógicas* se detallará utilizando el propio formato de Intel Corporation. Este grupo de instrucciones realiza operaciones lógicas (booleanas) sobre los datos de los registros, memoria y señalizadores. Acudir a la Figura 8.18 para los significados de las abreviaturas utilizadas en las siguientes descripciones del *Manual del Usuario* del Intel.

ANA r (AND registro)
(A) ← (A) ∧ (r)
El contenido del registro r realiza la operación lógica AND con el contenido del acumulador. El resultado se coloca en el acumulador. **El señalizador CY es puesto a 0 y AC a 1 (8085). El señalizador CY es puesto a 0 y el AC a 1 para la operación OR de los bits 3 de los operandos (8080).**

1	0	1	0	0	S	S	S

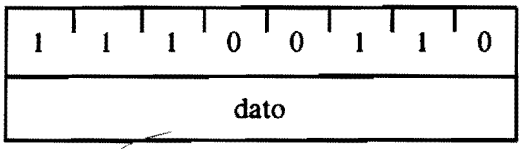
Ciclos: 1
Estados: 4
Direccionamiento: registro
Señalizadores: Z,S,P,CY,AC

ANA M (AND memoria)
(A) ← (A) ∧ ((H) (L))
El contenido de la posición de memoria cuya dirección está contenida en los registros H y L realiza la operación lógica AND con el contenido del acumulador. El resultado se coloca en el acumulador. **El señalizador CY es puesto a 0 y AC a 1 (8085). El señalizador CY es puesto a 0 y AC a 1 para la operación OR de los bits 3 de los operandos (8080).**

1	0	1	0	0	1	1	0

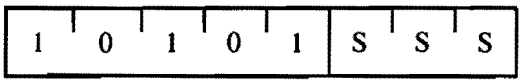
Ciclos: 2
Estados: 7
Direccionamiento: indirecto de reg.
Señalizadores: Z,S,P,CY,AC

ANI dato (AND inmediato)
(A) ← (A) ∧ (byte 2)
El contenido del segundo byte de la instrucción realiza la operación lógica AND con el contenido del acumulador. El resultado se coloca en el acumulador. **El señalizador CY es puesto a 0 y AC a 1 (8085). El señalizador CY es puesto a 0 y AC a 1 para la OR de los bits 3 de los operandos (8080).**



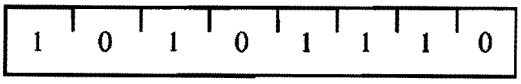
Ciclos: 2
Estados: 7
Direccionamiento: inmediato
Señalizadores: Z,S,P,CY,AC

XRA r (OR exclusivo registro)
(A) ← (A) ∀ (r)
El contenido del registro r realiza la operación OR exclusiva con el contenido del acumulador. El resultado se coloca en el acumulador. **Los señalizadores CY y AC son puestos a 0.**



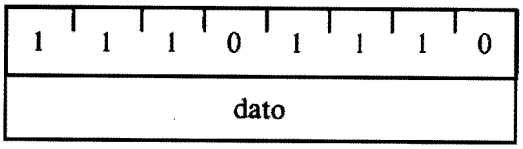
Ciclos: 1
Estados: 4
Direccionamiento: registro
Señalizadores: Z,S,P,CY,AC

XRA M (OR exclusiva memoria)
(A) ← (A) ∀ ((H) (L))
El contenido de la posición de memoria cuya dirección está contenida en los registros H y L realiza la operación OR exclusiva con el contenido del acumulador. El resultado se coloca en el acumulador. **Los señalizadores CY y AC son puestos a 0.**



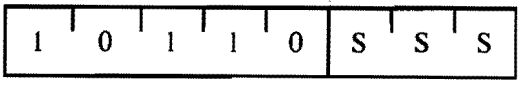
Ciclos: 2
Estados: 7
Direccionamiento: indirecto de reg.
Señalizadores: Z,S,P,CY,AC

XRI dato (OR exclusiva inmediata)
(A) ← (A) ∀ (byte 2)
El contenido del segundo byte de la instrucción realiza la operación OR exclusiva con el contenido del acumulador. El resultado se coloca en el acumulador. **Los señalizadores CY y AC son puestos a 0.**



Ciclos: 2
Estados: 7
Direccionamiento: inmediato
Señalizadores: Z,S,P,CY,AC

ORA r (OR registro)
(A) ← (A) ∨ (r)
El contenido del registro r realiza la operación OR inclusiva con el contenido del acumulador. El resultado se coloca en el acumulador. **Los señalizadores CY y AC son puestos a 0.**



Ciclos: 1
Estados: 4
Direccionamiento: registro
Señalizadores: Z,S,P,CY,AC

ORA M (OR memoria)
(A) ← (A) ∨ ((H) (L))
El contenido de la posición de memoria cuya dirección está contenida en los regis-

tros H y L realiza la operación OR inclusiva con el contenido del acumulador. El resultado se coloca en el acumulador. **Los señalizadores CY y AC son puestos a 0.**

1	0	1	1	0	1	1	0

Ciclos: 2
Estados: 7
Direccionamiento: indirecto de reg.
Señalizadores: Z,S,P,CY,AC

ORI dato (OR inmediato)
(A) ← (A) ∨ (byte 2)
El contenido del segundo byte de la instrucción realiza la operación OR inclusiva con el del contenido del acumulador. El resultado se coloca en el acumulador. **Los señalizadores CY y AC son puestos a 0.**

1	1	1	1	0	1	1	0
dato							

Ciclos: 2
Estados: 7
Direccionamiento: inmediato
Señalizadores: Z,S,P,CY,AC

CMP r (Compara registro)
(A) – (r)
El contenido del registro r es restado del acumulador. El acumulador permanece inalterado. Los señalizadores de condición son inicializados según el resultado de la sustracción. **El señalizador Z se pone a 1 si (A) = (r). El señalizador CY se pone a 1 si (A) < (r).**

1	0	1	1	1	S	S	S

Ciclos: 1
Estados: 4
Direccionamiento: registro
Señalizadores: Z,S,P,CY,AC

CMP M (Compara memoria)
(A) – ((H) (L))
El contenido de la posición de memoria cuya dirección está contenida en los registros H y L es restado del acumulador. El resultado permanece inalterado. Los señalizadores de condición se ponen a 1 según el resultado de la resta. **El señalizador Z se pone a 1 si (A) = ((H) (L)). El señalizador CY se pone a 1 si (A) < ((H) (L)).**

1	0	1	1	1	1	1	0

Ciclos: 2
Estados: 7
Direccionamiento: indirecto de reg.
Señalizadores: Z,S,P,CY,AC

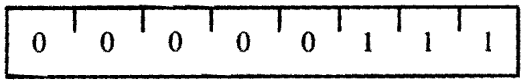
CPI dato (Comparación inmediata)
(A) – (byte 2)
El contenido del segundo byte de la instrucción es restado del acumulador. Los señalizadores de condición se ponen a 1 según el resultado de la resta. **El señalizador Z se pone a 1 si (A) = (byte 2). El señalizador CY se pone a 1 si (A) < (byte 2).**

1	1	1	1	1	1	1	0
dato							

Ciclos: 2
Estados: 7
Direccionamiento: inmediato
Señalizadores: Z,S,P,CY,AC

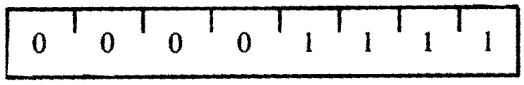
RLC (Desplazamiento circular a la izquierda)
$(A_{n+1}) \leftarrow (A_n)$; $(A_0) \leftarrow (A_7)$
$(CY) \leftarrow (A_7)$
El contenido del acumulador es desplazado a la izquierda una posición. El bit de orden inferior y el señalizador CY contienen ahora el valor del bit de orden más elevado antes de la operación de despla-

zamiento. **Sólo es afectado el señalizador CY.**



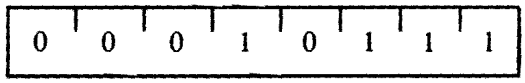
Ciclos: 1
Estados: 4
Señalizadores: CY

RRC (Desplazamiento circular a la derecha)
$(A_n) \leftarrow (A_{n+1})$; $(A_7) \leftarrow (A_0)$
$(CY) \leftarrow (A_0)$
El contenido del acumulador es desplazado una posición a la derecha. El bit de orden superior y el señalizador CY contienen ahora el bit de orden inferior del acumulador antes de realizar la operación de desplazamiento. **Solamente está afectado el señalizador CY.**



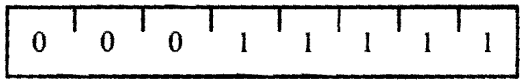
Ciclos: 1
Estados: 4
Señalizadores: CY

RAL (Desplazamiento circular a la izquierda a través del arrastre)
$(A_{n+1}) \leftarrow (A_n)$; $(CY) \leftarrow (A_7)$
$(A_0) \leftarrow (CY)$
El contenido del acumulador es desplazado a la izquierda una posición, circularmente, a través del señalizador CY. El bit de orden inferior toma el valor que tenía el bit de orden superior del acumulador antes de la operación de desplazamiento. **Solamente es afectado el señalizador CY.**



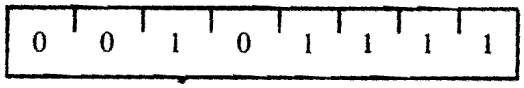
Ciclos: 1
Estados: 4
Señalizadores: CY

RAR (Desplazamiento circular a la derecha a través del arrastre)
$(A_n) \leftarrow (A_{n+1})$; $(CY) \leftarrow (A_0)$
$(A_7) \leftarrow (CY)$
El contenido del acumulador es desplazado, circularmente, a la derecha una posición a través del señalizador CY. El bit de orden superior toma el valor del señalizador de arrastre y el señalizador de arrastre toma el valor del bit de orden inferior del acumulador antes de realizar la operación de desplazamiento. **Solamente es afectado el señalizador CY.**



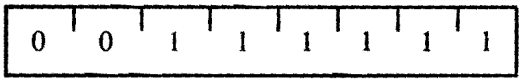
Ciclos: 1
Estados: 4
Señalizadores: CY

CMA (Complementa el acumulador)
$(A) \leftarrow (\overline{A})$
El contenido del acumulador es complementado (los bits cero se hacen 1, los bits uno se hacen 0). **Ningún señalizador está afectado.**



Ciclos: 1
Estados: 4
Señalizadores: ninguno

CMC (Complementa arrastre)
$(CY) \leftarrow (\overline{CY})$
El señalizador CY es complementado. **Ningún otro señalizador es afectado.**



Ciclos: 1
Estados: 4
Señalizadores: CY

STC (Pone a 1 el arrastre)
(CY) ← 1
El señalizador CY se pone a 1. **Los demás señalizadores no son afectados.**

0	0	1	1	0	1	1	1

Ciclos: 1
Estados: 4
Señalizadores: CY

ANA	A	A7
	B	A0
	C	A1
	D	A2
	E	A3
	H	A4
	L	A5
	M	A6
XRA	A	AF
	B	A8
	C	A9
	D	AA
	E	AB
	H	AC
	L	AD
	M	AE
ORA	A	B7
	B	B0
	C	B1
	D	B2
	E	B3
	H	B4
	L	B5
	M	B6
CMP	A	BF
	B	B8
	C	B9
	D	BA
	E	BB
	H	BC
	L	BD
	M	BE

Inmediato

ANI byte	E6
XRI byte	EE
ORI byte	F6
CPI byte	FE

Desplazamiento circular

RLC	07
RRC	0F
RAL	17
RAR	1F

Especiales

CMA	2F
STC	37
CMC	3F

Figura 8.27. Resumen de las instrucciones lógicas del 8080/8085. (Todos los nemotécnicos tienen derecho de copia © Intel Corporation, 1976.)

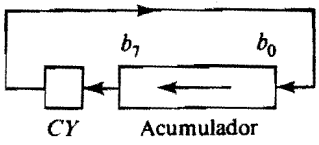
Notar el uso por Intel del símbolo ∧ para simbolizar la operación lógica AND en las descripciones de datos ANA *r*, ANA *M* y ANI. De igual forma, Intel utiliza el símbolo ∨ para denotar la operación lógica OR en las descripciones de datos ORA *r*, ORA *M* y ORI. Además, Intel utiliza

el símbolo ∀ para indicar la operación lógica XOR (OR exclusiva) en las descripciones de datos XRA *r*, XRA *M*, y XRI.

Considerar las cuatro instrucciones de desplazamiento del 8085 de la Figura 8.28. La acción de desplazamiento circular a la izquierda y a la derecha a través del arrastre (RAL, RAR) está esquematizada en la Figura 8.28*a* y *b*. Este desplazamiento circular a través del arrastre se observó cuando se utilizaron las instrucciones de desplazamiento circular en el microprocesador genérico.

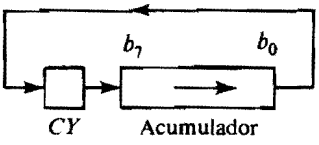
La acción de las instrucciones de desplazamiento circular a la izquierda y a la derecha (RLC, RRC) está indicada en la Figura 8.28*c* y *d*. Esta acción es algo diferente de la observada antes, a causa de que el dato no es desplazado circularmente a través de la posición de arrastre.

RAL
(Instrucción de desplazamiento circular a la izquierda a través del arrastre)



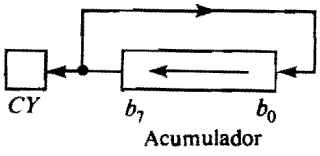
(a) Funcionamiento de la instrucción de desplazamiento circular a la izquierda a través del arrastre

RAR
(Instrucción de desplazamiento circular a la derecha a través del arrastre)



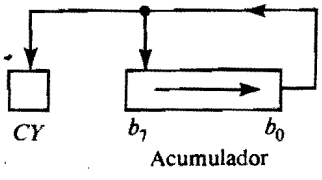
(b) Funcionamiento de la instrucción de desplazamiento circular a la derecha a través del arrastre

RLC
(Desplazamiento circular a la izquierda del acumulador)



(c) Funcionamiento de la instrucción de desplazamiento circular a la izquierda del acumulador

RRC
(Desplazamiento circular a la derecha del acumulador)



(d) Funcionamiento de la instrucción de desplazamiento circular a la derecha del acumulador

Figura 8.28.

PROBLEMAS RESUELTOS

8.54. El código de op para la instrucción ANA *M* es _____ (hex). Esta operación realiza la operación AND del contenido del acumulador con el contenido de la posición de la memoria de _____ (datos, programa) señalada por el registro par _____ (letras). Se trata de una instrucción de _____ (número) bytes.

Solución:

De acuerdo con la Figura 8.27 y la descripción «AND memoria» de Intel, el código de op para la instrucción ANA *M* es A6H. Esta operación realiza la operación AND del contenido del acumulador con el contenido de una posición de la memoria de datos señalada por el registro par *HL*. Se trata de una instrucción de 1 byte.

8.55. El código de op para la instrucción ORI es _____ (hex). Se trata de una instrucción de _____ (número) bytes que utiliza direccionamiento _____ (directo, inmediato). El contenido del acumulador realiza la operación OR con el contenido de la memoria de _____ (datos, programa).

Solución:

De acuerdo con la Figura 8.27 y la descripción de Intel de «OR inmediato», el código de op para la instrucción ORI es F6H. Se trata de una instrucción de 2 bytes que utiliza direccionamiento inmediato. El contenido del acumulador realiza la operación OR con el contenido de memoria del programa.

8.56. El contenido del acumulador de la Figura 8.29 después de la operación RLC será _____ (8 bits).

Solución:

Según el diagrama de la Figura 8.28*c*, el contenido del acumulador de la Figura 8.29 después de la operación RLC será 00000001.

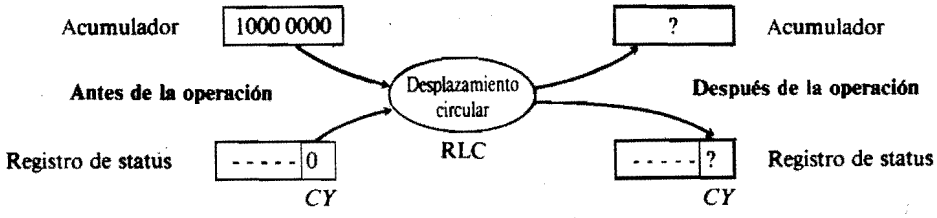


Figura 8.29.

8.57. El señalizador de arrastre estará a _____ (0, 1) después de la operación de desplazamiento circular a la izquierda (RLC) mostrado en la Figura 8.29.

Solución:

El señalizador de arrastre estará a 1 después de la operación de desplazamiento circular a la izquierda (RLC) mostrada en la Figura 8.29.

8.9. INSTRUCCIONES DE BIFURCACION DEL 8080/8085

El resumen de las instrucciones de bifurcación del microprocesador 8080/8085 se muestra en la Figura 8.30. Estas incluyen instrucciones de salto, llamada, vuelta y reinicialización. Solamente se da para cada instrucción los códigos de op y nemotécnicos de Intel en la Figura 8.30. Una

Salto		Vuelta	
JMP adr	C3	RET	C9
JNZ adr	C2	RNZ	C0
JZ adr	CA	RZ	C8
JNC adr	D2	RNC	D0
JC adr	DA	RC	D8
JPO adr	E2	RPO	E0
JPE adr	EA	RPE	E8
JP adr	F2	RP	F0
JM adr	FA	RM	F8
PCHL	E9		

Llamada		Reinicialización		
CALL adr	CD	RST	0	C7
CNZ adr	C4		1	CF
CZ adr	CC		2	D7
CNC adr	D4		3	DF
CC adr	DC		4	E7
CPO adr	E4		5	EF
CPE adr	EC		6	F7
CP adr	F4		7	FF
CM adr	FC			

adr = dirección de 16 bits (segundo y tercer bytes de instrucciones de 3 bytes)

Figura 8.30. Resumen de las instrucciones de bifurcación 8080/8085. (Todos los nemotécnicos tienen derecho de copia © Intel Corporation, 1976.)

breve descripción de los nemotécnicos se da en el resumen del repertorio de instrucciones de la Figura 8.16.

Como en las secciones anteriores, el *grupo de instrucciones de bifurcación* se detallará utilizando el formato propio de Intel Corporation. Este grupo de instrucciones altera el flujo secuencial normal del programa. Los dos tipos de instrucciones de bifurcación son la incondicional y la condicional. La incondicional simplemente realiza la operación especificada en el contador de programa. Las transferencias condicionales examinan el status de uno de los cuatro señalizadores de la MPU para determinar si se va a ejecutar la bifurcación especificada. Las condiciones que pueden ser especificadas aparecen en el *Manual del Usuario* de Intel como sigue:

CONDICION	CCC
NZ — no cero (Z = 0)	000
Z — cero (Z = 1)	001
NC — no arrastre (CY = 0)	010
C — arrastre (CY = 1)	011
PO — paridad par (P = 0)	100
PE — paridad impar (P = 1)	101
P — más (S = 0)	110
M — menos (S = 1)	111

JMP addr (Salto)

(PC) ← ((byte 3)(byte 2))

El control es transferido a la instrucción cuya dirección está especificada en los bytes 3 y 2 de la instrucción actual.

1	1	0	0	0	0	1	1
dir. orden inferior							
dir. orden superior							

Ciclos: 3
Estados: 10
Direccionamiento: inmediato
Señalizadores: ninguno

Jcondition addr (Salto condicional)
If (CCC),
(PC) ← ((byte 3)(byte 2))
Si la condición especificada es cierta, el control es transferido a la instrucción cuya dirección está especificada en los bytes 3 y 2 de la instrucción actual; en cualquier otro caso, el control continúa secuencialmente.

1	1	C	C	C	0	1	0
dir. orden inferior							
dir. orden superior							

Ciclos: 2/3 (8085), 3 (8080)
Estados: 7/10 (8085), 10 (8080)
Direccionamiento: inmediato
Señalizadores: ninguno

CALL addr (Llamada)
((SP) − 1) ← (PCH)
((SP) − 2) ← (PCL)
(SP) ← (SP) − 2
(PC) ← (byte 3)(byte 2)
Los ocho bits de orden superior de la dirección de la siguiente instrucción son trasnferidos a la posición de memoria cuya dirección es una unidad menor que el contenido del registro SP. Los ocho bits de orden inferior de la dirección de la siguiente instrucción son transferidos a la posición de memoria cuya dirección es dos unidades menor que el contenido del registro SP. El contenido del registro SP es decrementado en 2. El control es transferido a la instrucción cuya dirección está especificada en los bytes 3 y 2 de la instrucción actual.

1	1	0	0	1	1	0	1
dir. orden inferior							
dir. orden superior							

Ciclos: 5
Estados: 18 (8085), 17 (8080)
Direccionamiento: inmediato/indirecto de reg.
Señalizadores: ninguno

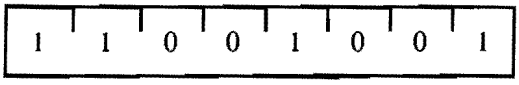
Ccondition addr (Llamada condicional)
If (CCC),
((SP) − 1) ← (PCH)
((SP) − 2) ← (PCL)
(SP) ← (SP) − 2
(PC) ← (byte 3)(byte 2)
Si la condición especificada es cierta, las acciones especificadas en la instrucción CALL (ver anterior) son realizadas; en cualquier otro caso, el control continúa secuencialmente.

1	1	C	C	C	1	0	0
dir. orden inferior							
dir. orden superior							

Ciclos: 2/5 (8085), 3/5 (8080)
Estados: 9/18 (8085), 11/17 (8080)
Direccionamiento: inmediato/indirecto de reg.
Señalizadores: ninguno

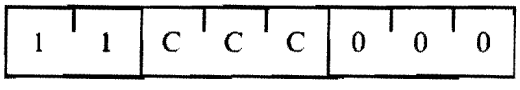
RET (Vuelta)
(PCL) ← ((SP));
(PCH) ← ((SP) + 1);
(SP) ← (SP) + 2;
El contenido de la posición de memoria cuya dirección está especificada en el re-

gistro SP es transferido a los ocho bits de orden inferior del registro PC. El contenido de la posición de memoria cuya dirección es una unidad superior al contenido del registro SP es transferido a los ocho bits de orden superior del registro PC. El contenido del registro SP es incrementado en 2.



Ciclos: 3
Estados: 10
Direccionamiento: indirecto de reg.
Señalizadores: ninguno

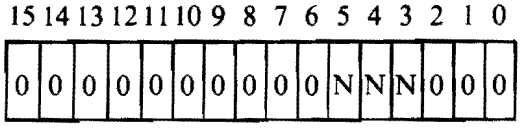
Rcondition (Vuelta condicional)
(CCC),
(PCL) ← ((SP))
(PCH) ← ((SP) + 1)
(SP) ← (SP) + 2
Si la condición especificada es verdadera, se realiza la acción especificada en la instrucción RET (ver anterior); en cualquier otro caso, el control continúa secuencialmente.



Ciclos: 1/3
Estados: 6/12 (8085),
5/11 (8080)
Direccionamiento: indirecto de reg.
Señalizadores: ninguno

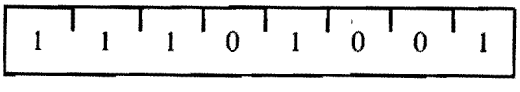
RST n (Rearrancar)
((SP) − 1) ← (PCH)
((SP) − 2) ← (PCL)
(SP) ← (SP) − 2
(PC) ← 8 * (NNN)
Los ocho bits de orden superior de la dirección de la siguiente instrucción son transferidos a la posición de memoria cuya dirección es una unidad menor que el contenido del registro SP. Los ocho bits de orden inferior de la dirección de la siguiente instrucción son transferidos a la posición de memoria cuya dirección es dos unidades menor que el contenido del registro SP. El contenido del registro SP es decrementado en dos. El control es transferido a la instrucción cuya dirección es ocho veces el contenido de NNN.

Ciclos: 3
Estados: 12 (8085),
11 (8080)
Direccionamiento: indirecto de reg.
Señalizadores: ninguno



Contador de programa después de la instrucción rearrancar

PCHL (Salta indirecto H y L – transfiere H y L a PC)
(PCH) ← (H)
(PCL) ← (L)
El contenido del registro H es transferido a los ocho bits de orden superior del registro PC. El contenido del registro L es transferido a los ocho bits de orden inferior del registro PC.



Ciclos: 1
Estados: 6 (8085), 5 (8080)
Direccionamiento: registro
Señalizadores: ninguno

Considerar el uso de la instrucción de salto condicional de la Figura 8.31. El código de op se lista como 11001010 en la Figura 8.31 para la instrucción JZ. La parte 001 del código de op (bits b_5, b_4 y b_3) especifica que la condición para la bifurcación es que el señalizador de cero debe estar a 1 ($Z = 1$). El señalizador de cero es igual a 1 en la Figura 8.31; por tanto una nueva dirección de 16 bits es transferida al contador de programa (*PC*). La dirección de orden inferior (00000000) es transferida, seguida por la dirección de orden superior (00100001). La nueva dirección del contador de programa entonces hace que la MPU salte a una nueva posición del programa.

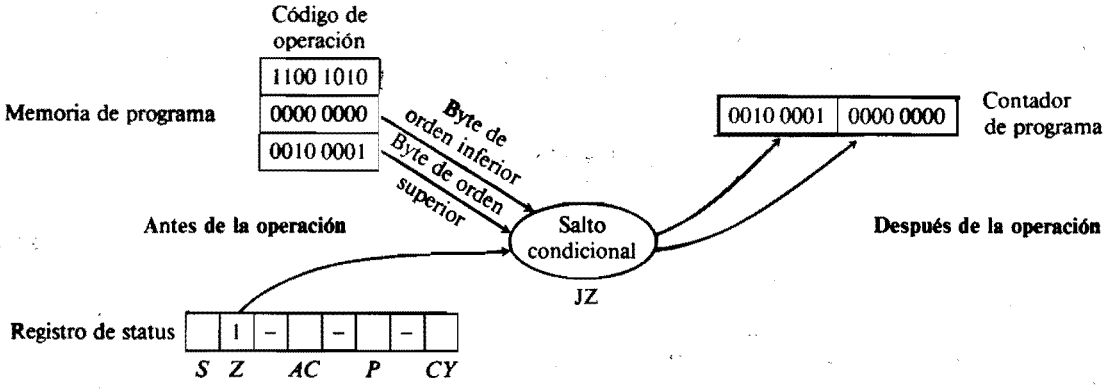


Figura 8.31. Instrucción salta si cero.

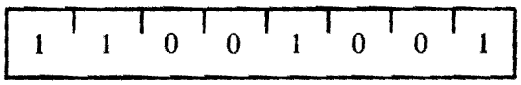
La instrucción CALL es una operación especial de bifurcación utilizada para saltar desde el programa principal a una subrutina. La instrucción CALL combina las operaciones de salto e introducción de datos en la pila. Como en el microprocesador genérico, la instrucción CALL del 8080/8085 introduce primero el contenido de 16 bits del contador de programa (*PC*) en la pila; después hace que la dirección de 16 bits de la subrutina sea transferida al contador de programa. El microprocesador entonces salta a la subrutina.

De forma distinta al microprocesador genérico, los procesadores 8080 y 8085 también tienen *instrucciones de llamada condicional.* La operación de llamada como se describió antes se ejecutará solamente si se cumple la condición especificada.

Como con el microprocesador genérico, las MPU 8080 y 8085 utilizan instrucciones de vuelta. Las instrucciones de vuelta indican al procesador la posición correcta de la memoria del programa principal una vez que finaliza una subrutina. De forma distinta al microprocesador genérico, las MPU 8080 y 8085 también tienen *instrucciones de vuelta condicional.* La operación de vuelta de subrutina se ejecutará solamente si se cumple la condición especificada.

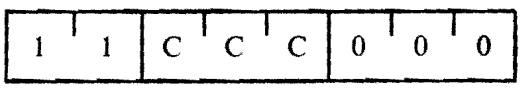
Las *instrucciones de rearranque* son instrucciones de llamada de propósito especial, diseñadas principalmente para utilizarlas con interrupciones. Igual que CALL, la instrucción de rearranque introduce el contenido del contador de programa en la pila y salta a una de ocho *direcciones predeterminadas.* Esto es diferente porque una instrucción CALL especifica inmediatamente la dirección del salto. Un código de 3 bits (bits b_5, b_4 y b_3) en el código de op de la instrucción RST especifica la dirección del salto. Por ejemplo, una instrucción RST 2 tiene el código de op 11010111 (D7H). El código de dirección 010 (decimal 2) es multiplicado por 8, dando 16_{10}, ó 10000_2. Esta es la dirección del salto (0000 0000 0001 0000_2 en este ejemplo).

gistro SP es transferido a los ocho bits de orden inferior del registro PC. El contenido de la posición de memoria cuya dirección es una unidad superior al contenido del registro SP es transferido a los ocho bits de orden superior del registro PC. El contenido del registro SP es incrementado en 2.



Ciclos: 3
Estados: 10
Direccionamiento: indirecto de reg.
Señalizadores: ninguno

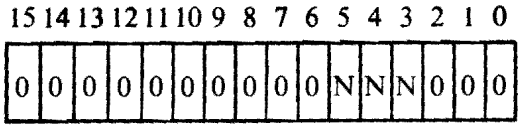
Rcondition (Vuelta condicional)
(CCC),
(PCL) ← ((SP))
(PCH) ← ((SP) + 1)
(SP) ← (SP) + 2
Si la condición especificada es verdadera, se realiza la acción especificada en la instrucción RET (ver anterior); en cualquier otro caso, el control continúa secuencialmente.



Ciclos: 1/3
Estados: 6/12 (8085), 5/11 (8080)
Direccionamiento: indirecto de reg.
Señalizadores: ninguno

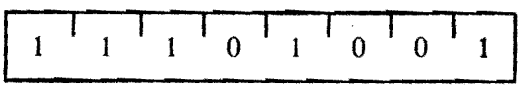
RST n (Rearrancar)
((SP) – 1) ← (PCH)
((SP) – 2) ← (PCL)
(SP) ← (SP) – 2
(PC) ← 8 * (NNN)
Los ocho bits de orden superior de la dirección de la siguiente instrucción son transferidos a la posición de memoria cuya dirección es una unidad menor que el contenido del registro SP. Los ocho bits de orden inferior de la dirección de la siguiente instrucción son transferidos a la posición de memoria cuya dirección es dos unidades menor que el contenido del registro SP. El contenido del registro SP es decrementado en dos. El control es transferido a la instrucción cuya dirección es ocho veces el contenido de NNN.

Ciclos: 3
Estados: 12 (8085), 11 (8080)
Direccionamiento: indirecto de reg.
Señalizadores: ninguno



Contador de programa después de la instrucción rearrancar

PCHL (Salta indirecto H y L – transfiere H y L a PC)
(PCH) ← (H)
(PCL) ← (L)
El contenido del registro H es transferido a los ocho bits de orden superior del registro PC. El contenido del registro L es transferido a los ocho bits de orden inferior del registro PC.



Ciclos: 1
Estados: 6 (8085), 5 (8080)
Direccionamiento: registro
Señalizadores: ninguno

Considerar el uso de la instrucción de salto condicional de la Figura 8.31. El código de op se lista como 11001010 en la Figura 8.31 para la instrucción JZ. La parte 001 del código de op (bits b_5, b_4 y b_3) especifica que la condición para la bifurcación es que el señalizador de cero debe estar a 1 ($Z = 1$). El señalizador de cero es igual a 1 en la Figura 8.31; por tanto una nueva dirección de 16 bits es transferida al contador de programa (*PC*). La dirección de orden inferior (00000000) es transferida, seguida por la dirección de orden superior (00100001). La nueva dirección del contador de programa entonces hace que la MPU salte a una nueva posición del programa.

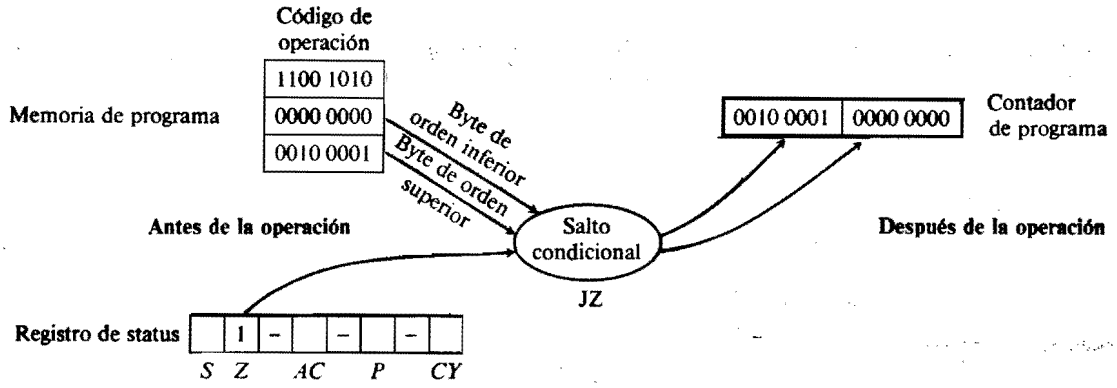


Figura 8.31. Instrucción salta si cero.

La instrucción CALL es una operación especial de bifurcación utilizada para saltar desde el programa principal a una subrutina. La instrucción CALL combina las operaciones de salto e introducción de datos en la pila. Como en el microprocesador genérico, la instrucción CALL del 8080/8085 introduce primero el contenido de 16 bits del contador de programa (*PC*) en la pila; después hace que la dirección de 16 bits de la subrutina sea transferida al contador de programa. El microprocesador entonces salta a la subrutina.

De forma distinta al microprocesador genérico, los procesadores 8080 y 8085 también tienen *instrucciones de llamada condicional.* La operación de llamada como se describió antes se ejecutará solamente si se cumple la condición especificada.

Como con el microprocesador genérico, las MPU 8080 y 8085 utilizan instrucciones de vuelta. Las instrucciones de vuelta indican al procesador la posición correcta de la memoria del programa principal una vez que finaliza una subrutina. De forma distinta al microprocesador genérico, las MPU 8080 y 8085 también tienen *instrucciones de vuelta condicional.* La operación de vuelta de subrutina se ejecutará solamente si se cumple la condición especificada.

Las *instrucciones de rearranque* son instrucciones de llamada de propósito especial, diseñadas principalmente para utilizarlas con interrupciones. Igual que CALL, la instrucción de rearranque introduce el contenido del contador de programa en la pila y salta a una de ocho *direcciones predeterminadas.* Esto es diferente porque una instrucción CALL especifica inmediatamente la dirección del salto. Un código de 3 bits (bits b_5, b_4 y b_3) en el código de op de la instrucción RST especifica la dirección del salto. Por ejemplo, una instrucción RST 2 tiene el código de op 11010111 (D7H). El código de dirección 010 (decimal 2) es multiplicado por 8, dando 16_{10}, ó 10000_2. Esta es la dirección del salto (0000 0000 0001 0000$_2$ en este ejemplo).

Considerar el problema ejemplo mostrado en la Figura 8.32 utilizando la instrucción RST 7. Observar que el puntero de pila contiene 2089H antes de la operación RST 7. La instrucción RST 7 introduce la cuenta del programa actual en la pila. Entonces la dirección predeterminada ($8 \times 7 = 56_{10} = 38H$) es transferida al contador de programa (*PC*). Esto hace que la MPU salte o rearranque en la nueva posición 0000 0000 0011 1000 (0038H).

La instrucción de rearranque es única, porque rara vez aparece en código fuente en los programas de aplicación. Con más frecuencia los dispositivos periféricos que buscan servicios de interrupción pasan esta instrucción de 1 byte al procesador. Los rearranques se denominan por tanto, a veces, *interrupciones software.*

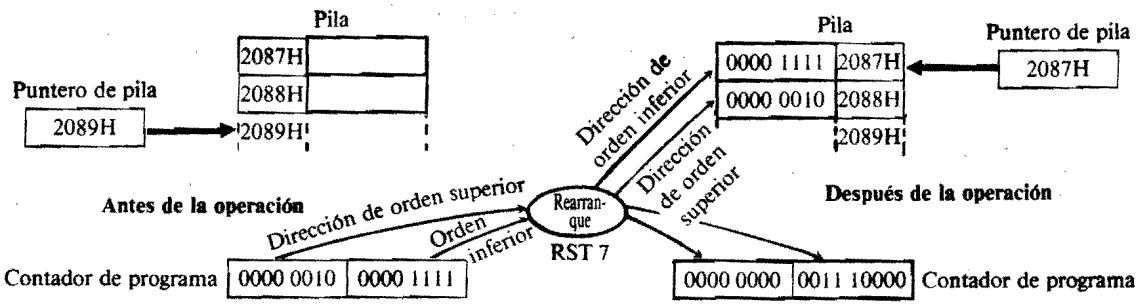


Figura 8.32. Instrucción de rearranque (RST 7).

PROBLEMAS RESUELTOS

8.58. La instrucción JMP utiliza una instrucción de _____ (número) byte y direccionamiento _____ (directo, inmediato). Durante la ejecución de la instrucción JMP, 2 bytes de la memoria de programa son transferidos al _____ _____ (contador de programa, puntero de pila).

Solución:

De acuerdo con la descripción de Intel, la instrucción JMP utiliza una instrucción de 3 bytes y direccionamiento inmediato. Durante la ejecución de la instrucción JMP, 2 bytes de la memoria del programa son transferidos al contador de programa.

8.59. Las instrucciones de llamada y _____ (reinicialización, vuelta) son utilizadas en pares para localizar y volver de las subrutinas.

Solución:

Las instrucciones de llamada y vuelta son utilizadas en pares para localizar y volver de las subrutinas.

8.60. La operación de _____ (reinicialización, vuelta) es una forma especial de instrucción de llamada utilizada principalmente con interrupciones.

Solución:

La operación de reinicialización es una forma especial de instrucción de llamada usada principalmente con interrupciones.

8.61. La instrucción JNC del 8080/8085 es una instrucción de salto _____ (condicional, incondicional) cuyo código de op es _____ en binario. El salto, si no hay operación de arrastre, se especifica con una instrucción de _____ (número) byte(s). La instrucción JNC utiliza el modo de direccionamiento _____ (directo, inmediato).

Solución:

De acuerdo con la descripción de Intel de «salto condicional», la instrucción JNC es una instrucción de salto condicional cuyo código de op es 11010010 en binario. El salto, si no hay arrastre, se especifica con una instrucción de 3 bytes. La instrucción JNC utiliza el modo de direccionamiento inmediato.

8.62. El nemotécnico para la instrucción de 1 byte del grupo de bifurcación del 8080/8085 que hace que la MPU salte a la posición de memoria 0000H es _____.

Solución:

El nemotécnico para la instrucción de 1 byte del grupo de bifurcación del 8080/8085 que hace que la MPU salte a la posición de memoria 0000H es RST 0.

8.63. El nemotécnico para la instrucción del grupo de bifurcación del 8080/8085 que hace que la MPU salte al contenido de la dirección del registro par *HL* es _____.

Solución:

El nemotécnico para la instrucción del grupo de bifurcación del 8080/8085 que hace que la MPU salte a la dirección contenida en el registro par *HL* es PCHL.

8.64. El nemotécnico del 8080/8085 para la operación de salto condicional mostrada en la Figura 8.33 es _____.

Solución:

El nemotécnico del 8080/8085 para la operación de salto condicional mostrada en la Figura 8.33 es JP, que significa saltar si más.

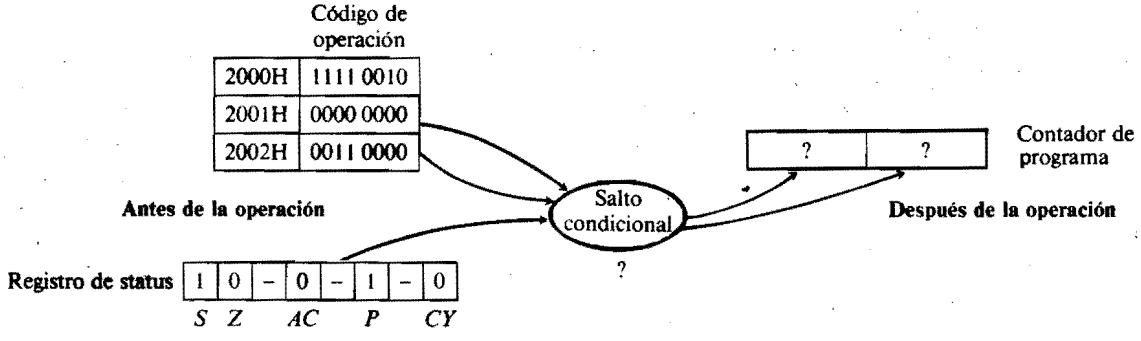


Figura 8.33.

8.65. Según el registro de status de la Figura 8.33, _____ (se cumplen, no se cumplen) las condiciones para que ocurra el salto condicional.

Solución:

El señalizador de signo es igual a 1 en la Figura 8.33, lo que significa que el número anterior del acumulador es negativo. No existe la condición positivo (más), y por tanto no se realiza el salto a una nueva dirección.

8.66. El contenido del contador de programa después de que se ha ejecutado la instrucción de salto condicional en la Figura 8.33 es _____ (16 bits).

Solución:

La condición más no se cumplía ($S = 1$); por tanto el contador de programa incrementa a la siguiente instrucción secuencial en 2003H, ó $0010\ 0000\ 0000\ 0011_2$.

8.67. El código de op para la instrucción de rearranque utilizada en la Figura 8.34 será _____ en binario.

Solución:

El código de op para la instrucción RST 4 utilizada en la Figura 8.34 será 11100111 en binario. (E7 en hexadecimal).

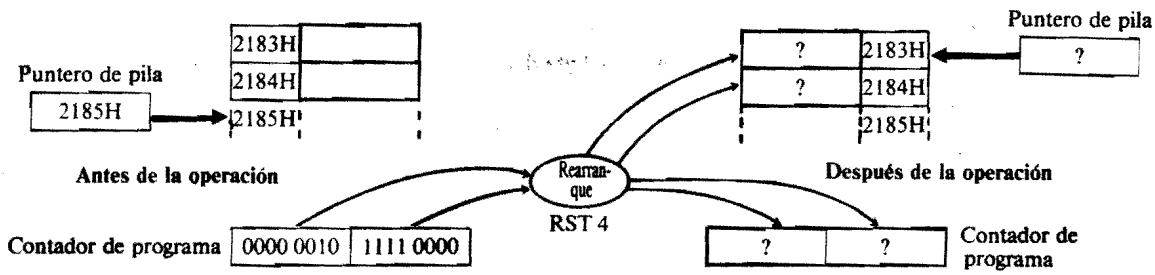


Figura 8.34.

8.68. Listar el contenido de las posiciones de la pila 2183H y 2184H después de la operación de reinicialización mostrada en la Figura 8.34.

Solución:

El contenido del contador de programa es transferido a la pila de la siguiente forma: Primero, la posición de la pila 2184H recibe el byte de orden inferior del contador de programa; la posición de la pila 2184H entonces contiene 00000010. Segundo, la posición de la pila 2183H recibe el byte de orden inferior del contador de programa; la posición de la pila 2183H entonces contiene 11110000.

8.69. Listar el contenido del contador de programa (*PC*) después de la operación de reinicialización mostrada en la Figura 8.34.

Solución:

La MPU multiplica el número de reinicialización (decimal 4 en este ejemplo) por 8, dando la dirección de 32_{10} (o 20H). El binario 0000 0000 0010 0000 (0020H) es la nueva dirección del contador de programa. La MPU entonces realiza el salto a la posición de memoria 0020H.

8.10. INSTRUCCIONES DE PILA, E/S Y CONTROL DE MAQUINA DEL 8080/8085

Un resumen de la pila, E/S e instrucciones de control máquina del microprocesador 8080/8085 se muestra en la Figura 8.35. Donde se incluyen instrucciones de introducir y sacar, entrar y salir, intercambiar, habilitación e inhabilitación de interrupciones, no operación y alto y lectura multi-

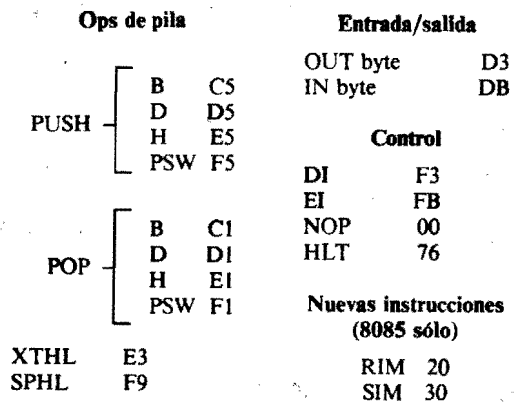


Figura 8.35. Resumen de las instrucciones de control de máquina, E/S, pila del 8080/8085. (Todos los nemotécnicos tienen derecho de copia © Intel Corporation, 1976.)

propósito e inicialización de máscaras de interrupción. Solamente los nemotécnicos y códigos de op de Intel están listados para cada instrucción en la Figura 8.35.

Como en las secciones anteriores, la *pila, E/S* y *grupo* de instrucciones *de control máquina* se detallarán utilizando el formato propio de Intel Corporation. Este grupo de instrucciones realiza entradas y salidas, manipula la pila y altera los señalizadores de control interno. Acudir a la Figura 8.18 para los significados de las abreviaturas utilizadas en las siguientes descripciones del *Manual del Usuario* de Intel.

PUSH rp (Introducir)
((SP) − 1) ← (rh)
((SP) − 2) ← (rl)
((SP) ← (SP) − 2
El contenido del registro de orden superior del registro par rp es transferido a la posición de memoria cuya dirección es una unidad menor que el contenido del registro SP. El contenido del registro de orden inferior del registro par rp es transferido a la posición de memoria cuya dirección es dos unidades menor que el contenido del registro SP. El contenido del registro SP es decrementado en 2. Nota: **El registro par rp = SP no puede ser especificado.**

1	1	R	P	0	1	0	1

Ciclos: 3
Estados: 12 (8085), 11 (8080)
Direccionamiento: indirecto de reg.
Señalizadores: ninguno

PUSH PSW (Introducir palabra de status del procesador)
((SP) − 1) ← (A)
$((SP) - 2)_0 \leftarrow (CY)$, $((SP) - 2)_1 \leftarrow X$
$((SP) - 2)_2 \leftarrow (P)$, $((SP) - 2)_3 \leftarrow X$
$((SP) - 2)_4 \leftarrow (AC)$, $((SP) - 2)_5 \leftarrow X$
$((SP) - 2)_6 \leftarrow (Z)$, $((SP) - 2)_7 \leftarrow (S)$
(SP) ← (SP) − 2 X: Indefinido.
El contenido del registro A es transferido a la posición de memoria cuya dirección es una unidad menor que el registro SP. El contenido de los señalizadores de con-

dición son ensamblados en una palabra de status de procesador y la palabra es transferida a la posición de memoria cuya dirección es dos unidades menor que el contenido del registro SP. El contenido del registro SP es decrementado en dos.

1	1	1	1	0	1	0	1

Ciclos: 3
Estados: 12 (8085), 11 (8080)
Direccionamiento: indirecto de reg.
Señalizadores: ninguno

PALABRA SEÑALIZADORA

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	X	AC	X	P	X	CY

X: indefinido

POP rp (Sacar)
(rl) ← ((SP))
(rh) ← ((SP) + 1)
(SP) ← (SP) + 2
El contenido de la posición de memoria, cuya dirección está especificada por el contenido del registro SP, es transferido al registro de orden inferior del registro par rp. El contenido de la posición de memoria, cuya dirección es una unidad mayor que el contenido del registro SP, es transferido al registro rp. El contenido del registro SP es incrementado en 2. Nota: **El registro par rp = SP no puede ser especificado.**

1	1	R	P	0	0	0	1

Ciclos: 3
Estados: 10
Direccionamiento: indirecto de reg.
Señalizadores: ninguno

POP PSW (Sacar palabra de status del procesador)
$(CY) \leftarrow ((SP))_0$
$(P) \leftarrow ((SP))_2$
$(AC) \leftarrow ((SP))_4$
$(Z) \leftarrow ((SP))_6$
$(S) \leftarrow ((SP))_7$
(A) ← ((SP) + 1)
(SP) ← (SP) + 2
El contenido de la posición de memoria cuya dirección está especificada por el contenido del registro SP es utilizado para restaurar los señalizadores de condición. El contenido de la posición de memoria cuya dirección es una unidad más que el contenido del registro SP es transferido al registro A. El contenido del registro SP es incrementado en 2.

1	1	1	1	0	0	0	1

Ciclos: 3
Estados: 10
Direccionamiento: indirecto de reg.
Señalizadores: Z,S,P,CY,AC

XTHL (Intercambia tope de la pila con H y L)
(L) ↔ ((SP))
(H) ↔ ((SP) + 1)
El contenido del registro L es intercambiado con el contenido de la posición de memoria cuya dirección está especificada por el contenido del registro SP. El contenido del registro H es intercambiado con el contenido de la posición de memoria cuya dirección es una unidad más que el contenido del registro SP.

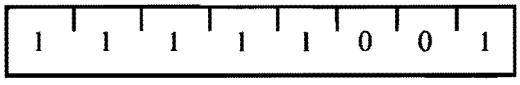
1	1	1	0	0	0	1	1

Ciclos: 5
Estados: 16 (8085), 18 (8080)
Direccionamiento: indirecto de reg.
Señalizadores: ninguno

SPHL (Transferir HL a SP)

(SP) ← (H) (L)

El contenido de los registros H y L (16 bits) es transferido al registro SP.

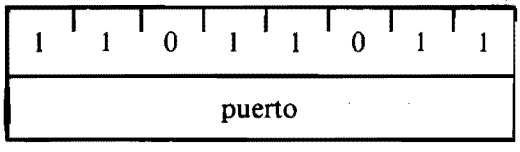


Ciclos: 1
Estados: 6 (8085), 5 (8080)
Direccionamiento: registro
Señalizadores: ninguno

IN puerto (Entrar)

(A) ← (dato)

El dato colocado en el bus de datos bidireccional de 8 bits, por el puerto especificado es transferido al registro A.

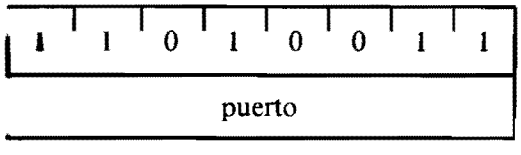


Ciclos: 3
Estados: 10
Direccionamiento: directo
Señalizadores: ninguno

OUT puerto (Salir)

(dato) ← (A)

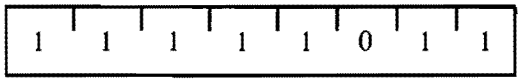
El contenido del registro A es colocado en el bus de datos bidireccional de ocho bits para que se transmita al puerto especificado.



Ciclos: 3
Estados: 10
Direccionamiento: directo
Señalizadores: ninguno

EI (Habilita interrupciones)

El sistema de interrupción es habilitado **después de la ejecución de la siguiente instrucción. Las interrupciones no son reconocidas durante la instrucción EI.**

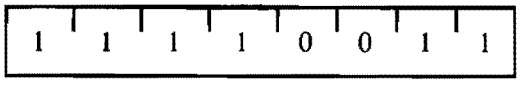


Ciclos: 1
Estados: 4
Señalizadores: ninguno

NOTA: Está prohibido colocar una instrucción EI en el bus en respuesta a $\overline{\text{INTA}}$ durante un ciclo INA (8085).

DI (Inhabilita interrupciones)

El sistema de interrupciones se habilita **inmediatamente después de la ejecución de la instrucción DI. Las interrupciones no son reconocidas durante la instrucción DI.**

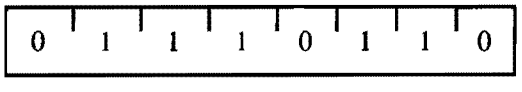


Ciclos: 1
Estados: 4
Señalizadores: ninguno

NOTA: Está prohibido colocar una instrucción DI en el bus en respuesta a $\overline{\text{INTA}}$ durante un ciclo INA (8085).

HLT (Alto)

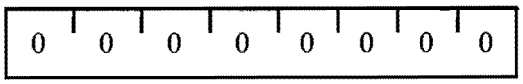
El procesador se detiene. Los registros y señalizadores no se afectan. (8080) Un segundo ALE es generado durante la ejecución de HLT para dar información del status del ciclo Halt. (8085)



Ciclos: 1 + (8085), 1 (8080)
Estados: 5 (8085), 7 (8080)
Señalizadores: ninguno

NOP (No operación)

No se realiza ninguna operación. Los registros y señalizadores no están afectados.



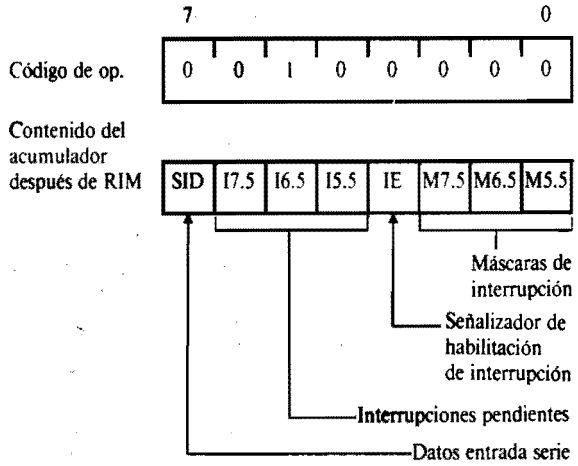
Ciclos: 1
Estados: 4
Señalizadores: ninguno

RIM (Lectura de máscaras de interrupción) (solamente 8085)

La instrucción RIM carga datos en el acumulador relativos a interrupciones y entradas serie. Este dato contiene la siguiente información:

- Status actual de la máscara de interrupción para las interrupciones hardware RST 5.5, 6.5 y 7.5 (1 = máscara inhabilitada).
- Status actual del señalizador de habilitación de interrupciones (1 = interrupciones habilitadas) excepto inmediatamente después de una interrupción TRAP. (Ver más abajo.)
- Interrupciones hardware pendientes (por ejemplo, señal recibida pero todavía no servida), en las líneas RST 5.5, 6.5 y 7.5.
- Datos de entrada serie.

Inmediatamente después de una interrupción TRAP, debe ser ejecutada la instrucción RIM como parte de la rutina de servicio si se necesita recuperar más tarde el status de interrupción actual. El bit 3 del acumulador está (solamente en este caso especial) cargado con el status del señalizador de habilitación de interrupción (IE) que existía antes de la interrupción RST 5.5, 6.5, 7.5 o INTR, el flip-flop del señalizador de interrupciones refleja el status actual de habilitación de interrupción. El bit 6 del acumulador (I7.5) es cargado con el status del flip-flop de RST 7.5, que está siempre en set (disparado por flanco) por una entrada en la línea de entrada RST 7.5, aun cuando esa interrupción haya sido previamente enmascarada. (Ver instrucción SIM.)



Ciclos: 1
Estados: 4
Señalizadores: ninguno

SIM (Pone a 1 las máscaras de las interrupciones) (solamente 8085)

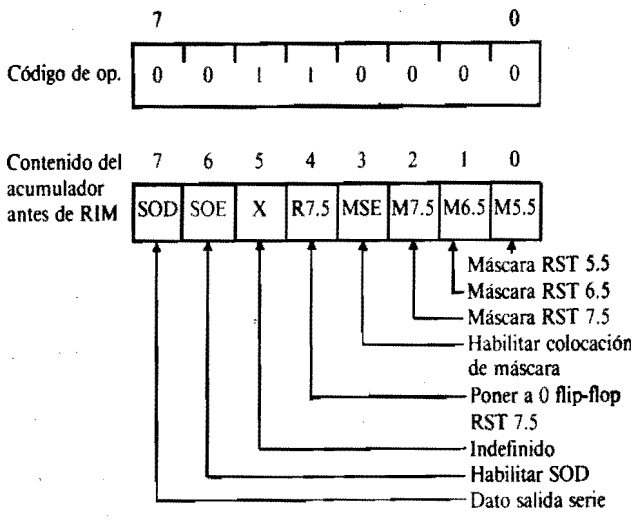
La ejecución de la instrucción SIM utiliza el contenido del acumulador (que debe ser cargado previamente) para realizar las siguientes funciones:

- Programar las máscaras de interrupciones para las interrupciones hardware RST 5.5, 6.5 y 7.5.
- Poner a 0 el cerrojo de entrada RST 7.5 disparado por flanco.
- Cargar el cerrojo de salida SOD.

Para programar las máscaras de interrupciones, primero poner a 1 el bit 3 del acumulador y poner a 1 cualquiera de los bits 0, 1 y 2, que inhabilitan las interrupciones RST 5.5, 6.5 y 7.5, respectivamente, Ejecutar entonces una instrucción SIM. Si el bit 3 del acumulador está a 0 cuando se ejecuta la

instrucción SIM, el registro de máscaras de interrupciones no cambiará. Si el bit 4 del acumulador está a 1 cuando se ejecuta la instrucción SIM, el cerrojo RST 7.5 está entonces en reinicialización. RST 7.5 se distingue por el hecho de que su cerrojo está siempre en set por un flanco de subida en la patilla de entrada de RST 7.5, aunque el salto a la rutina de servicio esté inhibido por enmascaramiento. Este cerrojo permanece en alta hasta que sea borrado por un REINICIALIZACION IN, por una instrucción SIM con el bit 4 del acumulador en alto, o por un conocimiento interno del procesador de una interrupción RST 7.5 posterior a la eliminación de la máscara (por una instrucción SIM). La señal REINICIALIZACION IN siempre pone a 1 los tres bits de máscara RST.

Si el bit 6 del acumulador está en el nivel 1 cuando se ejecuta la instrucción SIM, el estado del bit 7 del acumulador se carga en el cerrojo SOD y, por tanto, está disponible para conectarse a un dispositivo externo. El cerrojo SOD no es afectado por la instrucción SIM si el bit 6 está a 0. SOD está siempre en reinicialización por la señal REINICIALIZACION IN.



Ciclos: 1
Estados: 4
Señalizadores: ninguno

Este grupo contiene instrucciones para introducir o sacar datos de la pila. Como en el microprocesador genérico, la pila del 8080/8085 se localiza fuera del microprocesador y en la RAM. La parte superior de la pila está definida poniendo el puntero de pila del 8080/8085 en una dirección seleccionada por el programador. Todas las instrucciones PUSH y POP del 8080/8085 transfieren 2 bytes de datos a o desde la pila utilizando una sola instrucción. Las instrucciones PUSH *PSW* y POP *PSW* transfieren datos entre la *palabra de status del procesador* y la pila. La palabra de status del procesador se define cómo la combinación del registro de status de la MPU (señalizadores) y acumulador.

Las instrucciones IN y OUT del 8080/8085 funcionan de forma similar a las del microprocesador genérico. La operación IN introduce un byte de datos desde el puerto especificado por el segundo byte de la instrucción. De forma análoga, la operación OUT transfiere un byte de datos desde el acumulador al puerto de salida especificado en el segundo byte de la instrucción. Como en el microprocesador genérico, estas instrucciones son utilizadas con una técnica de E/S.

La operación XTHL es una instrucción de 1 byte para intercambiar el tope de la pila con el contenido del registro par *HL*. La operación SPHL es una instrucción de transferencia de datos. Transfiere el contenido del registro *HL* al puntero de pila del microprocesador. Esta instrucción puede utilizarse para inicializar el puntero de pila, definiendo así el tope de la pila de la RAM.

La instrucción NOP no realiza ninguna operación, como en el microprocesador genérico. La instrucción NOP es útil como un retardo de tiempo en un ciclo de temporización.

La instrucción HLT detiene el procesador. El contador de programa contiene la dirección de la siguiente instrucción secuencial. Los demás señalizadores y registros no cambian. Una vez en el

estado de alto, el microprocesador puede ser reinicializado solamente por un evento externo, típicamente una interrupción. La operación HLT se utiliza para detener la MPU cuando está esperando que un dispositivo periférico termine su tarea e interrumpa al procesador.

Las acciones del microprocesador pueden ser iniciadas de dos formas. Primero, la acción del microprocesador puede ser iniciada por una instrucción de la memoria del programa. Un segundo método de iniciación de la acción de la MPU es utilizar una interrupción externa. Las interrupciones hardware de la MPU 8080 u 8085 pueden ser controladas por dos instrucciones. Estas son las de habilitación de interrupciones (*EI*) e inhabilitación de interrupciones (*DI*). Cuando se ejecuta la intrucción EI, *se pondrá a 1 un flip-flop de habilitación de interrupción* en la CPU. Cuando el flip-flop de habilitación de interrupción está a 1 o habilitado, la MPU 8085 reconocerá y responderá a cualquiera de las cinco posibles interrupciones hardware.

La instrucción DI (inhabilitación de interrupciones) hará que el señalizador de habilitación de interrupciones (o flip-flop) se ponga a 0. Cuando el señalizador de habilitación de interrupciones está a 0, la MPU del 8085 ignorará todas las interrupciones excepto la interrupción *TRAP*. Esta interrupción no puede ser inhabilitada por ninguna instrucción de programa del 8085.

La instrucción EI se utiliza frecuentemente como parte de una secuencia de arranque. Cuando se conecta la alimentación, la MPU comienza a intentar ejecutar instrucciones desde algunas direcciones desconocidas. Habitualmente, se aplica una entrada de $\overline{REINICIALIZACION}$, forzando a que el contador de programa contenga 0000H. La entrada de $\overline{REINICIALIZACION}$ también hace que el señalizador de habilitación de interrupciones se ponga a 0, inhabilitando todas las interrupciones del 8085 excepto la *TRAP*. Una de las primeras instrucciones después de la reinicialización es EI para habilitar las interrupciones.

Las interrupciones, a veces, se referencian como enmascarables o no enmascarables. Una *interrupción no enmascarable* es la que no puede inhabilitar el sistema. Una *interrupción enmascarable* es la que puede inhabilitar el sistema. En el microprocesador 8085, la interrupción *TRAP* se considera no enmascarable mientras que las restantes son enmascarables.

Las instrucciones EI y DI habilitan e inhabilitan todas las interrupciones enmascarables como un grupo. Las interrupciones *RST 7.5, RST 6.5* y *RST 5.5* pueden ser habilitadas e inhabilitadas individualmente. El mecanismo real para habilitar e inhabilitar selectivamente las interrupciones se denomina *máscara de interrupción.* Si una interrupción es *enmascarada,* no es reconocida por el procesador (esto es análogo a un ladrón de bancos enmascarado y por tanto no reconocible). Sin embargo, si se elimina la máscara, la interrupción será reconocida por el procesador.

Considerar el ejemplo *simplificado* de interrupción mostrado en la Figura 8.36. Seguir las acciones de la MPU por los números inscritos en los círculos de la Figura 8.36.

1. La entrada *RST 5.5* (patilla 9) a la pastilla 8085 se activa con una señal en ALTA.
2. Para simplificar el ejemplo, se suponen tres cosas. Estas son:
   (*a*) No se activa la interrupción de más alta prioridad *TRAP.*
   (*b*) No se activa o enmascara la interrupción de alta prioridad *RST 7.5.*
   (*c*) No se activa o enmascara la interrupción de alta prioridad *RST 6.5.*
3. Se comprueba el bit de la máscara de interrupción (B_0) del *registro de status de interrupción.* Está en reinicialización (a 0), o no enmascarado, lo cual satisface la primera condición para que la interrupción *RST 5.5* sea reconocida por la MPU.
4. Se comprueba el señalizador de habilitación de interrupción (B_3) del registro de status de interrupción. Está a 1, lo que significa que todas las interrupciones están habilitadas.
5. La interrupción *RST 5.5* es reconocida por la MPU. El procesador entonces termina su instrucción actual y almacena el contenido del contador de programa en la pila.

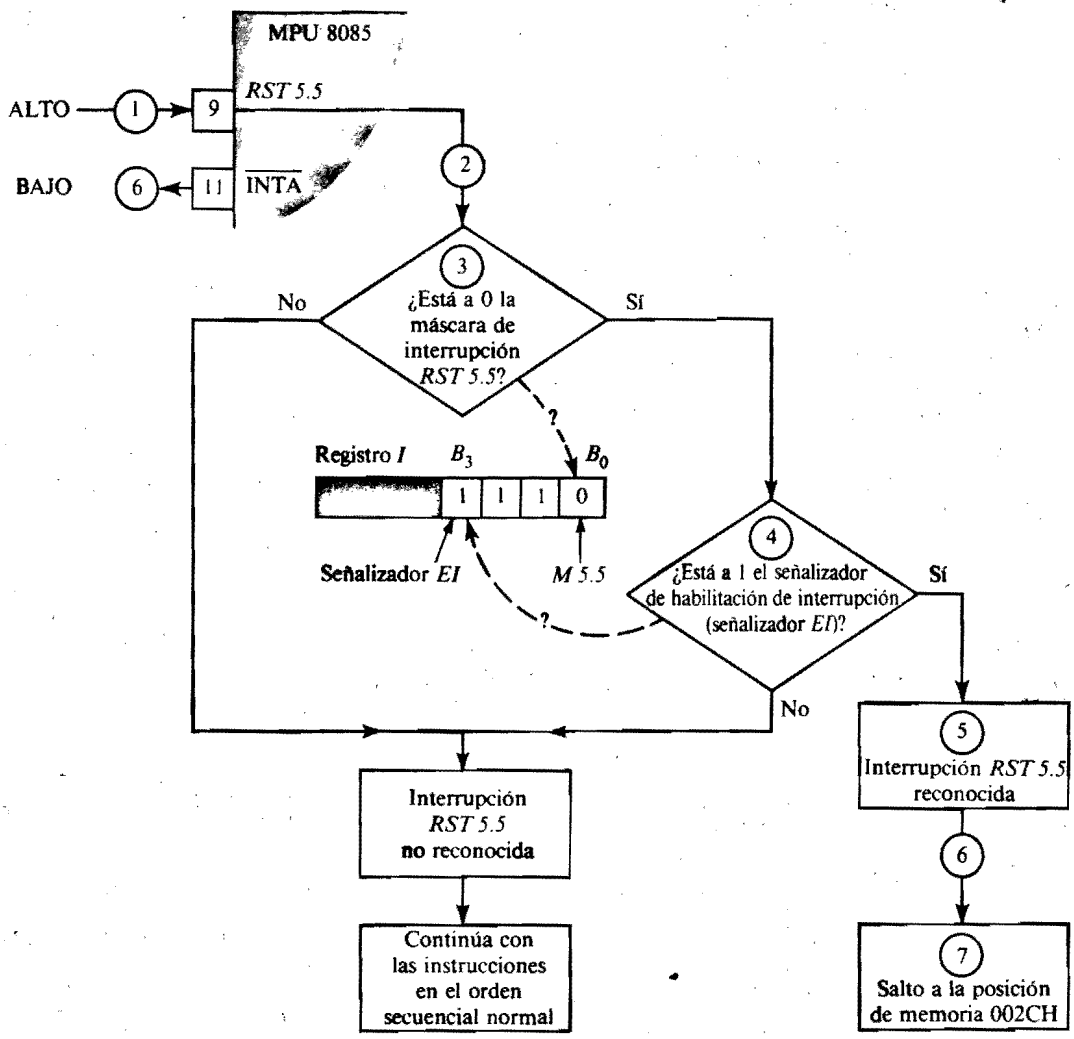


Figura 8.36. Pasos en el tratamiento de una interrupción no enmascarada *RST 5.5* de la MPU 8085.

6. La pastilla microprocesadora 8085 facilita una señal de conocimiento de interrupción ($\overline{INTA}$) a los periféricos.
7. La MPU entonces bifurca a la posición de memoria 002CH. El cálculo de la posición de memoria de reinicialización es $5.5 \times 8 = 44_{10} = 2C_{16}$.

La MPU examinaba la condición de los bits 2 con el registro de status de interrupción de la Figura 8.36. Primero durante el paso 3, se comprobó el bit de máscara 5.5 (*M* 5.5). El bit *M* 5.5

estaba a 0, o no enmascarado, lo que significaba que la MPU podía reconocer esta interrupción. Segundo, durante el paso 4, se comprobó el señalizador de habilitación de interrupción (señalizador *IE*). El señalizador *IE* de la Figura 8.36 estaba a 1 (habilitado), lo que significa que la MPU podía reconocer esta interrupción.

Algunos de los bits del registro de status de interrupción pueden ser cambiados utilizando la instrucción SIM única del 8085. La instrucción SIM (inicialización de las máscaras de interrupción) colocará el contenido del acumulador en el registro *I*. Anteriormente se aprendió que la instrucción SIM también se utiliza para sacar un bit de datos al cerrojo *SOD* (salida de datos serie) de la MPU 8085 (ver Fig. 8.7*b*).

El registro de máscaras de interrupción como lo utiliza la instrucción SIM está esbozado en la Figura 8.37. Los bits de las máscaras de *RST 5.5, RST 6.5* y *RST 7.5* pueden ser puestos a 1 (enmascarados) o a 0 (no enmascarados) si el bit 3 (habilitación de la máscara) está habilitado con un 1 lógico. Un 1 lógico en la posición del bit 4 pondrá a 0 el flip-flop de *RST 7.5*. El bit 5 de la Figura 8.37 no está definido. Los bits 6 y 7 del registro de máscaras de interrupción de la Figura 8.37 están asociados a la salida serie del procesador (la patilla *SOD* de la MPU 8085). Habilitar el bit 6 con 1 permite una transferencia de datos del bit 7 del registro *I* al cerrojo *SOD* en la patilla 4 de la pastilla 8085.

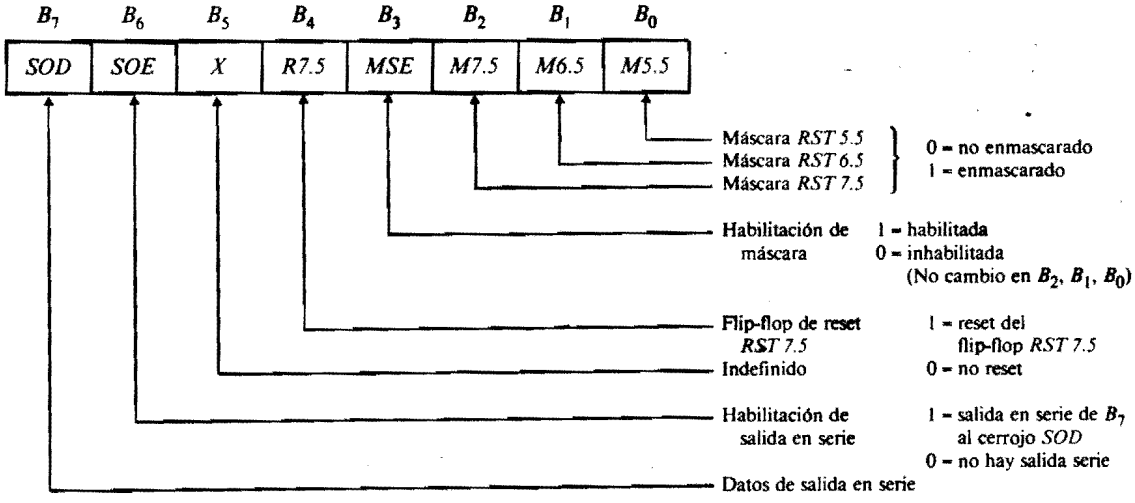


Figura 8.37. Organización del registro de interrupción como lo utiliza la instrucción SIM del 8085.

El registro de status de interrupción como lo lee la instrucción RIM está esbozado en la Figura 8.38. La instrucción RIM transfiere el contenido de los cerrojos internos de la Figura 8.38 al acumulador. La instrucción RIM se utilizó anteriormente en la Figura 8.7*a* para entrar datos serie desde la patilla *SID* de la MPU 8085 al bit 7 del acumulador. El status de otros diversos cerrojos también es visualizado en el acumulador junto con la entrada de datos serie.

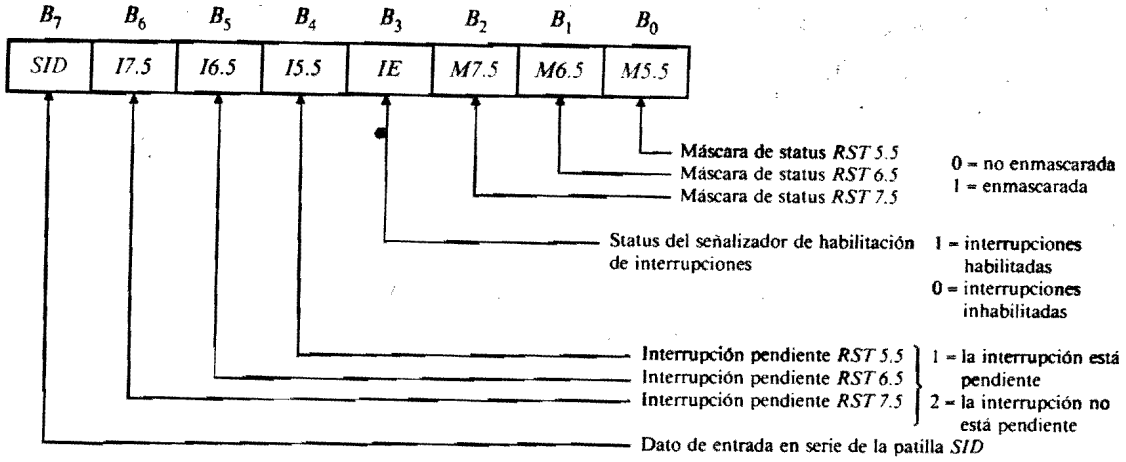


Figura 8.38. Organización del registro de interrupción cuando lee por la instrucción RIM del 8085.

PROBLEMAS RESUELTOS

8.70. Listar las instrucciones de la Figura 8.35 asociadas a la pila.

Solución:

Todas las instrucciones PUSH y POP listadas en la Figura 8.35 utilizan la pila. Las instrucciones XTHL y SPHL también están asociadas a la pila ya que cambian los contenidos del puntero de pila.

8.71. Listar las instrucciones de la Figura 8.35 que son utilizadas solamente por el microprocesador 8085 y no por la MPU más antigua 8080.

Solución:

Las operaciones RIM y SIM son solamente nuevas instrucciones utilizadas por la MPU 8085 que no fueron utilizadas por el procesador más antiguo 8080.

8.72. El contenido del registro par *HL* después de la operación XTHL de la Figura 8.39 es _____ (hex).

Solución:

Según la descripción de Intel de la instrucción «intercambiar tope de pila con *H* y *L*», el contenido del registro par *HL* después de la operación XTHL de la Figura 8.39 es 2000H.

8.73. ¿Cuál es el contenido de las posiciones de la pila 2080H y 2081H después de la operación XTHL de la Figura 8.39?

Solución:

Los contenidos de la pila después de la operación XTHL de la Figura 8.39 son 21H para la posición de la pila 2080H y 20H para la posición de la pila 2081H.

estaba a 0, o no enmascarado, lo que significaba que la MPU podía reconocer esta interrupción. Segundo, durante el paso 4, se comprobó el señalizador de habilitación de interrupción (señalizador *IE*). El señalizador *IE* de la Figura 8.36 estaba a 1 (habilitado), lo que significa que la MPU podía reconocer esta interrupción.

Algunos de los bits del registro de status de interrupción pueden ser cambiados utilizando la instrucción SIM única del 8085. La instrucción SIM (inicialización de las máscaras de interrupción) colocará el contenido del acumulador en el registro *I*. Anteriormente se aprendió que la instrucción SIM también se utiliza para sacar un bit de datos al cerrojo *SOD* (salida de datos serie) de la MPU 8085 (ver Fig. 8.7*b*).

El registro de máscaras de interrupción como lo utiliza la instrucción SIM está esbozado en la Figura 8.37. Los bits de las máscaras de *RST 5.5*, *RST 6.5* y *RST 7.5* pueden ser puestos a 1 (enmascarados) o a 0 (no enmascarados) si el bit 3 (habilitación de la máscara) está habilitado con un 1 lógico. Un 1 lógico en la posición del bit 4 pondrá a 0 el flip-flop de *RST 7.5*. El bit 5 de la Figura 8.37 no está definido. Los bits 6 y 7 del registro de máscaras de interrupción de la Figura 8.37 están asociados a la salida serie del procesador (la patilla *SOD* de la MPU 8085). Habilitar el bit 6 con 1 permite una transferencia de datos del bit 7 del registro *I* al cerrojo *SOD* en la patilla 4 de la pastilla 8085.

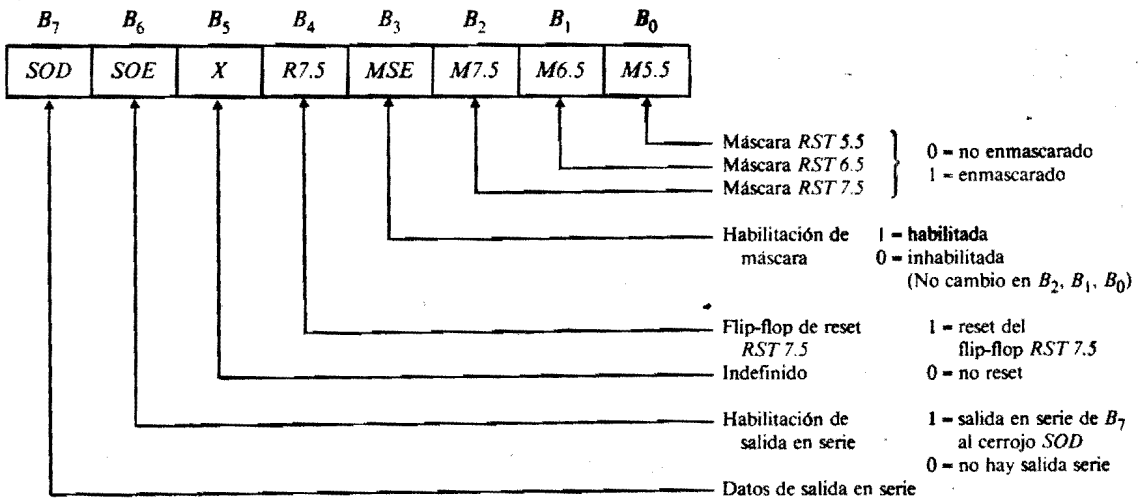


Figura 8.37. Organización del registro de interrupción como lo utiliza la instrucción SIM del 8085.

El registro de status de interrupción como lo lee la instrucción RIM está esbozado en la Figura 8.38. La instrucción RIM transfiere el contenido de los cerrojos internos de la Figura 8.38 al acumulador. La instrucción RIM se utilizó anteriormente en la Figura 8.7*a* para entrar datos serie desde la patilla *SID* de la MPU 8085 al bit 7 del acumulador. El status de otros diversos cerrojos también es visualizado en el acumulador junto con la entrada de datos serie.

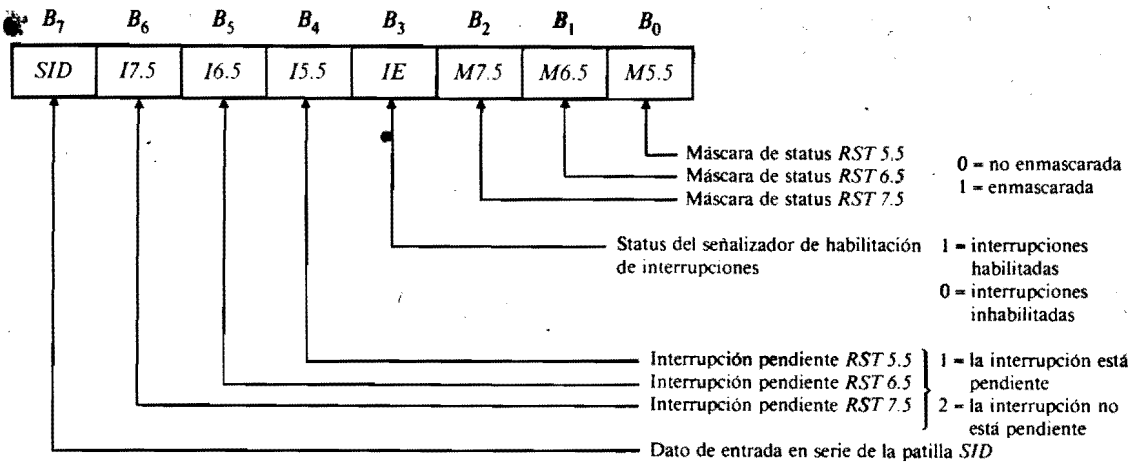


Figura 8.38. Organización del registro de interrupción cuando lee por la instrucción RIM del 8085.

PROBLEMAS RESUELTOS

8.70. Listar las instrucciones de la Figura 8.35 asociadas a la pila.

Solución:

Todas las instrucciones PUSH y POP listadas en la Figura 8.35 utilizan la pila. Las instrucciones XTHL y SPHL también están asociadas a la pila ya que cambian los contenidos del puntero de pila.

8.71. Listar las instrucciones de la Figura 8.35 que son utilizadas solamente por el microprocesador 8085 y no por la MPU más antigua 8080.

Solución:

Las operaciones RIM y SIM son solamente nuevas instrucciones utilizadas por la MPU 8085 que no fueron utilizadas por el procesador más antiguo 8080.

8.72. El contenido del registro par *HL* después de la operación XTHL de la Figura 8.39 es _____ (hex).

Solución:

Según la descripción de Intel de la instrucción «intercambiar tope de pila con *H* y *L*», el contenido del registro par *HL* después de la operación XTHL de la Figura 8.39 es 2000H.

8.73. ¿Cuál es el contenido de las posiciones de la pila 2080H y 2081H después de la operación XTHL de la Figura 8.39?

Solución:

Los contenidos de la pila después de la operación XTHL de la Figura 8.39 son 21H para la posición de la pila 2080H y 20H para la posición de la pila 2081H.

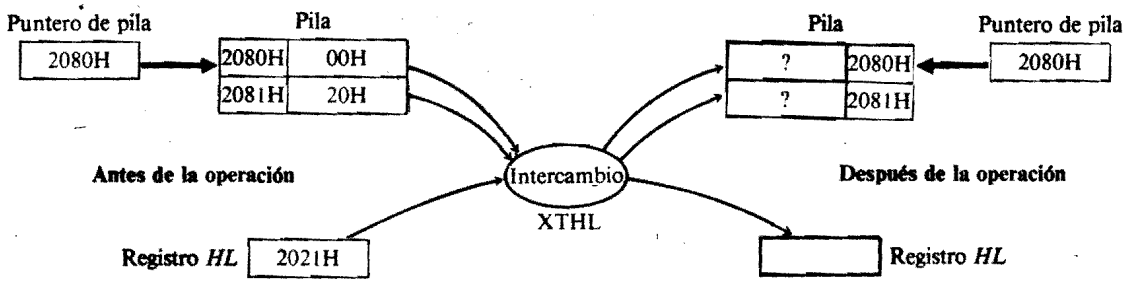


Figura 8.39.

8.74. La instrucción EI afectará ¿a qué bit del registro de status de interrupción mostrado en la Figura 8.38?

Solución:

La instrucción EI (habilita interrupciones) pondrá B_3 a 1 (el señalizador de habilitación de interrupciones).

8.75. La instrucción HLT tiene un código de op de _____ (hex). Afecta _____ (a todos, a ningún) (los) señalizador(es). Una vez en el estado alto, el procesador puede rearrancar sólo por un evento tal como _____ (interrupción, instrucción NOP).

Solución:

La instrucción HLT tiene un código de op de 76H. No afecta a los señalizadores. Una vez que está en el estado alto, el procesador puede rearrancar sólo por un acontecimiento externo como por ejemplo una interrupción.

8.76. La instrucción OUT utiliza el modo de direccionamiento _____ (directo, de registro). La dirección del puerto de salida se encuentra en la memoria de _____ (datos, programa).

Solución:

La instrucción OUT utiliza el modo de direccionamiento directo. La dirección del puerto de salida se encuentra en la memoria de programa.

8.77. La instrucción RIM puede ser utilizada para _____ (recibir, enviar) datos serie vía la patilla _____ (*SID, SOD*) del microprocesador 8085. Este simple bit de datos finaliza en el bit _____ (número) del acumulador.

Solución:

Según la descripción Intel de la operación «leer máscaras de interrupción», la instrucción RIM puede ser utilizada para recibir datos serie vía la patilla *SID* del microprocesador 8085. Este bit de datos finaliza en el bit 7 del acumulador.

8.78. Si una entrada tal como la interrupción *RST 6.5* está enmascarada, está _____ (inhabilitada, habilitada).

Solución:

Si una entrada tal como la interrupción *RST 6.5* está enmascarada, está inhabilitada.

8.79. Acudir a la Figura 8.40. Después de la operación SIM, listar el contenido de la posición de cada bit del registro *I*.

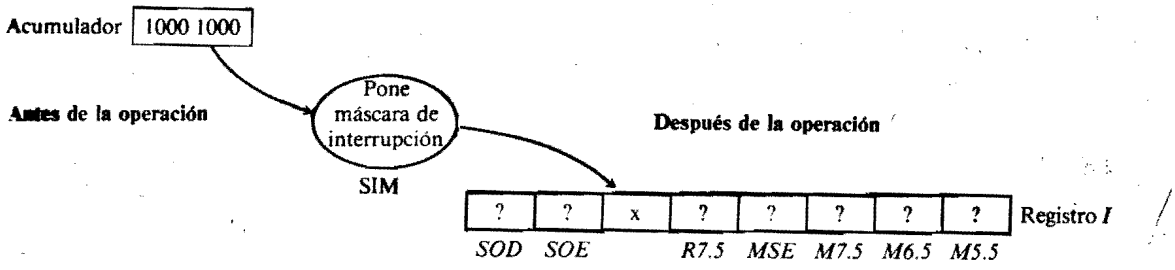


Figura 8.40.

Solución:

El registro *I* de la Figura 8.40 contendrá lo siguiente después de la operación SIM:
SOD (datos salida serie) = 1
SOE (habilitación salida serie) = 0
R 7.5 (flip-flop *RST 7.5* en reinicialización) = 0
MSE (habilitación de máscara) = 1
M 7.5 (máscara *RST 7.5*) = 0
M 6.5 (máscara de *RST 6.5*) = 0
M 5.5 (máscara *RST 5.5*) = 0

8.80. Acudir a la Figura 8.40. ¿Estará la salida de datos serie encerrada en el cerrojo *SOD* de la MPU del 8085?

Solución:

El bit de habilitación de salida serie del registro *I* de la Figura 8.40 está inhabilitado. Por tanto los datos de la salida serie no serán transferidos y encerrados en el cerrojo *SOD* de la MPU 8085.

8.81. Listar las interrupciones que quedarán no enmascaradas después de la instrucción SIM de la Figura 8.40.

Solución:

Las interrupciones *RST 7.5*, *RST 6.5* y *RST 5.5* quedarán no enmascaradas después de la instrucción SIM de la Figura 8.40.

PROBLEMAS SUPLEMENTARIOS

8.82. Intel Corporation introdujo el primer microprocesador, el 4004 de 4 bits, en _____ (año).
Res. 1971.

8.83. Acudir a la Figura 8.2. Las patillas 12 a 19 del 8085 tienen el doble propósito de líneas del bus de _____ / _____ .
Res. direcciones/datos.

8.84. Acudir a la Figura 8.2. La entrada *READY* a la MPU 8085 está más asociada con el estado de _____ (alto, espera) del procesador.

Res. espera.

8.85. Acudir a la Figura 8.2. ¿Qué dos patillas en la MPU 8085 (una de entrada y otra de salida) están más relacionadas con un acceso directo a memoria?

Res. entrada *HOLD*, salida *HLDA* (conocimiento de mantenimiento).

8.86. Acudir a la Figura 8.3. Las señales de control del 8085 son $IO/\overline{M} = 0$, $S_1 = 1$ y $S_2 = 0$. Esto significa que el microprocesador está realizando ¿qué tipo de ciclo máquina?

Res. lectura de memoria (ver Fig. 8.3).

8.87. Este registro de 16 bits siempre contiene una dirección y señala a la siguiente instrucción que va a ser ejecutada.

Res. contador de programa (*PC*).

8.88. El señalizador de signo en reinicialización (0) después de una operación aritmética en el 8085 significa que el resultado del acumulador es _____ (negativo, positivo).

Res. positivo.

8.89. El señalizador de cero en reinicialización (0) después de una operación aritmética en el microprocesador 8085 significa que el resultado del acumulador es _____ (no cero, cero).

Res. no cero.

8.90. La entrada *TRAP* del 8085 es una interrupción _____ (enmascarable, no enmascarable) y por tanto no puede ser inhabilitada.

Res. no enmascarable.

8.91. Acudir a la Figura 8.6. Cuando se activa la interrupción hardware *RST 6.5* del 8085, la CPU guarda el contenido del _____ _____ en la pila y bifurca a la dirección de memoria _____ (hex).

Res. contador de programa (*PC*), 34H.

8.92. Los acumuladores secundarios (*BC*, *DE* y *HL*) del 8085 pueden utilizarse como acumuladores o como _____ _____.

Res. contadores de datos (punteros de dirección).

8.93. El microprocesador 8085 utiliza _____ (número) señalizadores que contienen información de status.

Res. cinco.

8.94. La instrucción MOV *A,L* del 8085 (transfiere *L* a *A*) transfiere el contenido de un registro a otro. Por tanto utiliza el modo de direccionamiento de _____.

Res. registro.

8.95. La instrucción MVI *L* del 8085 transfiere datos del segundo byte de la instrucción al registro _____ (letra). Utiliza el modo de direccionamiento _____.

Res. *L*, inmediato.

8.96. Cada una de las instrucciones de direccionamiento indirecto de registro del 8085 utiliza _____ (1, 3) byte(s) de la memoria de programa.

Res. 1.

8.97. Cada una de las instrucciones de direccionamiento directo del 8085 utiliza _____ (1, 2 ó 3) bytes de la memoria de programa.

Res. 2 ó 3.

8.98. Intel clasifica la operación del 8080/8085 «almacenar *A* en memoria» como una instrucción de _____ (bifurcación, transferencia de datos).

Res. transferencia de datos.

8.99. Intel clasifica la operación de «desplazamiento circular» del 8080/8085 como una instrucción _____ (de transferencia de datos, lógica).

Res. lógica.

8.100. Intel clasifica una operación de «llamada» del 8080/8085 como una instrucción _____ (de bifurcación, lógica).

Res. bifurcación.

8.101. Acudir a la Figura 8.16. Algunos nemotécnicos del 8080/8085 contienen una I adicional como ADI *v*, ANI *v*, y MVI *A,v*. Cada vez que aparece la I adicional en estas instrucciones, se especifica el modo de direccionamiento _____.

Res. inmediato.

8.102. Acudir a la Figura 8.16. Listar los nemotécnicos para dos instrucciones utilizadas por el 8085 pero no por el microprocesador 8080.

Res. RIM, SIM.

8.103. En la instrucción MOV *D,E* del 8080/8085, el registro _____ (letra) es la fuente de datos mientras que el registro _____ (letra) es el destino.

Res. *E, D.*

8.104. En la instrucción MVI *A* del 8080/8085, la fuente del dato es _____.

Res. el segundo byte de la instrucción encontrada en la memoria de programa.

8.105. La instrucción MOV *B,M* del 8080/8085 utiliza direccionamiento _____.

Res. indirecto de registro.

8.106. La instrucción STA del 8080/8085 utiliza direccionamiento _____.

Res. directo.

8.107. La instrucción MOV *A,L* del 8080/8085 utiliza direccionamiento _____.

Res. de registro.

8.108. La instrucción ADD *M* del 8080/8085 suma el contenido de la posición de memoria señalada por el registro par _____ (dos letras) al contenido del registro _____ (letra). La suma se deposita en el registro _____ (letra).

Res. *HL, A, A.*

8.109. El código de op para la instrucción INX *H* mostrado en la Figura 8.41 es _____ (hex).
Res. 23H.

8.110. La instrucción INX *H* de la Figura 8.41 utiliza direccionamiento _____.
Res. de registro.

8.111. Listar el contenido de los registros *H* y *L* después de la operación de incrementación mostrada en la Figura 8.41.
Res. registro *H* = 00000001, registro *L* = 00000000.

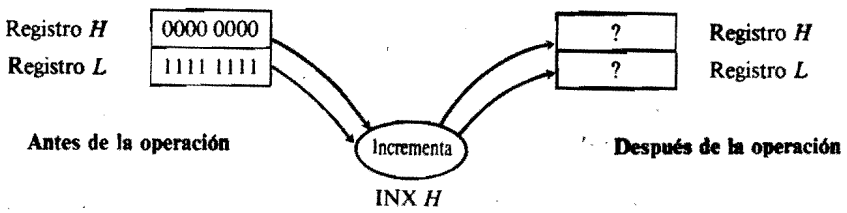


Figura 8.41.

8.112. El símbolo ∨ es usado por Intel para describir la operación lógica _____ (AND, OR).
Res. OR.

8.113. Las instrucciones CALL están más relacionadas con las subrutinas, mientras que las instrucciones de reinicialización están más próximas a las _____.
Res. interrupciones.

8.114. La instrucción JPE del 8080/8085 es una operación de salto _____ (condicional, incondicional).
Res. condicional.

8.115. La instrucción RNZ del 8080/8085 comprueba el status del señalizador de _____ antes de ejecutar la operación de vuelta.
Res. cero (*Z*).

8.116. La instrucción RP del 8080/8085, mostrada en la Figura 8.42, es una instrucción de _____ (número) byte(s) con un código de op _____ (hex).
Res. 1, F0.

8.117. El contador de programa del 8080/8085 contendrá _____ (hex) después de la operación «vuelta de subrutina si más» ilustrada en la Figura 8.42.
Res. 2004H.

8.118. El puntero de pila del 8080/8085 contendrá _____ (hex) después de la operación RP mostrada en la Figura 8.42.
Res. 2082H.

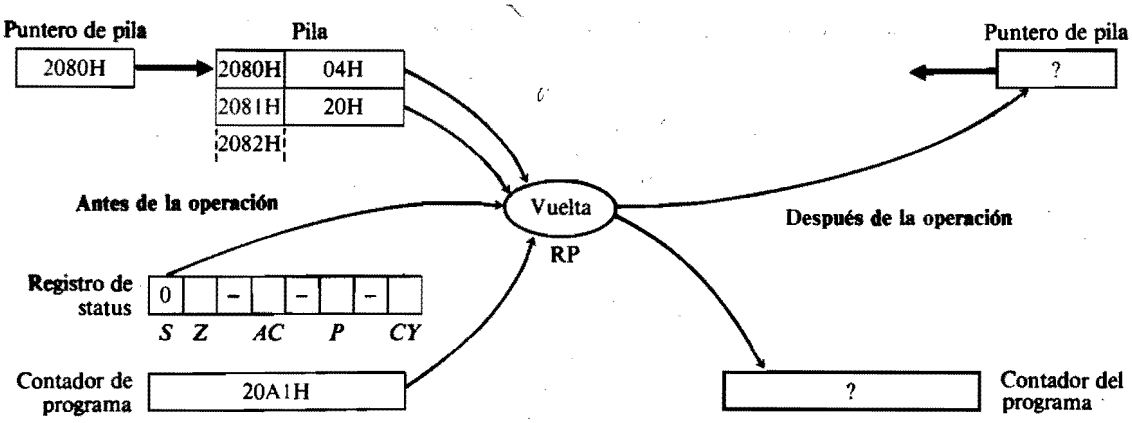


Figura 8.42.

8.119. Las operaciones PUSH y POP del 8080/8085 están especificadas con instrucciones de _____ (1, 3) bytes y utilizan el direccionamiento _____ .

Res. 1, indirecto de registro.

8.120. Las operaciones IN y OUT del 8080/8085 están especificadas con instrucciones de _____ (1, 2) bytes y utilizan direccionamiento _____ .

Res. 2, directo.

8.121. Al ejecutar la instrucción DI del 8085 ¿qué interrupciones se inhabilitarán?

Res. *INTR, RST 5.5, RST 6.5* y *RST 7.5.*

8.122. ¿Qué tres funciones pueden ser realizadas por la instrucción SIM del 8085?

Res. programar la máscara de interrupciones para las interrupciones hardware *RST 5.5, 6.5* y *7.5;* poner a 0 (reinicialización) el cerrojo de la interrupción disparada por flanco *RST 7.5* y cargar el cerrojo de salida *SOD.*

Capítulo 9

PROGRAMACION DEL MICROPROCESADOR 8080/8085

9.1. INTRODUCCION

Para el programador, la microcomputadora basada en el 8080/8085 consta del repertorio de instrucciones, memoria, registro de propósito general, contador de programa, puertos de entrada/salida, señalizadores, puntero de pila y pila. Todo ésto, excepto la memoria y entradas y salidas paralelo, forman parte del microprocesador.

Por conveniencia, el resumen del repertorio de instrucciones del microprocesador 8080/8085 se reimprime en la Figura 9.1. El repertorio de instrucciones se detalló en el Capítulo 8.

El modelo de programación del 8080/8085 se reproduce en la Figura 9.2*a*. Lista los registros que conciernen al programador. El registro de status se muestra con mayor detalle en la Figura 9.2*b*. Una breve descripción de cada uno de los cinco señalizadores del registro de status también se da en la Figura 9.2*b*.

Los programas fuente se escribirán en lenguaje ensamblador. El formato utilizado por Intel divide cada línea del lenguaje ensamblador en los siguientes campos:

Etiqueta *Código de op* *Operando* *Comentarios*

Estos campos son similares a los utilizados en la programación en lenguaje ensamblador del Capítulo 6. La entrada *del campo de etiqueta* es opcional y representa el «nombre» de la línea. Generalmente las líneas que son el destino de saltos o bifurcaciones tendrán entradas en el campo de etiqueta. El *campo de código de op* contiene el nemotécnico del código de operación de la instrucción del 8080/8085 que se va a ejecutar. El *campo de operando* da el dato con el que va a operar el código de operación especificado. El campo de operando a veces se denomina *argumento*. El *campo de comentarios* puede contener información útil para explicar la función de la instrucción. Los comentarios son opcionales, pero su uso es extremadamente importante para documentar el programa.

Una línea de programa en lenguaje ensamblador para el 8080/8085 podría ser:

```
DATA    MOV    A,M    ; Entrar dato al acumulador
```

Cada uno de los cuatro campos tiene una entrada en este ejemplo. Se explican como sigue:

1. Campo de etiqueta: Contiene la entrada DATA. Este es el «nombre» de esta línea. Probablemente es el destino de una instrucción de bifurcación posterior (o anterior).

	Nemotécnico	Código de op (hex)	Descripción
A	ADD *A*	87	Suma *A* a *A* (doble *A*)
	ADD *B*	80	Suma *B* a *A*
	ADD *C*	81	Suma *C* a *A*
	ADD *D*	82	Suma *D* a *A*
	ADD *E*	83	Suma *E* a *A*
	ADD *H*	84	Suma *H* a *A*
	ADD *L*	85	Suma *L* a *A*
	ADD *M*	86	Suma LOC memoria (*H* & *L*) a *A*
	ADI *v*	C6	Suma dato inmediato *v* a *A*
	ADC *A*	8F	Suma *A* a *A* con arrastre (doble *A* con arrastre)
	ADC *B*	88	Suma *B* a *A* con arrastre
	ADC *C*	89	Suma *C* a *A* con arrastre
	ADC *D*	8A	Suma *D* a *A* con arrastre
	ADC *E*	8B	Suma *E* a *A* con arrastre
	ADC *H*	8C	Suma *H* a *A* con arrastre
	ADC *L*	8D	Suma *L* a *A* con arrastre
	ADC *M*	8E	Suma LOC memoria (*H* & *L*) a *A* con arrastre
	ACI *v*	CE	Suma dato inmediato *v* a *A* con arrastre
	ANA *A*	A7	Examina *A* y borra arrastre
	ANA *B*	A0	AND *B* con *A*
	ANA *C*	A1	AND *C* con *A*
	ANA *D*	A2	AND *D* con *A*
	ANA *E*	A3	AND *E* con *A*
	ANA *H*	A4	AND *H* con *A*
	ANA *L*	A5	AND *L* con *A*
	ANA *M*	A6	AND LOC memoria (*H* & *L*) con *A*
	ANI *v*	E6	AND dato inmediato *v* con *A*
C	CALL *aa*	CD	Call subrutina en dirección *aa*
	CZ *aa*	CC	Si cero, CALL en dirección *aa*
	CNZ *aa*	C4	Si no cero, CALL en dirección *aa*
	CP *aa*	F4	Si más, CALL en dirección *aa*
	CM *aa*	FC	Si menos, CALL en dirección *aa*
	CC *aa*	DD	Si arrastre, CALL en dirección *aa*
	CNC *aa*	D4	Si no arrastre, CALL en dirección *aa*
	CPE *aa*	EC	Si paridad par, CALL en dirección *aa*
	CPO *aa*	E4	Si paridad impar, CALL en dirección *aa*
	CMA	2F	Complementa *A*
	CMC	3F	Complementa arrastre
	CMP *A*	BF	Pone a 1 el señalizador de cero
	CMP *B*	B8	Compara *A* con *B*
	CMP *C*	B9	Compara *A* con *C*
	CMP *D*	BA	Compara *A* con *D*
	CMP *E*	BB	Compara *A* con *E*
	CMP *H*	BC	Compara *A* con *H*
	CMP *L*	BD	Compara *A* con *L*
	CMP *M*	BE	Compara *A* con LOC memoria (*H* & *L*)
	CPI *v*	FE	Compara *A* con dato inmediato *v*
D	DAA	27	Ajuste decimal *A*
	DAD *B*	09	Suma *B* & *C* a *H* & *L*
	DAD *D*	19	Suma *D* & *E* a *H* & *L*
	DAD *H*	29	Suma *H* & *L* a *H* & *L* (doble *H* & *L*)
	DAD *SP*	39	Suma *SP* a *H* & *L*

Figura 9.1. Resumen del repertorio de instrucciones para el microprocesador 8080/8085. (Todos los nemotécnicos tienen derecho de copia © de Intel Corporation, 1976.)

	Nemotécnico	Código de op (hex)	Descripción
D	DCR *A*	3D	Decrementa *A*
	DCR *B*	05	Decrementa *B*
	DCR *C*	0D	Decrementa *C*
	DCR *D*	15	Decrementa *D*
	DCR *E*	1D	Decrementa *E*
	DCR *H*	25	Decrementa *H*
	DCR *L*	2D	Decrementa *L*
	DCR *M*	35	Decrementa LOC memoria (*H* & *L*)
	DCX *B*	0B	Decrementa *B* & *C*
	DCX *D*	1B	Decrementa *D* & *E*
	DCX *H*	2B	Decrementa *H* & *L*
	DCX *SP*	3B	Decrementa *SP*
	DI	F3	Inhabilita interrupciones
E	EI	FB	Habilita interrupciones
H	HLT	76	Alto hasta interrupción
I	IN *v*	DB	Entrada del dispositivo *v*
	INR *A*	3C	Incrementa *A*
	INR *B*	04	Incrementa *B*
	INR *C*	0C	Incrementa *C*
	INR *D*	14	Incrementa *D*
	INR *E*	1C	Incrementa *E*
	INR *H*	24	Incrementa *H*
	INR *L*	2C	Incrementa *L*
	INR *M*	34	Incrementa LOC memoria (*H* & *L*)
	INX *B*	03	Incrementa *B* & *C*
	INX *D*	13	Incrementa *D* & *E*
	INX *H*	23	Incrementa *H* & *L*
	INX *SP*	33	Incrementa *SP*
J	JMP *aa*	C3	Salta a dirección *aa*
	JZ *aa*	CA	Si cero, JMP en dirección *aa*
	JNZ *aa*	C2	Si no cero, JMP en dirección *aa*
	JP *aa*	F2	Si más, JMP en dirección *aa*
	JM *aa*	FA	Si menos, JMP en dirección *aa*
	JC *aa*	DA	Si arrastre, JMP en dirección *aa*
	JNC *aa*	D2	Si no arrastre, JMP en dirección *aa*
	JPE *aa*	EA	Si paridad par, JMP en dirección *aa*
	JPO *aa*	E2	Si paridad impar, JMP en dirección *aa*
L	LDA *aa*	3A	Carga *A* de dirección *aa*
	LDAX *B*	0A	Carga *A* de LOC memoria (*B* & *C*)
	LDAX *D*	1A	Carga *A* de LOC memoria (*D* & *E*)
	LHLD *aa*	2A	Carga *H* & *L* de dirección *aa*
	LXI *B,vv*	01	Carga *B* & *C* con dato inmediato *vv*
	LXI *D,vv*	11	Carga *D* & *E* con dato inmediato *vv*
	LXI *H,vv*	21	Carga *H* & *L* con dato inmediato *vv*
	LXI *SP,vv*	31	Carga *SP* con dato inmediato *vv*
M	MOV *A,B*	78	Transfiere *B* a *A*
	MOV *A,C*	79	Transfiere *C* a *A*
	MOV *A,D*	7A	Transfiere *D* a *A*
	MOV *A,E*	7B	Transfiere *E* a *A*
	MOV *A,H*	7C	Transfiere *H* a *A*
	MOV *A,L*	7D	Transfiere *L* a *A*
	MOV *A,M*	7E	Transfiere LOC memoria (*H* & *L*) a *A*

Figura 9.1. *(continuación).*

	Nemotécnico	Código de op (hex)	Descripción
M	MOV *B,A*	47	Transfiere *A* a *B*
	MOV *B,C*	41	Transfiere *C* a *B*
	MOV *B,D*	42	Transfiere *D* a *B*
	MOV *B,E*	43	Transfiere *E* a *B*
	MOV *B,H*	44	Transfiere *H* a *B*
	MOV *B,L*	45	Transfiere *L* a *B*
	MOV *B,M*	46	Transfiere LOC memoria (*H* & *L*) a *B*
	MOV *C,A*	4F	Transfiere *A* a *C*
	MOV *C,B*	48	Transfiere *B* a *C*
	MOV *C,D*	4A	Transfiere *D* a *C*
	MOV *C,E*	4B	Transfiere *E* a *C*
	MOV *C,H*	4C	Transfiere *H* a *C*
	MOV *C,L*	4D	Transfiere *L* a *C*
	MOV *C,M*	4E	Transfiere LOC memoria (*H* & *L*) a *C*
	MOV *D,A*	57	Transfiere *A* a *D*
	MOV *D,B*	50	Transfiere *B* a *D*
	MOV *D,C*	51	Transfiere *C* a *D*
	MOV *D,E*	53	Transfiere *E* a *D*
	MOV *D,H*	54	Transfiere *H* a *D*
	MOV *D,L*	55	Transfiere *L* a *D*
	MOV *D,M*	56	Transfiere LOC memoria (*H* & *L*) a *D*
	MOV *E,A*	5F	Transfiere *A* a *E*
	MOV *E,B*	58	Transfiere *B* a *E*
	MOV *E,C*	59	Transfiere *C* a *E*
	MOV *E,D*	5A	Transfiere *D* a *E*
	MOV *E,H*	5C	Transfiere *H* a *E*
	MOV *E,L*	5D	Transfiere *L* a *E*
	MOV *E,M*	5E	Transfiere LOC memoria (*H* & *L*) a *E*
	MOV *H,A*	67	Transfiere *A* a *H*
	MOV *H,B*	60	Transfiere *B* a *H*
	MOV *H,C*	61	Transfiere *C* a *H*
	MOV *H,D*	62	Transfiere *D* a *H*
	MOV *H,E*	63	Transfiere *E* a *H*
	MOV *H,L*	65	Transfiere *L* a *H*
	MOV *H,M*	66	Transfiere LOC memoria (*H* & *L*) a *H*
	MOV *L,A*	6F	Transfiere *A* a *L*
	MOV *L,B*	68	Transfiere *B* a *L*
	MOV *L,C*	69	Transfiere *C* a *L*
	MOV *L,D*	6A	Transfiere *D* a *L*
	MOV *L,E*	6B	Transfiere *E* a *L*
	MOV *L,H*	6C	Transfiere *H* a *L*
	MOV *L,M*	6E	Transfiere LOC memoria (*H* & *L*) a *L*
	MOV *M,A*	77	Transfiere *A* a LOC memoria (*H* & *L*)
	MOV *M,B*	70	Transfiere *B* a LOC memoria (*H* & *L*)
	MOV *M,C*	71	Transfiere *C* a LOC memoria (*H* & *L*)
	MOV *M,D*	72	Transfiere *D* a LOC memoria (*H* & *L*)
	MOV *M,E*	73	Transfiere *E* a LOC memoria (*H* & *L*)
	MOV *M,H*	74	Transfiere *H* a LOC memoria (*H* & *L*)
	MOV *M,L*	75	Transfiere *L* a LOC memoria (*H* & *L*)
	MVI *A,v*	3E	Transfiere dato inmediato *v* a *A*
	MVI *B,v*	06	Transfiere dato inmediato *v* a *B*
	MVI *C,v*	0E	Transfiere dato inmediato *v* a *C*
	MVI *D,v*	16	Transfiere dato inmediato *v* a *D*
	MVI *E,v*	1E	Transfiere dato inmediato *v* a *E*
	MVI *H,v*	26	Transfiere dato inmediato *v* a *H*
	MVI *L,v*	2E	Transfiere dato inmediato *v* a *L*

Figura 9.1. *(continuación).*

	Nemotécnico	Código de op (hex)	Descripción
M	MVI *M,v*	36	Transfiere dato inmediato *v* a memoria LOC (*H* & *L*)
N	NOP	00	No operación
O	ORA *A*	B7	Examina *A* y borra arrastre
	ORA *B*	B0	OR *B* con *A*
	ORA *C*	B1	OR *C* con *A*
	ORA *D*	B2	OR *D* con *A*
	ORA *E*	B3	OR *E* con *A*
	ORA *H*	B4	OR *H* con *A*
	ORA *L*	B5	OR *L* con *A*
	ORA *M*	B6	OR LOC memoria (*H* & *L*) con *A*
	ORI *v*	F6	OR dato inmediato *v* con *A*
	OUT *v*	D3	Saca *A* a dispositivo *v*
P	PCHL	E9	Salta a LOC memoria contenida en (*H* & *L*)
	POP *B*	C1	Saca *B* & *C* de pila
	POP *D*	D1	Saca *D* & *E* de pila
	POP *H*	E1	Saca *H* & *L* de pila
	POP *PSW*	F1	Saca *A* y los señalizadores de pila
	PUSH *B*	C5	Introduce *B* & *C* en pila
	PUSH *D*	D5	Introduce *D* & *E* en pila
	PUSH *H*	E5	Introduce *H* & *L* en pila
	PUSH *PSW*	F5	Introduce *A* y los señalizadores en pila
R	RAL	17	Desplazamiento circular *CY* + *A* a la izquierda
	RAR	1F	Desplazamiento circular *CY* + *A* a la derecha
	RLC	07	Desplazamiento circular *A* a la izquierda y arrastre
	RRC	0F	Desplazamiento circular *A* a la derecha y arrastre
	RIM	20	Lee máscara de interrupción (8085 sólo)
	RET	C9	Vuelve de subrutina
	RZ	C8	Si cero, vuelve de subrutina
	RNZ	C0	Si no cero, vuelve de subrutina
	RP	F0	Si más, vuelve de subrutina
	RM	F8	Si menos, vuelve de subrutina
	RC	D8	Si arrastre, vuelve de subrutina
	RNC	D0	Si no arrastre, vuelve de subrutina
	RPE	E8	Si paridad par, vuelve de subrutina
	RPO	E0	Si paridad impar, vuelve de subrutina
	RST 0	C7	Rearrancar subrutina en dirección 00H
	RST 1	CF	Rearrancar subrutina en dirección 08H
	RST 2	D7	Rearrancar subrutina en dirección 10H
	RST 3	DF	Rearrancar subrutina en dirección 18H
	RST 4	E7	Rearrancar subrutina en dirección 20H
	RST 5	EF	Rearrancar subrutina en dirección 28H
	RST 6	F7	Rearrancar subrutina en dirección 30H
	RST 7	FF	Rearrancar subrutina en dirección 38H
S	SIM	30	Pone a 1 máscara de interrupción (8085 sólo)
	SPHL	F9	Carga *SP* de *H* & *L*
	SHLD *aa*	22	Almacena *H* & *L* en LOC memoria *aa*
	STA *aa*	32	Almacena *A* en LOC memoria *aa*
	STAX *B*	02	Almacena *A* en LOC memoria (*B* & *C*)
	STAX *D*	12	Almacena *A* en LOC memoria (*D* & *E*)
	STC	37	Pone a 1 señalizador de arrastre

Figura 9.1. *(continuación).*

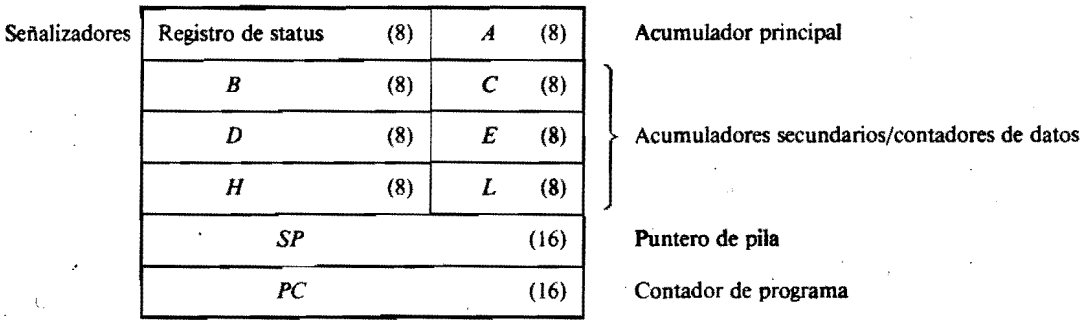
	Nemotécnico	Código de op (hex)	Descripción
S	SUB *A*	97	Borra *A*
	SUB *B*	90	Resta *B* de *A*
	SUB *C*	91	Resta *C* de *A*
	SUB *D*	92	Resta *D* de *A*
	SUB *E*	93	Resta *E* de *A*
	SUB *H*	94	Resta *H* de *A*
	SUB *L*	95	Resta *L* de *A*
	SUB *M*	96	Resta contenido de LOC memoria (*H* & *L*) de *A*
	SUI *v*	D6	Resta dato inmediato *v* de *A*
	SBB *A*	9F	Pone *A* con arrastre menos
	SBB *B*	98	Resta *B* de *A* con préstamo
	SBB *C*	99	Resta *C* de *A* con préstamo
	SBB *D*	9A	Resta *D* de *A* con préstamo
	SBB *E*	9B	Resta *E* de *A* con préstamo
	SBB *H*	9C	Resta *H* de *A* con préstamo
	SBB *L*	9D	Resta *L* de *A* con préstamo
	SBB *M*	9E	Resta LOC memoria (*H* & *L*) de *A* con préstamo
	SBI *v*	DE	Resta dato inmediato *v* de *A* con préstamo
X	XCHG	EB	Cambia *D* & *E* con *H* & *L*
	XTHL	E3	Cambia tope de pila con *H* & *L*
	XRA *A*	AF	Borra *A*
	XRA *B*	A8	OR exclusiva de *B* con *A*
	XRA *C*	A9	OR exclusiva de *C* con *A*
	XRA *D*	AA	OR exclusiva de *D* con *A*
	XRA *E*	AB	OR exclusiva de *E* con *A*
	XRA *H*	AC	OR exclusiva de *H* con *A*
	XRA *L*	AD	OR exclusiva de *L* con *A*
	XRA *M*	AE	OR exclusiva de LOC memoria (*H* & *L*) con *A*
	XRI *v*	EE	OR exclusiva de dato inmediato *v* con *A*

Figura 9.1. *(continuación).*

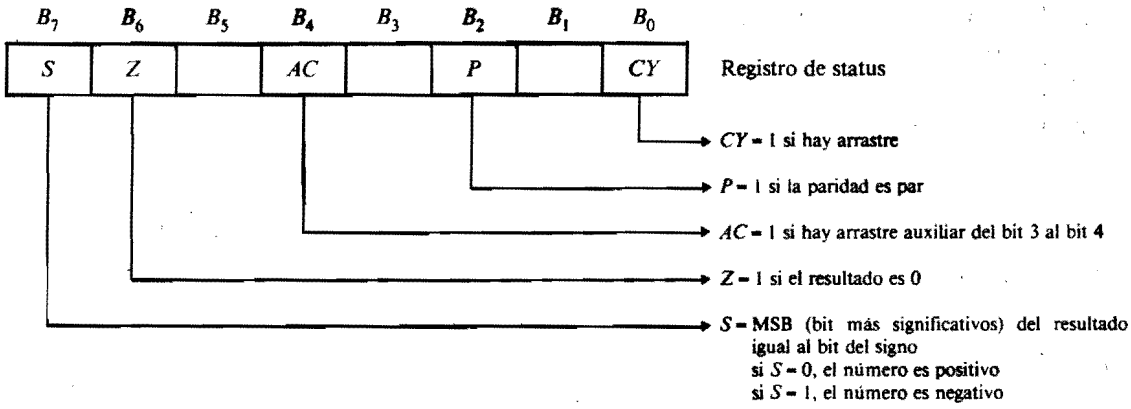
2. Campo código de op: Contiene la entrada MOV. Esta dice a la CPU que es una instrucción de transferencia de datos. La CPU entonces busca el siguiente campo de operando para encontrar la fuente y destino de la transferencia del dato.
3. Campo de operando: Contiene la entrada *A,M*. *A* (acumulador) es el destino del dato mientras que la memoria (*M*) es la fuente. La posición de memoria está indicada por el registro par *HL* en esta instrucción de registro indirecto.
4. Campo comentario: Contiene la entrada «; Entrar datos al acumulador». Este campo describe qué ocurre cuando se ejecuta la instrucción. En este caso, el dato es introducido en el acumulador desde una posición de memoria señalada por el registro par *HL*. El punto y coma (;) se utiliza como *delimitador* para separar el campo del operando del campo de comentario.

Los campos de código de op y operando deben rellenarse, pero los campos de etiqueta y comentario son opcionales. Algunas instrucciones inherentes requieren una entrada en el campo de código de op y ninguna en el campo de operando.

Recordar del Capítulo 6 que los diagramas de flujo, programas fuente en lenguaje ensamblador y ensambladores son todos ayudas en la programación de un sistema basado en microprocesador. El listado del programa objeto ensamblado es el resultado final deseado utilizado por la microcomputadora.



(a) Modelo de programación para el microprocesador 8080



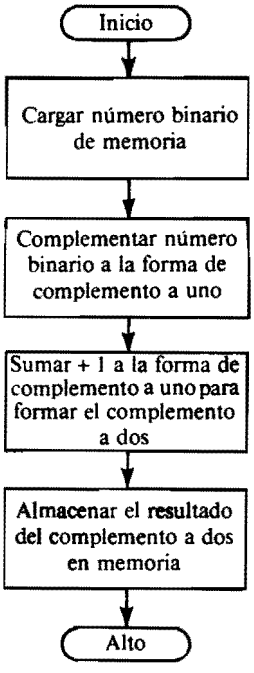
(b) Señalizadores del registro de status del 8080/8085

Figura 9.2.

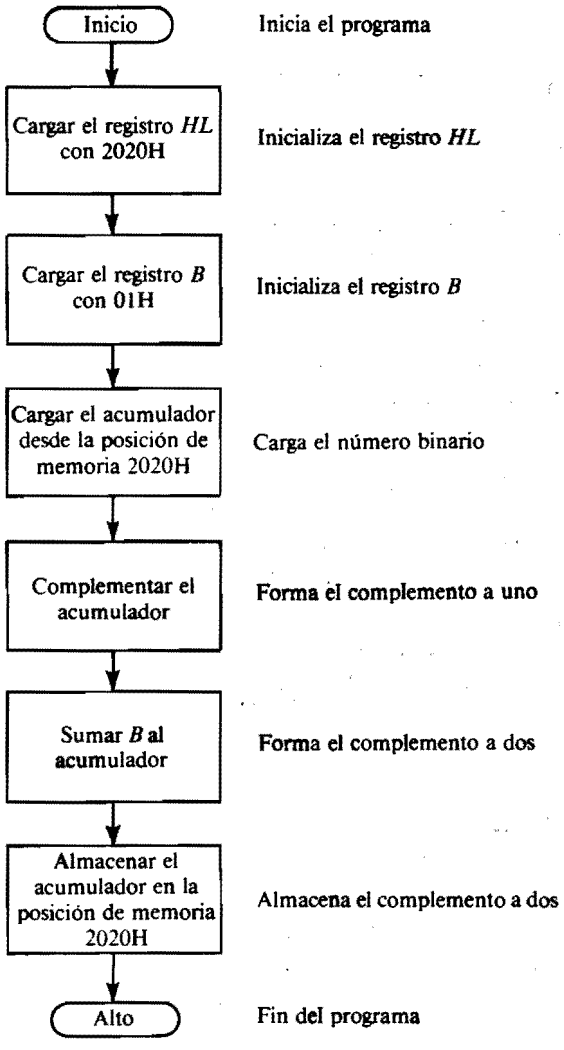
9.2. PROGRAMAS LINEALES

Considerar el diagrama de flujo funcional de la Figura 9.3*a*. Representa un *programa lineal* ya que no contiene bifurcaciones. El diagrama de flujo también representa un programa que carga un número binario, lo convierte a su forma en complemento a 2 y finalmente almacena la representación en complemento a 2 en memoria. Recordar que los diagramas funcionales como el de la Figura 9.3*a* son una descripción general de la solución del problema.

Un diagrama detallado para el mismo problema se muestra en la Figura 9.3*b*. Este diagrama se completó con las funciones de un microprocesador específico —la MPU 8080/8085 en este caso—. Del diagrama detallado de la Figura 9.3*b*, puede obtenerse un programa en lenguaje ensamblador, el cual aparece en la Figura 9.4*a*. Observar que cada instrucción del programa tiene



(a) Diagrama para el programa binario -a-complemento-a-2



(b) Diagrama detallado para el programa binario -a-complemento-a-2

Figura 9.3.

un bloque rectangular comparable en la Figura 9.3*b*. Como ejemplo, el primer bloque rectangular de la Figura 9.3*b*, que dice «cargar el registro par *HL* con 2020H», se corresponde con la sentencia del programa ensamblador «LXI *H*, 2020H» en la Figura 9.4.

El siguiente paso será traducir el programa fuente en lenguaje ensamblador de la Figura 9.4*a* a su equivalente en lenguaje máquina. Este «ensamblamiento» o proceso de traducción puede hacerse utilizando un programa especial de computadora denominado ensamblador o puede hacerse manualmente. El repertorio de instrucciones de la Figura 9.1 puede ser utilizado para

Etiqueta	Código op	Operando	Comentarios
	LXI	*H*,2020H	; Inicializa registro par *HL* con 2020H
	MVI	*B*,01H	; Inicializa registro *B* con 01H
	MOV	*A,M*	; Transfiere número binario de posición de memoria 2020H al acumulador
	CMA		; Complementa acumulador
	ADD	*B*	; Suma *B* a *A* para formar complemento a 2
	MOV	*M,A*	; Transfiere número complemento a 2 del acumulador a posición de memoria 2020H
	HLT		; Detiene la MPU

(a) Programa en lenguaje ensamblador para el problema binario-a-complemento-a-2

Dirección (hex)	Contenido (hex)	Etiqueta	Código op	Operando	Comentarios
2000	21		LXI	*H*,2020H	; Inicializa registro par *HL* con 2020H
2001	20				
2002	20				
2003	06		MVI	*B*,01H	; Inicializa registro *B* con 01H
2004	01				
2005	7E		MOV	*A,M*	; Transfiere número binario de posición de memoria 2020H al acumulador
2006	2F		CMA		; Complementa acumulador
2007	80		ADD		; Suma *B* a *A* para formar complemento a 2
2008	77		MOV	*M,A*	; Transfiere número complemento a 2 de acumulador a posición de memoria 2020H
2009	76		HLT		; Detiene la MPU

(b) Programa objeto para el problema binario-a-complemento-a-2

Figura 9.4.

traducir manualmente el programa en lenguaje ensamblador. El programa de la Figura 9.4*b* tiene las direcciones de la memoria de programa y el listado del código máquina hexadecimal para cada instrucción y operando. La sección no sombreada a la izquierda de la Figura 9.4*b* representa el código máquina tal como puede aparecer en la memoria de programa.

Observar que el programa listado en la Figura 9.4*b* no contiene entradas en el campo de etiqueta. Esto es porque las etiquetas se utilizan habitualmente para mostrar el destino de una instrucción de bifurcación. El programa de la Figura 9.4 representa un programa lineal y por tanto no contiene saltos.

PROBLEMAS RESUELTOS

9.1. El listado de la Figura 9.4 es un ejemplo de programa _____ (de bifurcación, lineal).

Solución:

El listado de la Figura 9.4 es un ejemplo de programa lineal.

9.2. Acudir a la Figura 9.5. El resultado encontrado en la memoria de datos después de correr el programa listado en la Figura 9.4 será _____ (8 bits). Este resultado será la representación en complemento a _____ (1, 2) de 00000010_2.

Solución:

El resultado que se encuentre en la memoria de datos de la Figura 9.5 después de correr el programa será 11111110. Este resultado es la representación en complemento a 2 de 00000010_2.

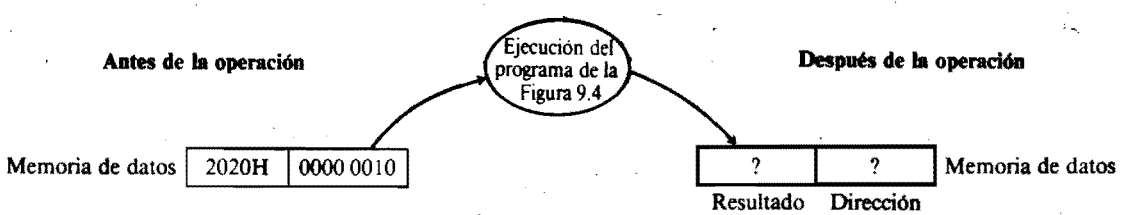


Figura 9.5.

9.3. Acudir a la Figura 9.5. La dirección de la memoria de datos donde se depositará el resultado después de la ejecución del programa será _____ (hex).

Solución:

El registro par *HL* apuntará todavía a la posición 2020H después del programa. Por tanto la dirección en la memoria de datos de la Figura 9.5 donde el resultado se depositará después de la ejecución del programa será 2020H.

9.4. El diagrama de la Figura 9.6 representa un segmento de programa que introduce un número ASCII y enmascara los 4 bits _____ (menos, más) significativos. El resultado es el equivalente _____ (BCD, binario) del número ASCII.

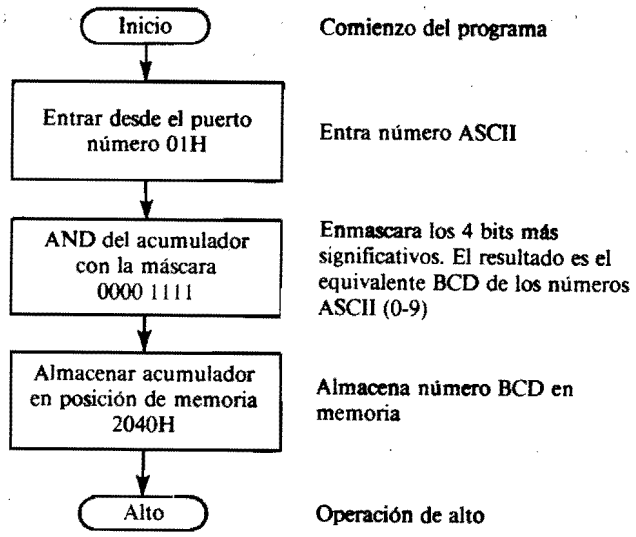


Figura 9.6. Diagrama detallado del problema.

Solución:

El diagrama de la Figura 9.6 representa un segmento de programa que introduce un número ASCII y enmascara los 4 bits más significativos. El resultado es el equivalente BCD del número ASCII.

9.5. Basándose en el diagrama detallado en la Figura 9.6, completar las líneas 2 y 3 del segmento de programa del 8080/8085 mostrado más abajo.

Etiqueta	Código de op	Operando	Comentarios
LINEA 1	IN	01H	; Entrada número ASCII desde puerto 01H
LINEA 2	—	—	; AND acum. con 00001111 (0FH)
LINEA 3	—	—	; Almacenar acum. en posición 2040H
LINEA 4	HLT		; Parar MPU

Solución:

Un posible segmento de programa que realice la tarea detallada en la Figura 9.6 es el siguiente:

Etiqueta	Código de op	Operando	Comentarios
LINEA 1	IN	01H	; Entrada número ASCII desde puerto 01H
LINEA 2	ANI	0FH	; AND acum. con 00001111 (0FH)
LINEA 3	STA	2040H	; Almacenar acum. en posición 2040H
LINEA 4	HLT		; Parar MPU

9.6. Listar el código máquina hexadecimal del 8080/8085 requerido para ejecutar el programa del Problema 9.5. El programa comenzará en la posición de memoria 2000H como sigue:

Posición de memoria (hex)	Contenido (hex)	
2000	BD	instrucción de entrada
2001	01	
2002	—	
2003	—	
2004	—	
—	—	
—	—	

Solución:

Acudir al repertorio de instrucciones de la Figura 9.1 para los códigos de op. Lo que sigue es un listado en código máquina 8080/8085 que ejecutará el programa del Problema 9.5:

Posición de memoria (hex)	Contenido (hex)
2000	BD
2001	01
2002	E6
2003	0F
2004	32
2005	40
2006	20
2007	76

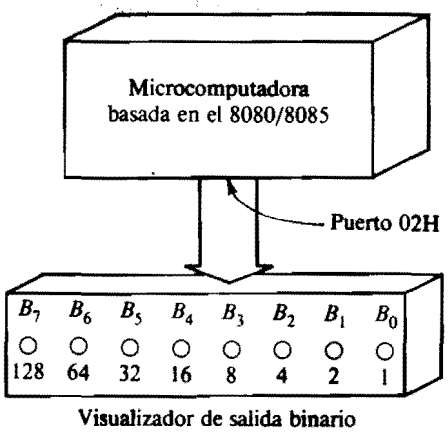
9.3. PROGRAMAS CICLICOS

Considerar el diagrama de bloques de la Figura 9.7*a* de un sistema microcomputador basado en el 8080/8085. Suponer que el visualizador de 8 bits binario cuenta de forma ascendente empezando en 00000000_2. La Figura 9.7*b* muestra un diagrama detallado que resuelve el problema de cuenta binaria. Un segmento del programa en el lenguaje ensamblador 8080/8085 que realizará la función de cuenta binaria está en la Figura 9.7*c*.

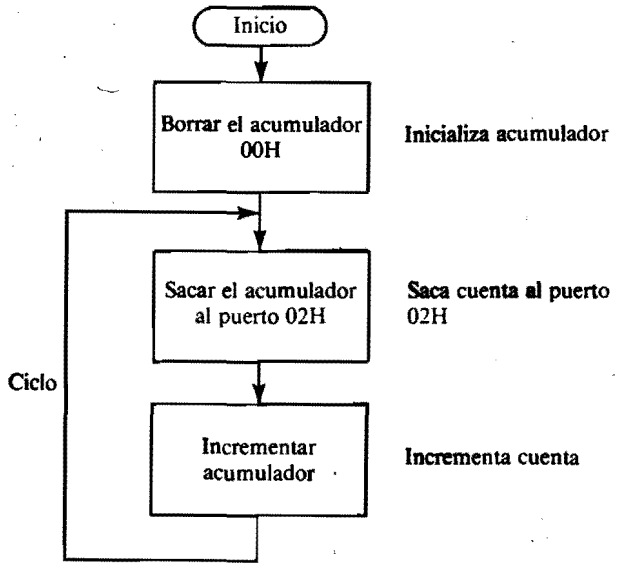
El primer bloque rectangular del diagrama de la Figura 9.7*b* muestra que el acumulador se inicializa con 00H. El segundo bloque representa la transferencia de la cuenta del acumulador al puerto de salida 02H. El tercer bloque muestra que la cuenta se incrementa en 1 por el acumulador. A continuación el programa vuelve a enlazar al bloque de salida. El programa visualizará a gran velocidad, en binario, la salida de la cuenta ascendente.

El programa en el lenguaje ensamblador 8080/8085 de la Figura 9.7*c* realizará la función de cuenta binaria representada en el diagrama anterior. La instrucción XRA *A* borra el acumulador. La instrucción OUT transfiere el contenido del acumulador al puerto de salida 02H. La cuenta en el acumulador es incrementada en 1 utilizando la instrucción INR *A*. La instrucción JMP hace que la MPU bifurque, hacia atrás, a la instrucción con la etiqueta simbólica LOOP. El programa entonces está en un ciclo continuo a través de la secuencia OUT, INR *A* y JMP.

En la práctica actual, el programa representado en la Figura 9.7 contará mucho más rápido de lo que una persona pueda observar. Debido al rápido secuenciamiento, las lámparas de visualización de la salida binaria parecerán que están luciendo continuamente. Para ralentizar el proceso de cuenta, es habitual añadir un *retardo de tiempo* en el programa. El diagrama de la Figura 9.8*a* representa un programa de cuenta binaria que incluye un retardo de tiempo.



(a) Diagrama de bloque simplificado del sistema para visualizar una cuenta bancaria



(b) Diagrama detallado para el problema de cuenta binaria

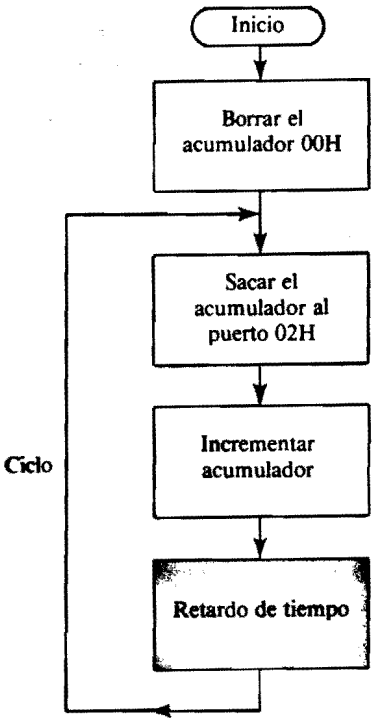
Etiqueta	Código op	Operando	Comentarios
	XRA	*A*	; Borra el acumulador
LOOP	OUT	02H	; Saca el acumulador al puerto 02H
	INR	*A*	; Incrementa acumulador
	JMP	LOOP	; Salta hacia atrás al comienzo del ciclo (dirección simbólica = LOOP)

(c) Programa en lenguaje ensamblador para el problema de cuenta binaria

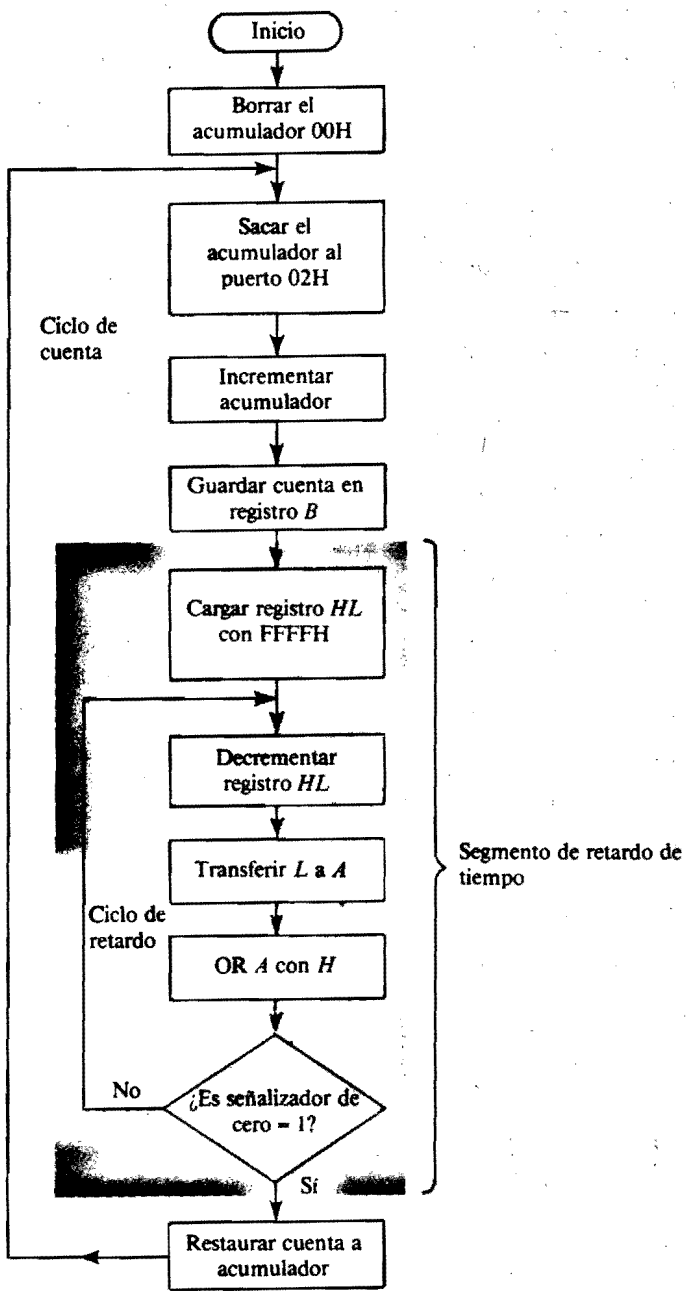
Figura 9.7.

El diagrama detallado de la Figura 9.8*b* representa el programa de cuenta binaria con un retardo de tiempo. Los cinco bloques de la parte inferior del diagrama representan el segmento de retardo de tiempo del programa. El retardo de tiempo se genera al repetir el lazo de retardo 65.535 (FFFFH) veces.

El programa en lenguaje ensamblador para el programa de cuenta con el retardo de tiempo se muestra en la Figura 9.8*c*. Observar que cada bloque en el diagrama de flujo detallado (Fig. 9.8*b*) se implementa con una simple instrucción de lenguaje ensamblador. El ciclo del retardo lo da la



(a) Diagrama para el problema de cuenta binaria con un retardo de tiempo adicional



(b) Diagrama detallado para el problema de cuenta binaria con retardo de tiempo

Figura 9.8.

Etiqueta	Código op	Operando	Comentarios
	XRA	*A*	; Borra el acumulador 00H
COUNT	OUT	02H	; Saca el acumulador al puerto 02H
	INR	*A*	; Incrementa acumulador
	MOV	*B,A*	; Guarda cuenta en registro *B*
	LXI	*H*,FFFFH	; Carga registro *HL* con FFFFH
DELAY	DCX	*H*	; Decrementa registro *HL*
	MOV	*A,L*	; Transfiere registro *L* al acumulador para examen
	ORA	*H*	; Operación lógica OR del registro *H* con el acumulador; pone el señalizador *Z* a 1 si el registro *H* y *A* contienen ceros
	JNZ	DELAY	; Si el señalizador *Z* = 0, entonces salta a posición DELAY (ciclo de retardo); en otro caso continuar
	MOV	*A,B*	; Restaura cuenta (transferir de *B* al acumulador)
	JMP	COUNT	; Salta a posición COUNT (ciclo de cuenta)

(c) **Programa en lenguaje** ensamblador para el problema de cuenta binaria con retardo de tiempo

Figura 9.8. *(continuación).*

dirección simbólica o etiqueta DELAY en el programa de la Figura 9.8*c*. El ciclo mayor de cuenta lo da la dirección simbólica COUNT. El programa realizará ciclos a través del lazo de retardo 65.536 veces cada vez a través del ciclo de cuenta. Las instrucciones DCX *H*, MOV *A,L*, ORA *H* y JNZ en conjunto consumen unos 24 μs cuando utilizan la MPU 8085. Como el ciclo de retardo se repite 65.536 veces, el tiempo de retardo es

$$24\ \mu s \times 65.536 = 1.572.864\ \mu s$$

El segmento de retardo de tiempo del programa produce un retardo de 1.6 s aproximadamente. El programa visualiza la primera cuenta, se retarda 1.6 s, visualiza la segunda cuenta, se retarda 1.6 s, etc.

El propósito de las instrucciones MOV *A,L* y ORA *H* en el programa en lenguaje ensamblador de la Figura 9.8*c* puede no parecer claro de inmediato. Estas dos instrucciones están incluidas para poner a 1 el señalizador *Z* cuando el registro par *HL* sea finalmente decrementado a 0000H. Desgraciadamente, la operación previa «decrementar registro par *HL*» (instrucción DCX *H*) *no tiene efecto en ninguno de los señalizadores,* por lo que se han añadido las instruciones MOV *A,L* y ORA *H*.

Las rutinas de retardo de tiempo son comúnmente utilizadas en los programas de microcomputadora. Están basadas en la idea de que cada instrucción del procesador utiliza una cantidad específica de tiempo para su ejecución. Las técnicas de programación disponibles permiten generar retardos de tiempo mayores que el de 1.6 s, generado en el último ejemplo.

Considerar el uso de dos ciclos en el programa representado en la Figura 9.8*b*. Un ciclo está contenido en el otro. Esta técnica de programación se denomina *anidamiento.* El ciclo de retardo de la Figura 9.8*b* se dice que es el anidado, o ciclo interior. El proceso de repetir instrucciones hasta que se cumpla una condición, como en el ciclo anidado, se denomina *iteración.*

PROBLEMAS RESUELTOS

9.7. La última instrucción del programa de la Figura 9.7*c* es JMP. ¿A qué dirección debe saltar el programa debido a esta instrucción?

Solución:

La instrucción «JMP LOOP» de la Figura 9.7*c* indica al procesador que salte a la dirección simbólica LOOP que corresponde a la instrucción OUT de este ejemplo.

9.8. Listar las cinco instrucciones de la Figura 9.8*c* que forman el segmento de retardo de tiempo del programa.

Solución:

Las instrucciones LXI *H*; DCX *H*; MOV *A,L*; ORA *H*; y JNZ de la Figura 9.8*c* forman el segmento de retardo de tiempo del programa. Este segmento está específicamente identificado en el diagrama detallado de la Figura 9.8*b*.

9.9. El ciclo de retardo de tiempo representado en el diagrama de la Figura 9.8*b* los programadores lo denominan ciclo _____ (infinito, anidado).

Solución:

El ciclo de retardo de tiempo representado en el diagrama de la Figura 9.8*b* se denomina ciclo anidado.

9.10. Acudir a la Figura 9.8*c*. La primera vez que el procesador 8080/8085 se encuentra la instrucción JNZ en este programa, el señalizador *Z* estará a _____ (0, 1). Después de ejecutar la instrucción JNZ, el procesador salta entonces a la siguiente instrucción, que es _____ .

Solución:

La primera vez que se encuentra la instrucción JNZ en la Figura 9.8*c*, el señalizador *Z* estará a 0. El procesador entonces salta a la siguiente instrucción, que es DCX *H* (a la posición con la dirección simbólica DELAY).

9.11. Acudir a la Figura 9.8*c*. Cuando el registro par *HL* ha sido decrementado a 0000H por la instrucción DCX *H*, el señalizador *Z* está a _____ (0, 1) por la operación ORA *H*. Basándose en la condición de este señalizador, la instrucción JNZ es seguida por la operación _____ .

Solución:

Cuando el registro par *HL* de la Figura 9.8*c* ha sido decrementado a 0000H por la instrucción DCX *H*, el señalizador *Z* está a 1 por la operación ORA *H*. Basándose en esta condición del señalizador, la instrucción JNZ es seguida por la operación MOV *A,B*.

9.12. Listar el código máquina hexadecimal 8080/8085 requerido para ejecutar el programa de la Figura 9.8*c*. El programa comenzará en la posición de memoria 2000H como sigue:

Posición de memoria (hex)	Contenido (hex)	
2000	AF	borra acumulador
2001	—	
2002	—	
—	—	
—	—	

Solución:

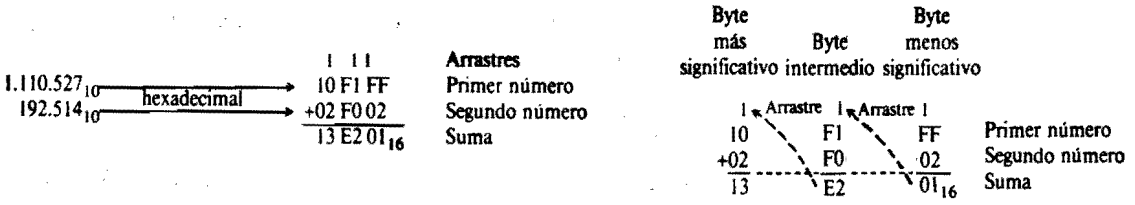
Acudir al repertorio de instrucciones de la Figura 9.1. Lo que sigue es un listado del código máquina 8080/8085 que ejecutará el programa de la Figura 9.8*c*:

Posición de memoria (hex)	Contenido (hex)
2000	AF
2001	D3
2002	02
2003	3C
2004	47
2005	21
2006	FF
2007	FF
2008	2B
2009	7D
200A	B4
200B	C2
200C	08
200D	20
200E	78
200F	C3
2010	01
2011	20

9.4. PROGRAMAS MATEMATICOS

Considerar el problema de sumar los números decimales 1.110.527 y 192.514 utilizando un sistema basado en el 8080/8085. Convertir estos números a hexadecimales, la solución manual a este problema de suma puede ser como la de la Figura 9.9*a*. La suma hexadecimal de $10F1FF + 02F002 = 13E201_{16}$.

El acumulador del microprocesador 8080/8085 maneja solamente datos de 8 bits (un byte) cada vez. El procesador probablemente podría manipular este problema si el programador dividiese el número hexadecimal en grupos de longitud-de-un-byte. Cada grupo, más pequeño, de longitud-de-un-byte puede entonces ser sumado separadamente. Un ejemplo de esta técnica se muestra en la Figura 9.9*b* ($FF + 02 = 01_{16}$ más un arrastre). A continuación, se suman los bytes intermedios (arrastre $1 + F1 + F0 = E2_{16}$ más un arrastre). Finalmente, se suman los bytes más significativos (arrastre $1 + 10 + 02 = 13_{16}$). Observar que en las dos últimas sumas, los arrastres de los bytes menos significativos deben ser considerados al calcular la suma. El microprocesador 8080/8085 tiene instrucciones especiales de «suma con arrastre» para este propósito.



(a) Ejemplo de suma hexadecimal

(b) Suma de 24 bits agrupada en bytes tal y como la ejecuta un microprocesador de 8 bits

Figura 9.9.

El procedimiento para realizar la suma de 24 bits (suma de 3 bytes) puede ser como sigue:

1. Sumar el byte menos significativo del primer número con el byte menos significativo del segundo número.
2. Registrar la suma parcial menos significativa.
3. Sumar el arrastre de la suma previa, el byte intermedio del primer número y el byte intermedio del segundo número.
4. Registrar la suma parcial intermedia.
5. Sumar el arrastre de la suma anterior, el byte más significativo del primer número y el byte más significativo del segundo número.
6. Registrar la suma parcial más significativa.

En el problema de suma, suponer que el primero y segundo números tienen asignadas las posiciones de memoria que se indican en la Figura 9.10*a*. La posición del resultado de la suma de 24 bits también está detallada en el mapa de memoria de la Figura 9.10*a*. Utilizando estas posiciones de la memoria de datos, el programa lineal en lenguaje ensamblador de la Figura 9.10*b* se desarrolló para resolver este problema de suma de 3 bytes. Observar que la primera instrucción

ADD *M* es una operación de suma sin arrastre. Sin embargo, las instrucciones segunda y tercera ADC *M* son operaciones de suma con arrastre.

La suma de números mayores requiere el uso de suma multibyte. En este ejemplo los números sumados eran de 3 bytes. A este tipo de suma se denomina *suma en triple precisión.*

PROBLEMAS RESUELTOS

9.13. El programa listado en la Figura 9.10*b* es un ejemplo de suma en _____ (doble, triple) precisión.

Solución:

El programa listado en la Figura 9.10*b* es un ejemplo de suma en triple precisión.

9.14. Suponer que $10F1FF_{16}$ se va a sumar a 02F002, como en la Figura 9.9*b*. Listar los contenidos de las posiciones de la memoria de datos 2020 a 2025H en la Figura 9.10*a* antes de que se ejecute el programa de suma multibyte.

Memoria de datos

Dirección (hex)	Contenido	
2020		Byte menos significativo del primer número
2021		Byte menos significativo del segundo número
2022		Byte intermedio del primer número
2023		Byte intermedio del segundo número
2024		Byte más significativo del primer número
2025		Byte más significativo del segundo número
2026		Suma: Byte menos significativo
2027		Suma: Byte intermedio
2028		Suma: Byte más significativo

(a) Mapa de memoria de datos para el programa de suma de 24 bits

Figura 9.10.

Etiqueta	Código op	Operando	Comentarios
	LXI	*H*,2020H	; Inicializa registro par *HL* con 2020H (registro par *HL* utilizado como puntero)
	MOV	*A,M*	; Carga byte menos significativo del primer número de la posición de memoria LOC 2020H en acumulador
	INX	*H*	; Incrementa puntero *HL* a 2021H
	ADD	*M*	; Suma posición de memoria 2021H al acumulador (suma byte menos significativo)
	STA	2026H	; Almacena suma parcial menos significativa en posición de memoria LOC 2026H
	INX	*H*	; Incrementa puntero *HL* a 2022H
	MOV	*A,M*	; Carga byte intermedio del primer número de la posición de memoria 2022H en acumulador
	INX	*H*	; Incrementa puntero *HL* a 2023H
	ADC	*M*	; Suma posición de memoria 2023H, acumulador y arrastre (suma bytes intermedios con arrastre)
	STA	2027H	; Almacena suma parcial intermedia en posición de memoria 2027H
	INX	*H*	; Incrementa puntero *HL* en 2024H
	MOV	*A,M*	; Carga byte más significativo del primer número de la posición de memoria 2024H en acumulador
	INX	*H*	; Incrementa puntero *HL* a 2025H
	ADC	*M*	; Suma posición de memoria 2025H, acumulador y arrastre (suma bytes más significativos con arrastre)
	STA	2028H	; Almacena suma parcial más significativa en posición de memoria 2028H
	HLT		; Alto la MPU

(b) Programa en lenguaje ensamblador para la suma de 24 bits

Figura 9.10. ***(continuación).***

Solución:

Los contenidos de la memoria de datos de la Figura 9.10*a* antes de ejecutar el programa para sumar $10F1FF_{16} + 02F002_{16}$ son los siguientes:

Dirección (hex)	Contenido (hex)
2020	FF
2021	02
2022	F1
2023	F0
2024	10
2025	02

9.15. Listar los contenidos de las posiciones de memoria 2026H, 2027H y 2028H en la Figura 9.10 después que el programa se haya ejecutado sumando 10F1FF y 02F002.

Solución:

Las posiciones de la memoria de datos de la Figura 9.10 («suma») contienen lo siguiente después que el programa se haya ejecutado ($10F1FF + 02F002 = 13E201_{16}$):

Dirección (hex)	Contenido (hex)	
2026	01	(byte de la suma menos significativo)
2027	E2	(byte intermedio de la suma)
2028	13	(byte más significativo de la suma)

9.16. Acudir a la Figura 9.10*b*. El registro par *HL* es utilizado como registro _____ (de propósito general, puntero) en este programa de suma.

Solución:

El registro par *HL* es utilizado como puntero de dirección en el programa de suma de la Figura 9.10*b*.

9.17. La instrucción ADD *M* mostrada en la Figura 9.10*b* es una operación de _____ (suma, suma con arrastre).

Solución:

La instrucción ADD *M* mostrada en la Figura 9.10*b* es una operación de suma.

9.18. La instrucción STA 2027H de la Figura 9.10*b* hace que el contenido del registro _____ (*A*, *B*) sea almacenado en la posición de memoria _____ (hex).

Solución:

La instrucción STA 2027H de la Figura 9.10*b* hace que el contenido del registro *A* (acumulador) sea almacenado en la posición de memoria 2027H.

9.19. Acudir a la Figura 9.11. El primer número que se va a sumar está almacenado en ¿qué posiciones de memoria cuando se utiliza este programa?

Solución:

De acuerdo con el mapa de memoria de la Figura 9.11*a*, el primer número de 4 bytes que se va a sumar se almacena en las posiciones de memoria 2020H a 2023H.

9.20. Después que se ejecute el programa de suma multibyte de la Figura 9.11 ¿en qué cuatro posiciones de memoria se almacena la suma?

Solución:

La instrucción STAX *D* almacena las sumas parciales en las posiciones de memoria señaladas por el registro par *DE*. La suma se almacena por tanto en las posiciones de memoria 2020H a 2023H después de que se haya ejecutado el programa.

9.21. ¿Cómo debe cambiarse la instrucción MVI *C* de la Figura 9.11*b* para realizar la suma de 24 bits en lugar de la de 32?

Solución:

La instrucción debe ser «MVI *C*,03H» si el programa de la Figura 9.11*b* va a realizar una suma de 24 bits. El operando de la instrucción MVI *C* (03H en este caso) es el número de bytes que se va a sumar.

9.22. Acudir a la Figura 9.11. Listar el contenido de las siguientes posiciones de memoria cuando se sumen $12F0C3FF_{16}$ y $01B4D503_{16}$.

Memoria de datos

Dirección (hex)	Contenido	
2020		Byte menos significativo
2021		Primer número
2022		
2023		Byte más significativo
2030		Byte menos significativo
2031		Segundo número
2032		
2033		Byte más significativo

(a) Mapa de memoria de datos para el programa de la suma de 32 bits

Figura 9.11.

Etiqueta	Código op	Operando	Comentarios
	MVI	*C*,04H	; Inicializa registro *C* con número de bytes que se van a sumar
	LXI	*D*,2020H	; Inicializa puntero *DE* a posición del byte menos significativo del primer número
	LXI	*H*,2030H	; Inicializa puntero *HL* a posición del byte menos significativo del segundo número
	XRA	*A*	; Borra señalizador de arrastre
LOOP	LDAX	*D*	; Carga byte del primer número en el acumulador
	ADC	*M*	; Suma byte del segundo número, acumulador y bit de arrastre
	STAX	*D*	; Almacena suma parcial en posición de memoria señalada por el registro *DE*
	DCR	*C*	; Decrementa registro *C*
	JZ	END	; Si señalizador $Z = 1$, entonces salta a dirección simbólica etiquetada END
	INX	*D*	; Incrementa puntero *DE*
	INX	*H*	; Incrementa puntero *HL*
	JMP	LOOP	; Salta siempre a dirección simbólica etiquetada LOOP
END	HLT		; Alto la MPU

(b) **Programa en lenguaje ensamblador para el problema-ejemplo**

Figura 9.11. *(continuación).*

Posición de memoria (hex)	Contenido (hex)
2020	FF
2021	—
2022	—
2023	—
2030	—
2031	—
2032	—
2033	—

Solución:

Antes de sumar $12F0C3FF_{16}$ a $01B4D503_{16}$, la memoria de datos mostrada en la Figura 9.11 debe ser cargada con los siguientes datos:

Posición de memoria (hex)	Contenido (hex)
2020	FF
2021	C3
2022	F0
2023	12
2030	03
2031	D5
2032	B4
2033	01

9.23. ¿Cuál es el resultado de sumar $12F0C3FF_{16}$ y $01B4D503_{16}$? Listar los contenidos y posiciones donde se almacenará esta suma multibyte una vez que se ejecute el programa de la Figura 9.11.

Solución:

La suma hexadecimal 12F0C3FF + 01B4D503 = $14A59902_{16}$. El resultado (suma) del programa de la Figura 9.11 será almacenado como sigue en la memoria de datos:

Posición de memoria (hex)		Contenido (hex)
2020	02	(byte menos significativo de la suma)
2021	99	
2022	A5	
2033	14	(byte más significativo de la suma)

9.24. Acudir a la Figura 9.11*b*. Si el señalizador $Z = 0$, entonces la instrucción _____ será ejecutada después de la instrucción JZ en este programa.

Solución:

Si el señalizador $Z = 0$ en la Figura 9.11*b*, entonces la instrucción INX *D* será ejecutada después de la instrucción JZ en este programa.

PROBLEMAS SUPLEMENTARIOS

9.25. Para el programador, la microcomputadora basada en el 8080/8085 consta de memoria, puertos de entrada/salida, señalizadores, pila, puntero de pila, contador _____, repertorio _____ y registros de propósito general.

Res. de programa, de instrucciones.

9.26. Citar los cuatro campos de una línea de instrucción de lenguaje ensamblador.

Res. etiqueta, código de op (operación), operando, comentarios.

9.27. El campo de operando se denomina a veces _____.

Res. argumento.

9.28. Las sentencias en lenguaje ensamblador contienen, habitualmente, entradas en los campos _____ y _____, mientras que los campos de etiqueta y comentarios son opcionales.

Res. código de op (operación), operando.

9.29. El programa del 8080/8085 representado en la Figura 9.12 carga los números ASCII de las posiciones de memoria _____ (hex) y _____ (hex), enmascara los 4 bits _____ (menos, más) significativos, suma los números, ajusta en decimal la suma y almacena la suma _____ (BCD, binaria) en la posición de memoria 2032H.

Res. 2030H, 2031H, más, BCD.

9.30. La función representada por el bloque 1 de la Figura 9.12*b* podría producirse por ¿qué instrucción del 8080/8085?

Res. LXI *H*, 2030H.

9.31. La función representada por el bloque 2 de la Figura 9.12*b* podría producirse por ¿qué instrucción del 8080/8085?

Res. MOV *A,M.*

9.32. La función representada por el bloque 3 de la Figura 9.12*b* podría producirse por ¿qué instrucción del 8080/8085?

Res. ANI 0FH.

9.33. La función representada por el bloque 4 de la Figura 9.12*b* podría producirse por ¿qué instrucción del 8080/8085?

Res. MOV *B,A.*

9.34. La función representada por el bloque 5 de la Figura 9.12*b* podría producirse por ¿qué instrucción del 8080/8085?

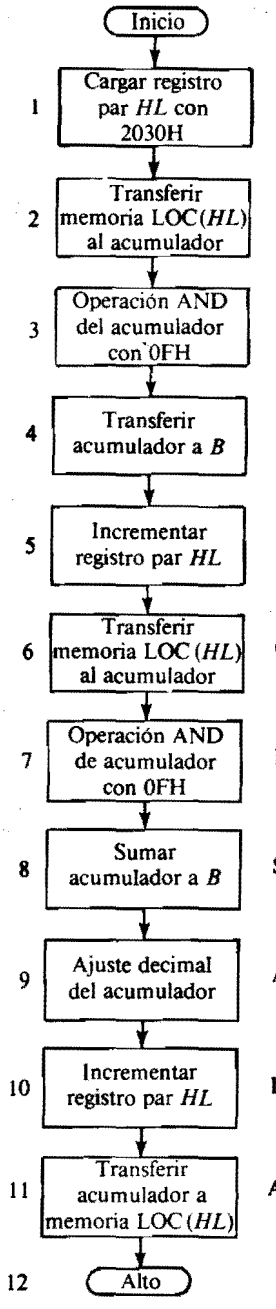
Res. INX *H.*

Memoria de datos

Dirección (hex)	Contenido (hex)	
2030	B5	Representación ASCII del número 5 BCD
2031	B9	Representación ASCII del número 9 BCD
2032		← Suma BCD

(a) Mapa de memoria de datos para el problema ejemplo

Figura 9.12.



Inicializa registro par *HL* con 2030H

Carga acumulador con primer número ASCII desde 2030H

Enmascara los cuatro bits más significativos del acumulador

Guarda número BCD en registro *B*

Incrementa registro par *HL* a 2031H

Carga acumulador con segundo número ASCII desde 2031H

Enmascara los cuatro bits más significativos del acumulador

Suma números BCD en los registros *A* y *B*

Ajusta la suma en *A* para que sea igual al número BCD

Incrementa registro par *HL* a 2032H

Almacena la suma BCD en posición de memoria 2032H

(b) Diagrama detallado para el problema ejemplo

Figura 9.12. *(continuación).*

9.35. La función representada por el bloque 8 de la Figura 9.12*b* podría producirse por ¿qué instrucción del 8080/8085?

Res. ADD *B*.

9.36. La función representada por el bloque 9 de la Figura 9.12*b* podría producirse por ¿qué instrucción del 8080/8085?

Res. DAA.

9.37. La función representada por el bloque 11 de la Figura 9.12*b* podría producirse por ¿qué instrucción del 8080/8085?

Res. MOV *M,A*.

9.38. La función representada por el bloque 12 de la Figura 9.12*b* podría producirse por ¿qué instrucción del 8080/8085?

Res. HLT.

9.39. Escribir un programa en lenguaje ensamblador 8080/8085 para el problema representado en la Figura 9.12.

Res.

```
Etiqueta   Código de op   Operando    Comentarios
           LXI           H,2030H     ; Inicializa registro HL
           MOV           A,M         ; Carga A con primer número ASCII
           ANI           0FH         ; Enmascara 4 bits más significativos de A
           MOV           B,A         ; Guarda número BCD en B
           INX           H           ; Incrementa registro HL a 2031H
           MOV           A,M         ; Carga A con segundo número ASCII
           ANI           0FH         ; Enmascara 4 bits más significativos de A
           ADD           B           ; Suma número BCD en A y B
           DAA                       ; Ajusta suma en A para que sea igual al número BCD
           INX           H           ; Incrementa registro HL a 2032H
           MOV           M,A         ; Almacena suma BCD en posición de memoria 2032H
           HLT                       ; Detiene MPU
```

9.40. Después que el programa de la Figura 9.12 ha sido ejecutado, la suma BCD _____ (BCD) será depositada en la posición _____ de la memoria de datos (hex).

Res. $0001\ 0100_{BCD}$, 2032H.

Capítulo 10

EL MICROPROCESADOR MOTOROLA 6800

10.1. INTRODUCCION

Motorola Corporation introdujo su microprocesador 6800 de 8 bits en la década de los años setenta. Motorola ha desarrollado desde entonces «la familia 6800» con nuevas pastillas que generalmente son una extensión software y hardware compatible con la MPU 6800. El 6800, 6802 (6800 con reloj interno y alguna RAM) y el más potente 6809 son microprocesadores de 8 bits que son fáciles de utilizar. Versiones especializadas de la familia 6800, como las 6801, 6804, 68HC04, 6805, 68HC05 y 68HC11 se describen como *computadoras de una sola pastilla.* El fabricante las denomina *unidades microcomputadoras* (MCU). Una pastilla MCU, tal como la 68HC11, puede incluir una unidad microprocesadora de 8 bits (como el 6800), un temporizador, ROM, RAM, un conversor A/D, un oscilador, líneas de E/S programables e incluso una EPROM. El 68HC11 podría compartir el repertorio de instrucciones del 6800. MCU en pastillas de 8 bits son populares en los sistemas computadoras dedicadas de bajo costo.

En 1980, Motorola introdujo su microprocesador avanzado de 16 bits: el 68000. El 68000 fue el primer microprocesador que introdujo una arquitectura interna de 32 bits y grandes conjuntos de registros de propósito general. Otros miembros de la familia 68000 son el 68HC000 (CMOS), 68008, 68010, 68020 y 68030. Las características de la MPU 68000 son una arquitectura interna de 32 bits con un bus externo de datos de 16 bits, y 16 megabytes (M bytes) de direcciones de memoria. La MPU 68008 tiene la misma arquitectura interna de 32 bits con un bus de datos externo de 8 bits. El 68010 es una MPU de memoria virtual mejorada de 16 bits externos/32 bits internos. El muy popular 68020 es una MPU de memoria virtual de 32 bits internos/32 bits externos. Más recientemente, Motorola ha sacado su más potente microprocesador de 32 bits mejorado: el 68030. Las MPU de la serie 68000 *no son compatibles* con la familia más antigua de microprocesadores: la 6800 de 8 bits.

La memoria virtual es una característica de MPU avanzadas, tales como la 68010, 68020 y 68030. El término *memoria virtual* significa que un programador puede escribir programas como si la capacidad de memoria fuese ilimitada. La técnica de memoria virtual permite programas mayores que la capacidad de la memoria principal. La parte «extra» del programa se almacena en memoria secundaria (como por ejemplo discos), pero es accedida como si estuviese en memoria principal.

El microprocesador 6800 se usa normalmente en un sistema microcomputador. Una configuración mínima de sistema se ilustra en la Figura 10.1. El microprocesador 6800 requiere una fuente de alimentación de +5 V y un reloj externo de dos fases (no se necesita reloj externo en la MPU 6802). La entrada de rearranque a la MPU 6800 mostrada en la Figura 10.1 es una interrupción hardware conectada a la patilla de REINICIALIZACION de la pastilla y se utiliza para

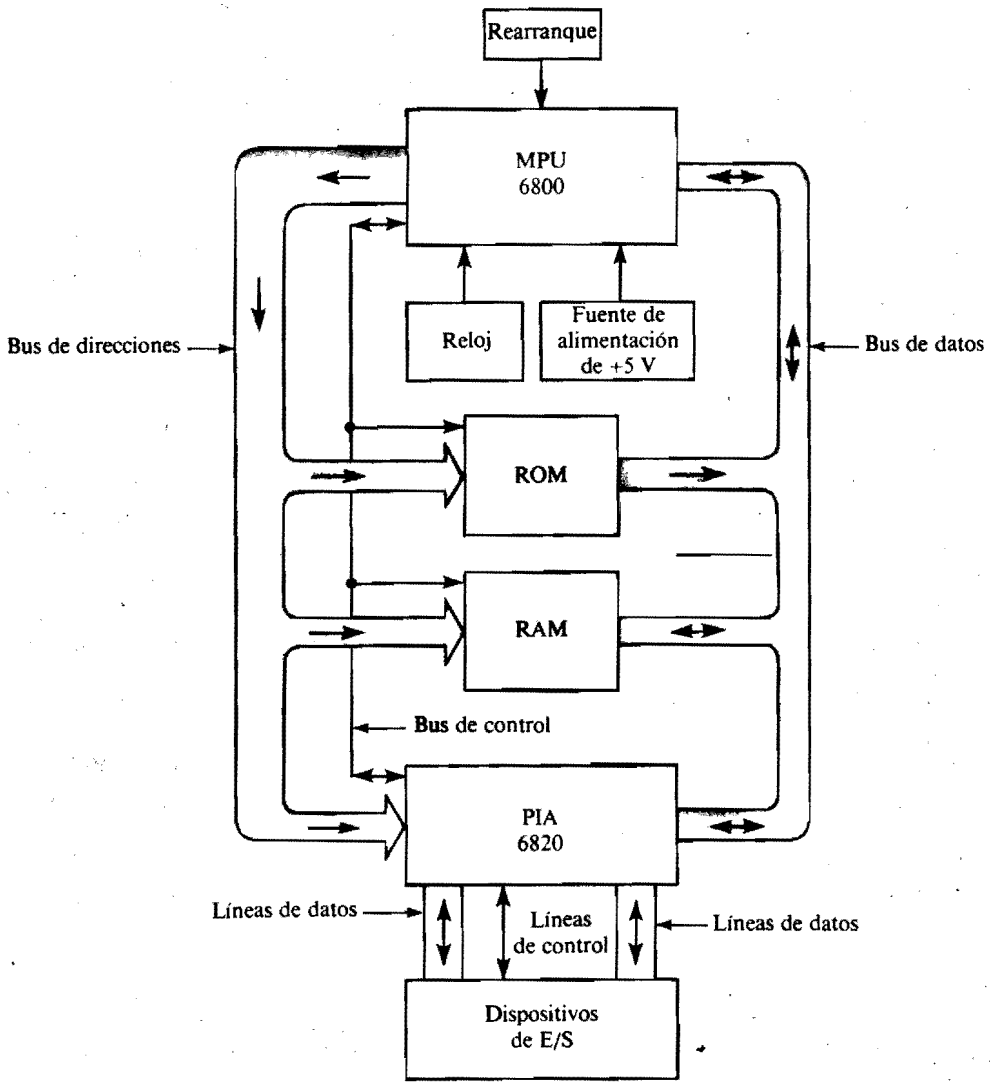


Figura 10.1. Diagrama de bloques de un sistema mínimo basado en el 6800.

inicializar el sistema. La MPU 6800 tiene 16 líneas de dirección y 8 líneas de datos bidireccionales. Las líneas de control están etiquetadas como el bus de control de la Figura 10.1. A la MPU en la Figura 10.1 se le ha añadido la memoria en forma de ROM y RAM.

La interfaz entre la MPU y los dispositivos de entrada/salida en la Figura 10.1 se hace con el *adaptador de interfaz de periféricos 6820* (PIA) especial de Motorola. Observar que la PIA 6820 proporciona dos caminos de datos de 8 bits que pueden ser utilizados como puertos de entrada o de salida. La PIA tiene también 4 líneas de control entre la PIA y los dispositivos periféricos. La PIA 6820 tiene registros internos que la pueden hacer programable y permite almacenamiento

temporal de datos para simplificar la transferencia de datos. Observar, en la Figura 10.1, que los pulsos de interrupción pueden ser enviados desde la PIA, vía el bus de control, a la MPU 6800. En resumen, la PIA 6820 forma un enlace de comunicación de dos direcciones y un dispositivo de almacenamiento temporal entre el microprocesador y los dispositivos periféricos.

Motorola también produce otros dispositivos especiales compatibles con el microprocesador 6800. Algunos son el *adaptador de interfaz de comunicaciones asíncronas 6850* (ACIA) que se utiliza para transmisión asíncrona de datos serie, las *pastillas de reloj de la serie 6870 de Motorola* que pueden ser utilizadas para proporcionar los pulsos de reloj de dos fases que requiere la MPU, y el *controlador de acceso de memoria directa 6844* que proporciona, a los sistemas basados en el 6800, la lógica para soportar cuatro canales de acceso directo a memoria (DMA). Otras muchas pastillas de soporte están disponibles para las series de microprocesadores 6800 y 68000.

10.2. FUNCIONES Y DIAGRAMA DE PATILLAS DEL 6800

El microprocesador de 8 bits Motorola 6800 está ubicado en un encapsulamiento de doble línea de 40 patillas (DIP). El diagrama de patillas de la MPU 6800 se ilustra en la Figura 10.2*a*. En la Figura 10.2*b* hay una tabla que resume los nombres y descripciones de las patillas. La patilla estándar MPU MC6800 opera con una frecuencia de reloj de 1,0 MHz. Existen versiones más rápidas de la MPU 6800. El MC68A00 es el mismo procesador pero opera con una frecuencia de reloj de 1,5 MHz, mientras que el MC68B00 opera con 2,0 MHz.

Las entradas y salidas de la MPU 6800 pueden agruparse como entradas de reloj, entradas de alimentación, entradas/salidas del bus de datos, salidas del bus de direcciones y entradas/salidas de control. Las flechas en el diagrama de patillas de la Figura 10.2*a* muestran la dirección del flujo de señales.

La función de cada patilla del microprocesador 6800 está descrita brevemente en los párrafos que siguen. En cada párrafo, acudir a la Figura 10.2 para la localización de cada patilla y un resumen de sus funciones.

Tierra (V_{ss}) (patillas 1 y 21)

Estas patillas están conectadas a la parte negativa de la fuente de alimentación de 5 V d.c.

Alto ($\overline{HALT}$) (patilla 2)

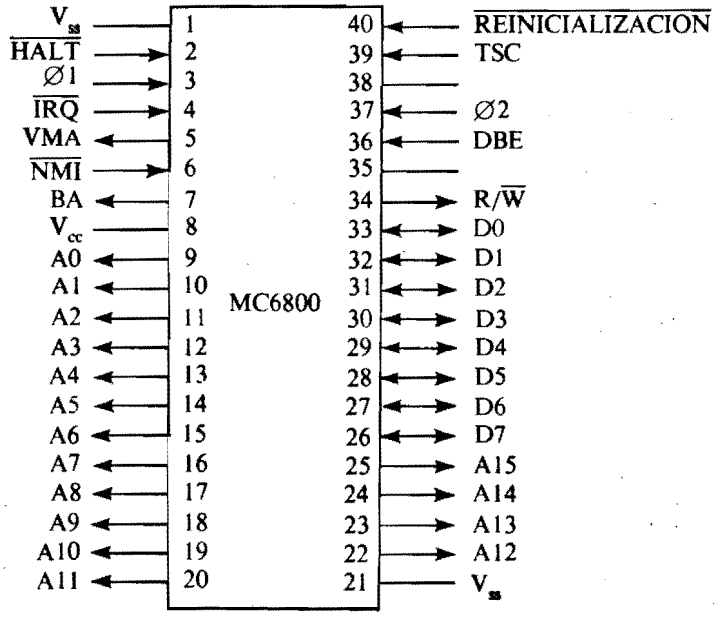
Cuando la entrada $\overline{HALT}$ está activada por un nivel BAJO, la MPU 6800 entra en el *modo alto*. En este modo las líneas de tres estados van a su estado de alta impedancia y el procesador se detiene. Esta entrada puede utilizarse para «ir paso a paso» en el proceso de depuración de un programa. Muchas veces la entrada $\overline{HALT}$ se inhabilita al conectarla a +5 V.

Reloj ($\varnothing 1$ y $\varnothing 2$) (patillas 3 y 37)

Estas patillas de entrada reciben señales de reloj no solapadas de la circuitería de reloj externa al sistema.

Línea de petición de interrupción ($\overline{IRQ}$) (patilla 4)

Cuando la entrada $\overline{IRQ}$ es activada con un nivel BAJO, el microprocesador 6800 completará la instrucción actual, introducirá todos los registros en la pila, pondrá a 1 el señalizador I, y



(a) Diagrama de patillas de la MPU 6800

Nombres de las patillas	Descripción	Tipo
*A0–A15	Líneas de dirección	Salida, tres estados
*D0–D7	Líneas del bus de datos	Bidireccional, tres estados
*HALT	Alto	Entrada
*TSC	Control de tres estados	Entrada
*R/W	Lectura/escritura	Salida, tres estados
*VMA	Dirección válida de memoria	Salida
*DBE	Habilitación bus de datos	Entrada
*BA	Bus disponible	Salida
*IRQ	Petición de interrupción	Entrada
REINICIALIZACION	Reinicialización	Entrada
NMI	Interrupción no enmascarable	Entrada
Ø1, Ø2	Señales de reloj	Entrada
V_{ss}, V_{cc}	Alimentación	

* Estas señales se conectan al bus del sistema.

(b) Nombres de las patillas y funciones

Figura 10.2.

saltará a la rutina de servicio de interrupción señalada por las posiciones de memoria FFF8H y FFF9H. La interrupción $\overline{IRQ}$ puede ser enmascarada.

Dirección válida de memoria (VMA) (patilla 5)

Cuando la salida *VMA* va al nivel ALTO, indica a otros dispositivos del sistema que la dirección en el bus de direcciones es válida.

Interrupción no enmascarable ($\overline{NMI}$) (patilla 6)

Cuando la entrada $\overline{NMI}$ va al nivel BAJO, la respuesta del microprocesador 6800 es similar a su respuesta cuando la entrada $\overline{IRQ}$ alcanza el nivel BAJO. La interrupción $\overline{NMI}$ es no enmascarable y su vector de direcciones se encuentra en las posiciones de memoria FFFCH y FFFDH. Los contenidos de FFFCH y FFFDH apuntan entonces al comienzo de la rutina de servicio de interrupción.

Bus disponible (BA) (patilla 7)

Cuando la salida *BA* alcanza el nivel ALTO indica a los dispositivos externos que el microprocesador 6800 ha dejado de ejecutar instrucciones. La MPU 6800 cesa de ejecutar instrucciones debido a una instrucción de espera (WAI) o a una entrada hardware $\overline{HALT}$. La línea de control *BA* proporciona un método de decir a los dispositivos externos que los buses de direcciones y datos de la MPU 6800 están en el estado de tres estados y pueden ser utilizados para cosas como, por ejemplo, operaciones DMA.

Alimentación (V_{cc}) (patilla 8)

La parte positiva de la fuente de alimentación de 5 V d.c. se conecta a la patilla 8 del CI microprocesador 6800.

Líneas de dirección (A_0–A_{15}) (patillas 9-20, 22-25)

Las 16 salidas de dirección de la MPU 6800 (A_0–A_{15}) están conectadas al bus de direcciones del sistema. Estas líneas son utilizadas para direccionar dispositivos como ROM, RAM y de E/S.

Líneas de datos (D_0–D_7) (patillas 26-33)

Las ocho patillas de datos (D_0–D_7) de la MPU 6800 están conectadas al bus de datos bidireccional del sistema. Las líneas del bus de datos están para transferir datos a y desde memoria o dispositivos de E/S.

Control de lectura/escritura ($R/\overline{W}$) (patilla 34)

La salida $R/\overline{W}$ es una línea de control que indica a un dispositivo externo (RAM, ROM, PIA, etcétera) si la MPU 6800 está leyendo o escribiendo. Una salida ALTA de la línea $R/\overline{W}$ indica una lectura, mientras que una salida BAJA indica que se está realizando una operación de escritura.

Habilitación del bus de datos (DBE) (patilla 36)

La entrada *DBE* controla las excitaciones del bus de datos del microprocesador 6800. Una entrada ALTA en la patilla *DBE* habilita las líneas de datos, mientras que una entrada BAJA inhabilita, o pone en tres estados, a las líneas de datos. Debido a que las transferencias de datos entre memoria y E/S ocurren cuando el reloj $\varnothing 2$ está en el nivel ALTO, la patilla *DBE* está muchas veces conectada a la línea $\varnothing 2$ del reloj.

Control de tres estados (TSC) (patilla 39)

Cuando la patilla de entrada *TSC* está excitada en ALTO por un dispositivo externo, todas las líneas de dirección y la línea de $E/\overline{W}$ estarán en tres estados. La línea *TSC* se utiliza con configuraciones DMA. Cuando no se utiliza DMA la patilla *TSC* puede conectarse a tierra.

Línea de reinicialización ($\overline{REINICIALIZACION}$) (patilla 40)

Cuando la patilla de $\overline{REINICIALIZACION}$ pasa del nivel ALTO al BAJO, el microprocesador 6800 pone el señalizador *I* a 1 y salta a la rutina de servicio de interrupciones de reinicialización señalada por FFFEH y FFFFH. La entrada de $\overline{REINICIALIZACION}$ se utiliza típicamente para inicializar o reinicializar el sistema.

PROBLEMAS RESUELTOS

10.1. El Motorola 6800 es un microprocesador de _____ (8, 16) bits.

Solución:

El Motorola 6800 es un microprocesador de 8 bits. Su acumulador es de 8 bits, y transfiere datos en grupos de 8 bits vía el bus de datos.

10.2. Una sencilla fuente de alimentación de _____ −V se utiliza para alimentar el microprocesador 6800. En la Figura 10.2*a*, la parte positiva de la fuente de alimentación está conectada a la patilla _____, mientras que la negativa va a las patillas etiquetadas _____.

Solución:

Se utiliza una fuente de alimentación de 5 V para alimentar el microprocesador 6800. En la Figura 10.2*a*, la parte positiva de la fuente de alimentación está conectada a la patillas etiquetadas V_{ss} (patillas 1 y 21).

10.3. El microprocesador 6800 utiliza _____ (número) patillas como líneas de dirección y _____ (número) patillas como líneas de datos bidireccionales.

Solución:

El microprocesador 6800 utiliza 16 patillas como líneas de dirección y 8 patillas como líneas de datos bidireccionales.

10.4. Las tres interrupciones hardware del microprocesador 6800 son las de $\overline{REINICIALIZACION}$, _____ y _____.

Solución:

Las tres interrupciones hardware del microprocesador 6800 son las de $\overline{REINICIALIZACION}$, $\overline{NMI}$ (interrupción no enmascarable) e $\overline{IRQ}$ (petición de interrupción).

10.5. Activar la patilla de entrada _____ en el 6800 hará que el procesador se detenga y todas las líneas de tres estados vayan a su estado de alta impedancia.

Solución:

Al activar la patilla de entrada $\overline{HALT}$ en el 6800 hará que el procesador se detenga y todas las líneas de tres estados vayan a su estado de alta impedancia.

10.6. La MPU 6800 utiliza un reloj de _____ -fases sin solapamiento.

Solución:

La MPU 6800 utiliza un reloj de dos fases sin solapamiento.

10.7. Acudir a la Figura 10.2*a*. La patilla 34 en la MPU 6800 está etiquetada _____, que significa _____ / _____. Cuando esta salida de control está en ALTA, el procesador está _____ (leyendo, escribiendo), y cuando está en BAJA, está _____ (leyendo, escribiendo).

Solución:

Según la información de la Figura 10.2 la patilla 34 de la MPU 6800 está etiquetada $R/\overline{W}$, que significa lectura/escritura. Cuando esta salida de control está en el nivel ALTO, el procesador está leyendo, y cuando está en el nivel BAJO, está escribiendo.

10.8. Acudir a la Figura 10.2*a*. La patilla 5 de la MPU 6800 está etiquetada _____, lo cual significa _____ _____ _____.

Solución:

Según la información de la Figura 10.2, la patilla 5 de la MPU 6800 está etiquetada *VMA*, lo cual significa dirección válida de memoria.

10.3. ARQUITECTURA DEL 6800

Un diagrama de bloques de la organización interna del microprocesador Motorola 6800 se muestra en la Figura 10.3. El 6800 tiene un bus de datos, con buffers de tres estados, que contiene 8 bits (D_0–D_7). Los números próximos a cada entrada o salida de la Figura 10.3 son los números de patilla. En la parte superior de la Figura 10.3 está el bus de direcciones con buffers de 16 bits (A_{15}–A_0). La MPU 6800 contiene un contador de programa de 16 bits, un puntero de pila de 16 bits, un registro índice de 16 bits, dos acumuladores de 8 bits (A y B) y un registro de códigos de condición de 8 bits. Las nueve señales de control se muestran a la izquierda de la Figura 10.3. Son las entradas de reinicialización, de interrupción no enmascarable, de continúa/alto, de petición de interrupción, control de tres estados y habilitación del bus de datos y las salidas de bus disponible, dirección de memoria válida y lectura/escritura. Las entradas de reloj $\varnothing 1$ y $\varnothing 2$ también se muestran a la izquierda en la Figura 10.3.

La sección de la unidad aritmética-lógica (ALU) realiza las operaciones lógicas y aritméticas, mientras que el registro de códigos de condición asociado contiene los seis señalizadores. El registro de instrucción alimenta al decodificador de instrucciones y al bloque de control. El decodificador de instrucciones interpreta la instrucción y determina el procedimiento que se va a seguir para su ejecución. El decodificador de instrucciones dirige al control para generar las señales internas y externas, adecuadas, para completar la instrucción. Las líneas de control interno que coordinan la operación del microprocesador no se muestran en la Figura 10.3.

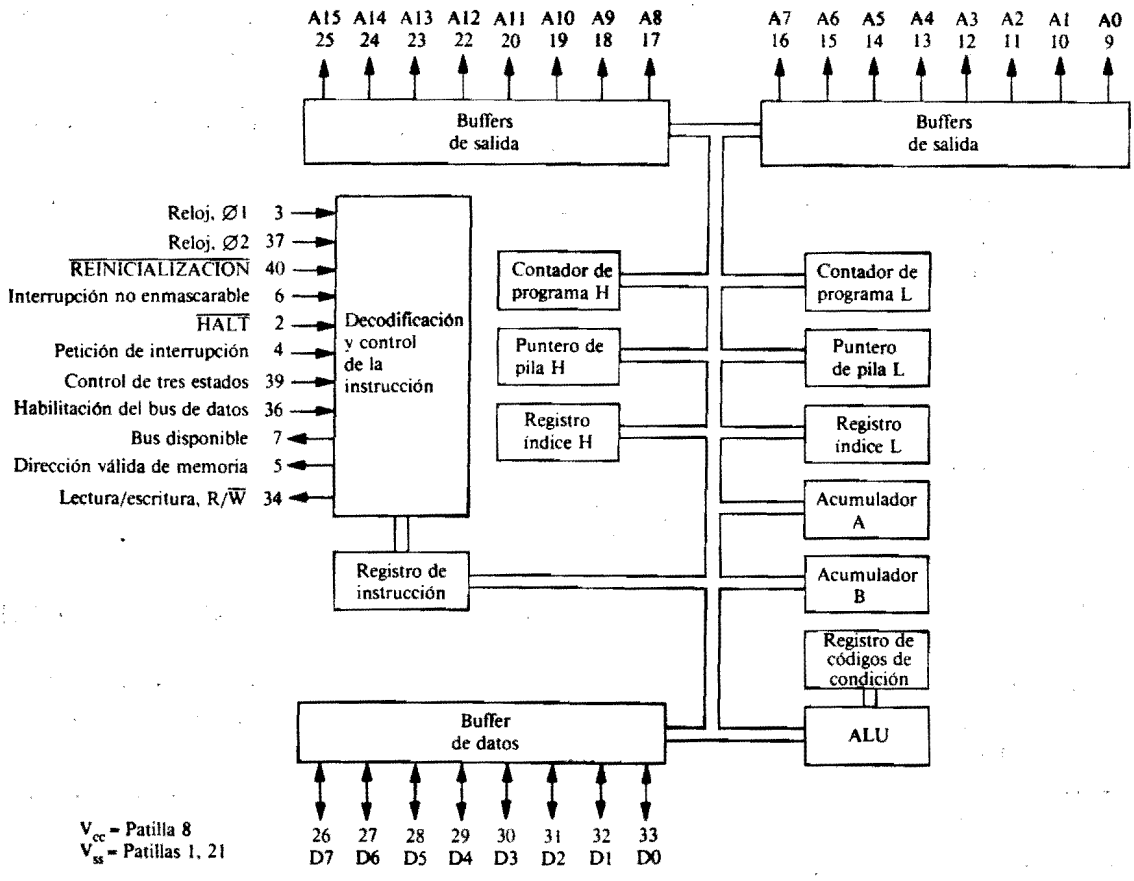


Figura 10.3. Diagrama de bloque funcional de la MPU 6800. *(Cortesía de Motorola, Inc.)*

Registros

El diagrama de bloques de la MPU 6800 de la Figura 10.3 hace que esta unidad parezca más simple que la 8085. Esto se debe principalmente al uso de solamente dos acumuladores. La MPU 6800 se considera a veces como un procesador orientado a memoria porque normalmente hace muchas transferencias con memoria. El 6800 también utiliza E/S de mapa de memoria. Las entradas y salidas son tratadas como posiciones de memoria.

Los registros del microprocesador 6800 son los siguientes:

1. Los dos *acumuladores* de 8 bits (*ACCA* y *ACCB*) son el foco de muchas de las operaciones del microprocesador. Las operaciones del acumulador incluyen las aritméticas, lógicas, de carga y almacenamiento, entrada y salida y otras. El segundo acumulador (*ACCB*) no es exactamente un registro de propósito general, sino un acumulador real, ya que todas las operaciones que pueden realizarse en el acumulador *A* también pueden realizarse en el acumulador *B*.

2. El *contador de programa (PC)* es un registro de 16 bits que siempre contiene la dirección de la siguiente instrucción que se va a ejecutar. Tiene el mismo propósito que el contador de programa de los microprocesadores genéricos 8080 y 8085.
3. El *puntero de pila (SP)* es un registro de propósito especial de 16 bits. El puntero de pila contiene una dirección de 16 bits que define la parte superior de la pila en la RAM. Las MPU del 8080/8085 y genérica también contienen punteros de pila.
4. El *registro índice (IX)* es un registro de propósito especial de 16 bits. Su uso principal es para apuntar y modificar direcciones.
5. El *registro de códigos de condición (CCR)* es un registro de 8 bits que contiene los seis señalizadores utilizados por la MPU 6800.

Señalizadores

La MPU 6800 utiliza seis bits de código de condición o señalizadores. Estos seis señalizadores están representados en la Figura 10.4, y se agrupan en un solo registro de 8 bits que Motorola denomina registro de códigos de condición (*CCR*). Como en otros microprocesadores, las instrucciones de bifurcación examinan los señalizadores para determinar si existe o no una condición de bifurcación.

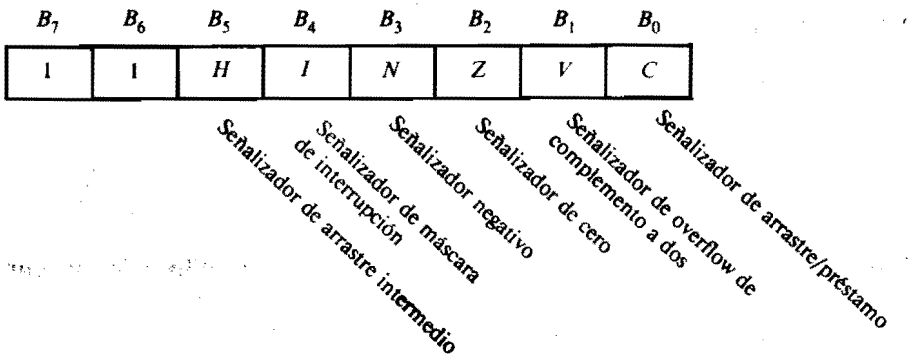
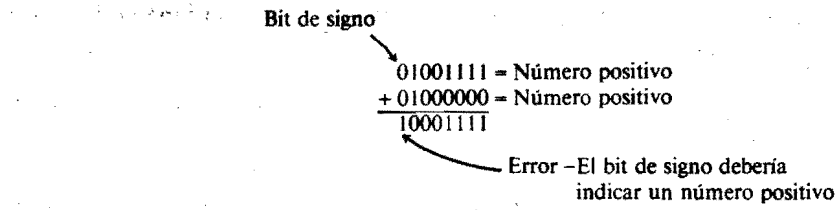


Figura 10.4. Señalizadores del registro de código de condición del 6800.

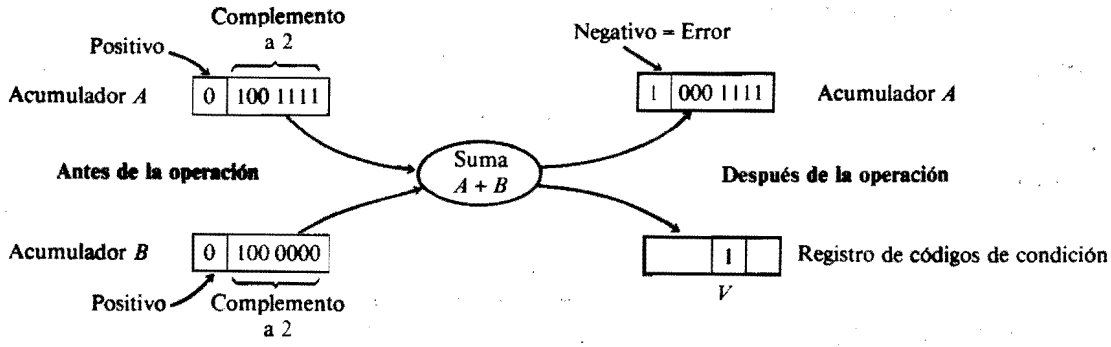
Como en el microprocesador genérico, y en los microprocesadores 8080 y 8085, el *señalizador de arrastre (C)* se pone a 1 siempre que se genera un arrastre (o «préstamo») en el MSB del acumulador. Una suma mayor que la capacidad del acumulador de 8 bits pone el señalizador *C* a 1.

El *señalizador de «overflow» (V)* del registro de códigos de condición, de la MPU del 6800, indica un overflow de complemento a 2. Cuando se trata con números con signo en el 6800, el MSB (B_7) del (de los) acumulador(es) es el bit de signo. Los 7 bits restantes se escriben en forma de complemento a 2 y contendrán números comprendidos entre $+127_{10}$ y -128_{10}. Este es el rango de los números con signo. Si el resultado de una operación aritmética excede este rango, el señalizador de overflow *(V)* se pone a 1. Esto avisa al usuario que se ha producido un error.

Considerar la suma de los números positivos 79_{10} y 64_{10}. El decimal +79 es igual a 01001111 en la notación en complemento a 2. El decimal +64 es igual a 01000000 en complemento a 2. Estos números en complemento a 2 son sumados en la Figura 10.5*a*. Debido al arrastre que se produce entre las posiciones B_6 y B_7, el bit de signo del resultado ha cambiado a 1, indicando un número negativo, lo cual es un error. El procesador 6800 en la Figura 10.5*b* se utiliza para sumar



(a) Suma en complemento a 2 mostrando el efecto sobre el bit del signo



(b) Efecto en el señalizador de overflow

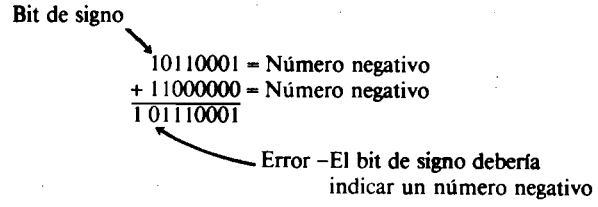
Figura 10.5.

los mismos números positivos (en la notación en complemento a 2) utilizando los acumuladores *A* y *B*. La suma (10001111 en este ejemplo) se deposita en el acumulador *A* después de la operación de suma. El señalizador de overflow *(V)* se pone a 1, indicando que la suma es mayor que $+127_{10}$ (suma = $79_{10} + 64_{10} = 143_{10}$ en este ejemplo).

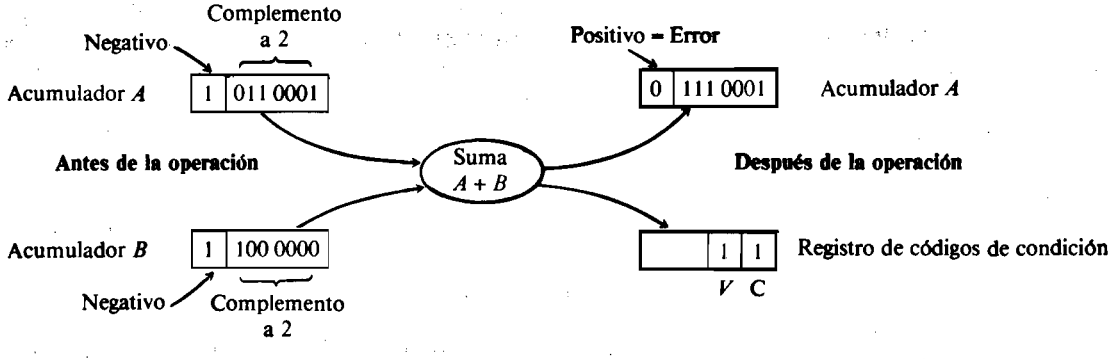
Considerar la suma de dos números negativos como -79_{10} y -64_{10}. El decimal −79 es igual a 10110001 en la notación en complemento a 2 y el −64 es igual a 11000000. Como la mayoría de los bits significativos de ambos números en complemento a 2 son 1, representan números negativos entre −1 y −128. Estos números en complemento a 2 se suman en la Figura 10.6*a*. El resultado es 1 0111 0001. El bit de signo debe permanecer a 1 (para los negativos), pero ha cambiado. El resultado es erróneo porque excede el límite del rango de -128_{10}.

La tarea de sumar los números negativos -79_{10} (10110001 en complemento a 2) y -64_{10} (11000000 en complemento a 2) utilizando la MPU 6800 se muestra en la Figura 10.6*b*. Los números en complemento a 2 están en los acumuladores *A* y *B*. La suma aparece en el acumulador *A* después de la operación de sumar. La suma es errónea a causa de un problema de overflow. El señalizador de overflow *(V)* se pone a 1, avisando al usuario que se ha excedido el rango del procesador 6800. El señalizador de arrastre *(C)* también se pone 1, indicando el arrastre fuera de la posición B_7. Esto se muestra en la Figura 10.6*a*.

El *señalizador de cero (Z)* del registro de códigos de condición de la MPU 6800 se pone a 1 siempre que el contenido del acumulador sea cero como resultado de una operación o de transferencia de datos. Si el señalizador de cero se pone a 0, significa que el contenido del acumulador *no es* cero.



(a) Suma en complemento a 2 mostrando el efecto sobre el bit del signo



(b) Efecto en el señalizador de overflow

Figura 10.6.

El *señalizador negativo (N)* del registro de códigos de condición de la MPU 6800 se utiliza para indicar un resultado negativo. Tiene el mismo propósito que el señalizador de signo en el procesador 8080/8085. El bit de signo es B_7 en el acumulador. Si el resultado de la última operación aritmética, lógica o de transferencia de datos fuese negativo, el señalizador N se pondrá a 1. Sin embargo, si el resultado fuese positivo, el señalizador se pondrá a 0. El señalizador N estará igual que el MSB del acumulador.

El *señalizador de máscara de interrupción (I)* del registro de códigos de condición en la MPU 6800 inhibe todas las interrupciones $\overline{IRQ}$ cuando se pone a 1. Si el señalizador I está a 0, en la patilla de entrada $\overline{IRQ}$ de la pastilla del procesador 6800 habrá un nivel BAJO. Algunas instrucciones especiales afectan al señalizador de máscara de interrupción. Estas instrucciones se detallarán en una sección posterior.

El *señalizador de semiarrastre (H)* en el registro de códigos de condición de la MPU 6800 indica un arrastre de B_3 a B_4, en el acumulador, durante una operación de suma. El señalizador H se pone a 1 si hay un arrastre de B_3 a B_4 y a 0 en el caso contrario.

Los dos bits más significativos del registro de códigos de condición mostrado en la Figura 10.4 no se utilizan como señalizadores. Siempre se ponen a 1 en el microprocesador 6800.

Unidad aritmético-lógica

La *unidad aritmético-lógica* (ALU) debe ser considerada el corazón del microprocesador 6800. Realiza todas las operaciones aritméticas y lógicas. Los resultados de estas operaciones son

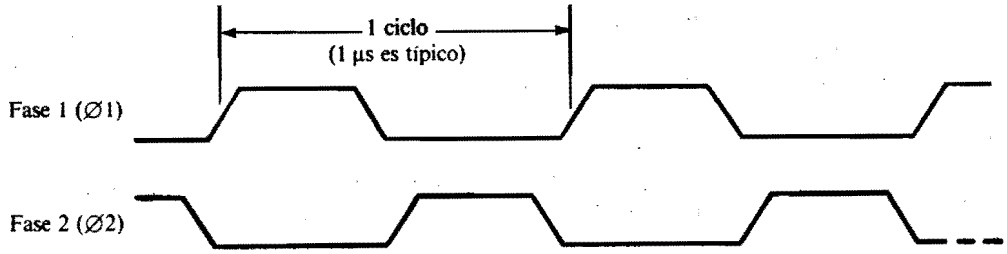
almacenados en uno de los acumuladores. La ALU realiza operaciones como sumar, restar, comparar, AND, OR y XOR.

Decodificador de instrucciones

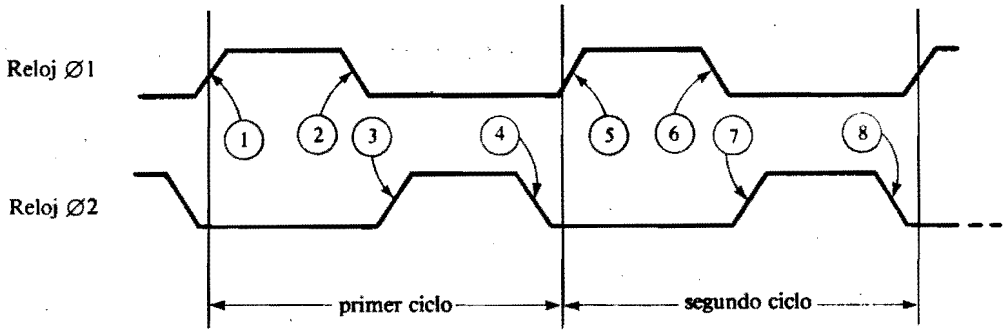
El *decodificador de instrucciones* representado en la Figura 10.3 interpreta las entradas del registro de instrucciones. Basándose en estas entradas, el decodificador de instrucciones dirige a la sección de control para que ejecute la instrucción actual.

Reloj del sistema

El microprocesador 6800 requiere un reloj de dos fases sin solapamiento. La circuitería del reloj está localizada fuera de la pastilla 6800 y alimenta las patillas de entrada Ø1 y Ø2 mostradas en la Figura 10.3. El reloj sincroniza las operaciones internas del microprocesador. También sincroniza los dispositivos externos. Las formas de onda se muestran en la Figura 10.7. En la Figura 10.7*a* se identifica un *ciclo de reloj*. La duración del ciclo puede variar de 1 a 10 µs para el microprocesador estándar 6800. El ciclo de reloj no puede ser ralentizado a menos de 10 µs por ciclo. Si opera muy lentamente, algunos de los registros internos dinámicos pueden perder sus datos. Observar en la Figura 10.7*a* que cuando la fase 1 (Ø1) está en ALTA, la fase 2 (Ø2) está siempre en BAJA. Lo opuesto también es cierto.



(a) Formas de onda de dos fases de reloj sin solapamiento



(b) Temporización de la instrucción «cargar inmediato acumulador *A*»

Figura 10.7.

Considerar el diagrama de tiempos de una instrucción cuando se utiliza la MPU 6800. Para ejecutar una instrucción, cualquier microprocesador realiza tres pasos: (1) búsqueda del código de op, (2) decodificación y (3) ejecución de la instrucción. El ejemplo esbozado en la Figura 10.7*b* es para la instrucción «cargar el acumulador *A* inmediato». Durante el primer ciclo mostrado en la Figura 10.7*b*, la MPU 6800 realizará la búsqueda del código de op y la decodificación de la instrucción. Durante el segundo ciclo, se ejecutará la instrucción. Los números inscritos en los círculos de la Figura 10.7*b* permiten que se siga la secuencia de eventos cuando la MPU 6800 está realizando la operación *cargar el acumulador A inmediato.*

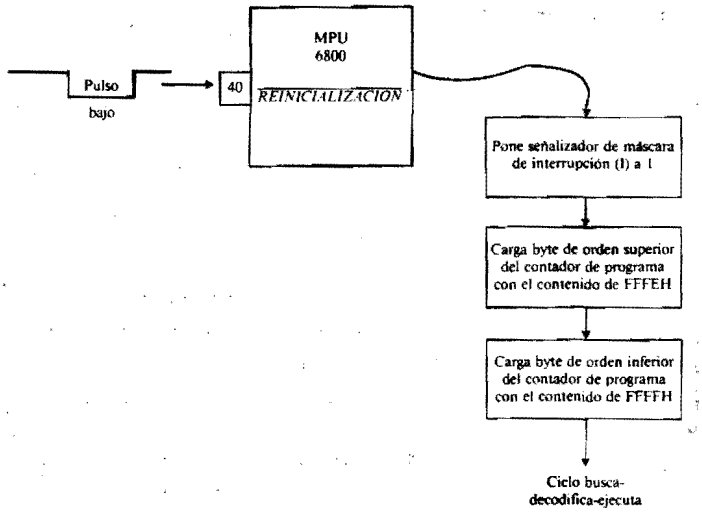
1. En la transición L a H (ALTA a BAJA) del reloj $\varnothing 1$, la dirección es transferida del contador de programa al bus de direcciones. Las patillas *VMA* y $R/\overline{W}$ del 6800 van al nivel ALTO. Una línea *VMA* en ALTA indica, a la memoria, que la dirección del bus de direcciones es válida. Una línea $R/\overline{W}$ ALTA señala a la memoria que el procesador está en un ciclo de lectura.
2. En la transición H a L del reloj $\varnothing 1$, se incrementará el contador de programa. Este no cambiará la dirección actualmente sobre el bus de direcciones porque está almacenada en el registro de direcciones de memoria.
3. En la transición L a H del reloj $\varnothing 2$, la MPU 6800 indica a la memoria que coloque el código de op para la instrucción de «cargar inmediato el acumulador *A*» en el bus de datos.
4. En la transición H a L del reloj $\varnothing 2$, la MPU acepta el código de op del bus de datos, lo transfiere al registro de instrucciones y decodifica la instrucción. Una vez decodificada, el procesador determina que durante el segundo ciclo se ejecute la instrucción de «cargar inmediato el acumulador *A*». Durante el primer ciclo de la MPU el código de operación fue leído de memoria (buscado) y decodificado.
5. En la transición L a H del reloj $\varnothing 1$, la dirección del operando es transferida desde el contador de programa al registro de direcciones de memoria y al bus de direcciones.
6. En la transición H a L del reloj $\varnothing 1$, se incrementa el contador de programa.
7. En la transición L a H del reloj $\varnothing 2$, el procesador indica a la memoria que coloque el operando (dato a ser cargado en el acumulador *A*) en el bus de datos.
8. En la transición H a L del reloj $\varnothing 2$, la MPU 6800 acepta el operando del bus de datos y lo transfiere al acumulador *A*.

Los ocho pasos anteriores se realizan normalmente en 2 μs cuando se utiliza un microprocesador estándar 6800. Observar que los eventos que ocurren durante el reloj $\varnothing 1$ están en la MPU 6800 mientras que el reloj $\varnothing 2$ está aplicado a dispositivos externos. El reloj $\varnothing 2$ se utiliza para controlar operaciones de memoria en este ejemplo.

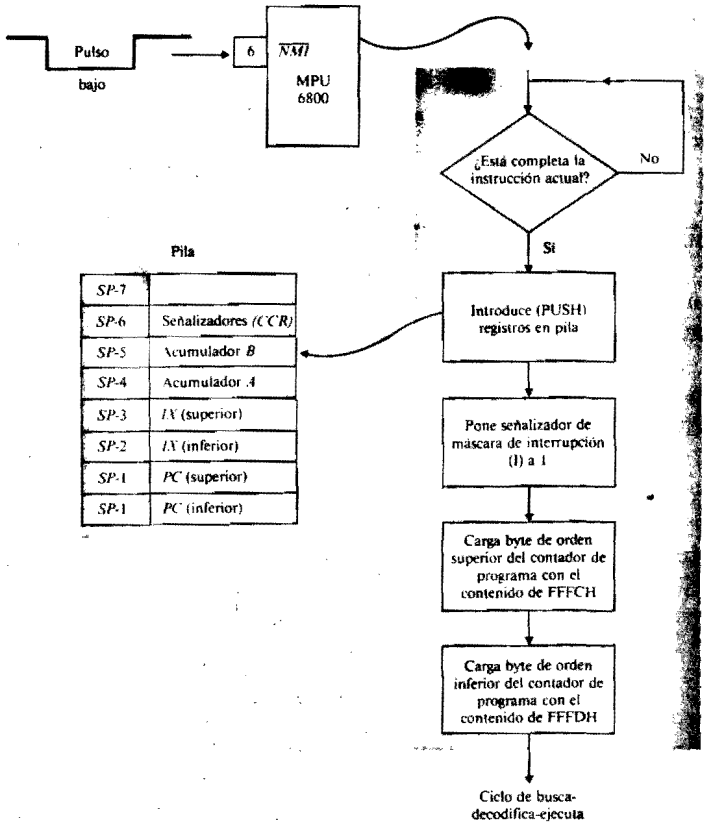
Interrupciones

El microprocesador 6800 tiene tres tipos de *interrupciones hardware.* Son las de reinicialización, interrupción no enmascarable y petición de interrupción. La MPU 6800 también tiene una única instrucción de interrupción *software* (SWI).

Cuando la patilla de interrupción $\overline{REINICIALIZACION}$ en la MPU del 6800 está en el nivel BAJO durante un tiempo y después va al nivel ALTO, el procesador salta automáticamente a una *secuencia especial de interrupción de reinicialización.* Esta secuencia se detalla en la Figura 10.8*a*. El señalizador de la máscara de interrupción *(I)* se pone primero a 1, y después el contador de

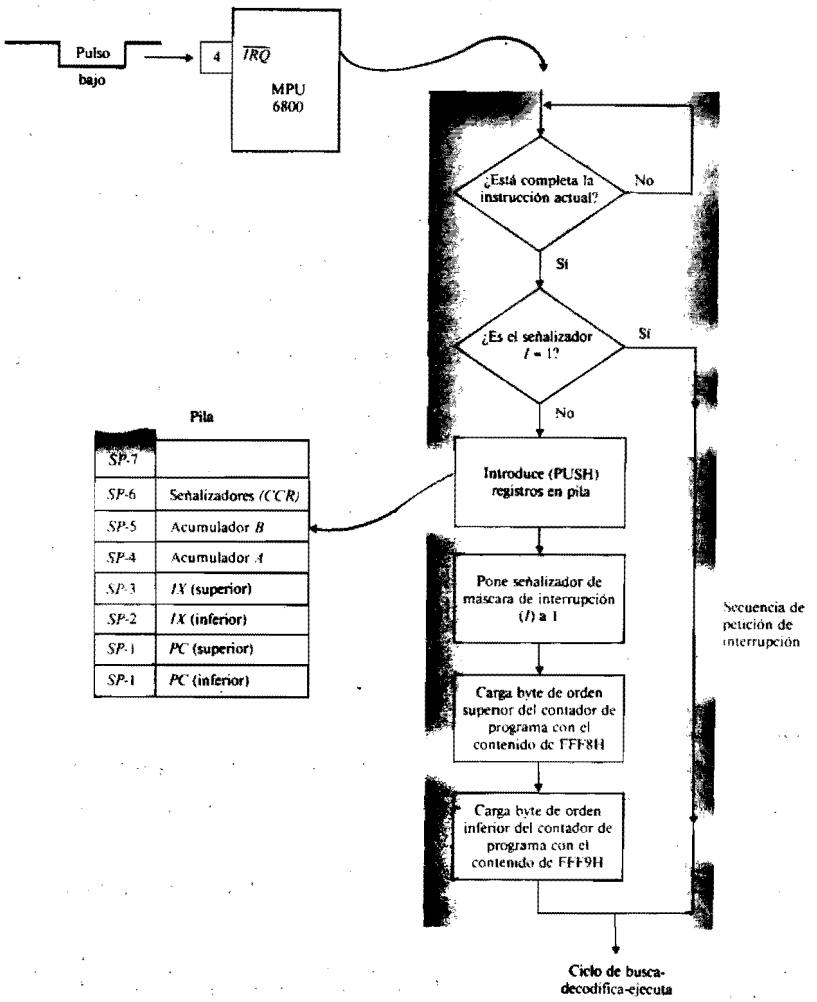


(a) Secuencia de la interrupción reinicialización



(b) Secuencia de interrupción no enmascarable

Figura 10.8.



(c) Secuencia de petición de interrupción

Figura 10.8. *(continuación).*

programa se carga con la dirección contenida en las posiciones de memoria FFFEH y FFFFH. Estas dos posiciones de memoria están reservadas para el *vector de reinicialización.* El vector de reinicialización no es nada más que una dirección del programa que dará servicio a la entrada de reinicialización. Es típico en la MPU 6800 que la ROM esté localizada en las direcciones superiores de memoria, mientras que la RAM habitualmente empieza en la dirección 0000H.

La entrada de *interrupción no enmascarable (NMI)* a la MPU 6800 está activada en la Figura 10.8*b*. Cuando la patilla NMI de la MPU 6800 va al nivel BAJO, el procesador salta a la *secuencia de interrupción no enmascarable* mostrada a la derecha de la Figura 10.8*b*. La instruc-

ción actual se completa, el contenido de los registros del 6800 se guarda en la pila, el señalizador *I* se pone a 1, y el vector de interrupción no enmascarable busca en las posiciones de memoria FFFCH y FFFDH. La MPU entonces saltará a la rutina de servicio de interrupción no enmascarable.

La entrada de *petición de interrupción* ($\overline{IRQ}$) a la MPU 6800 está activada en la Figura 10.8*c*. Cuando la patilla $\overline{IRQ}$ de la MPU 6800 va al nivel BAJO, el procesador salta a la *secuencia de petición de interrupción* detallada a la derecha de la Figura 10.8*c*. Se completa la instrucción actual y se comprueba el señalizador *I*. Si éste está a 1, el procesador ignora la petición de interrupción y continúa con el programa regular; la entrada de petición de interrupción es enmascarable y, por tanto, tiene una prioridad inferior que la de las entradas $\overline{REINICIALIZACION}$ o $\overline{NMI}$. Si el señalizador *I* hubiese estado a 0, el procesador introduciría, a continuación, el contenido de los registros en la pila en el orden mostrado en la Figura 10.8*c*. El siguiente paso en la secuencia de interrupción sería poner a 1 el señalizador *I*. A continuación el contador de programa se cargaría con el vector de petición de interrupción localizado en las posiciones de memoria FFF8H y FFF9H. El procesador saltaría entonces a la dirección de la rutina de servicio de petición de interrupción.

Las asignaciones de memoria del vector de interrupción en todos los sistemas basados en el 6800 se muestran en la Figura 10.9. Estas son posiciones reservadas de memoria cuyos contenidos son las direcciones donde comienzan las distintas rutinas de servicio de interrupción.

Las instrucciones SWI y asociadas del microprocesador 6800 se cubrirán en una sección posterior.

ROM	
Dirección	Contenido
FFF8	Byte superior
FFF9	Byte inferior
FFFA	Byte superior
FFFB	Byte inferior
FFFC	Byte superior
FFFD	Byte inferior
FFFE	Byte superior
FFFF	Byte inferior

Vector de petición de interrupción (dirección)

Vector de interrupción software (dirección)

Vector de interrupción no enmascarable (dirección)

Vector de reinicialización (dirección)

Figura 10.9. Posiciones de memoria del vector de interrupción en los sistemas basados en el 6800.

PROBLEMAS RESUELTOS

10.9. La MPU 6800 tiene acumulador(es) de _____ (número) 8 bits, un contador de programa de 16 bits, un puntero de pila de _____ (número) bits, un registro _____ de 16 bits, un registro de instrucción y un registro _____ que contiene seis señalizadores.

Solución:

De acuerdo con la Figura 10.3, la MPU 6800 tiene dos acumuladores de 8 bits, un contador de programa de 16 bits, un puntero de pila de 16 bits, un registro índice de 16 bits, un registro de instrucción y un registro de códigos de condición que contiene seis señalizadores.

10.10. La MPU 6800 tiene un bus de direcciones con buffers de tres estados de _____ (número) bits y un bus de _____ de buffers de tres estados de 8 bits.

Solución:

De acuerdo con la Figura 10.3, la MPU 6800 tiene un bus de direcciones de buffers de tres estados de 16 bits y un bus de datos con buffers de tres estados de 8 bits.

10.11. El registro de códigos de condición del 6800 contiene seis señalizadores. Listarlos.

Solución:

De acuerdo con la Figura 10.4, el registro de códigos de condición del 6800 contiene; señalizador de semiarrastre *(H)*, señalizador de máscara de interrupción *(I)*, señalizador negativo *(N)*, señalizador de cero *(Z)*, señalizador de «overflow» (rebose) de complemento a 2 *(V)* y señalizador de arrastre/préstamo *(C)*.

10.12. Acudir a la Figura 10.10. El contenido del acumulador *A* después de la operación de suma es _____ (8 bits).

Solución:

Sumar los números en complemento a 2 01000011 + 10000011 = 10000011. El contenido del acumulador *A* después de la operación de suma es 10000011.

10.13. Acudir a la Figura 10.10. El señalizador de overflow *(V)* está a _____ (0, 1) después de la operación de suma.

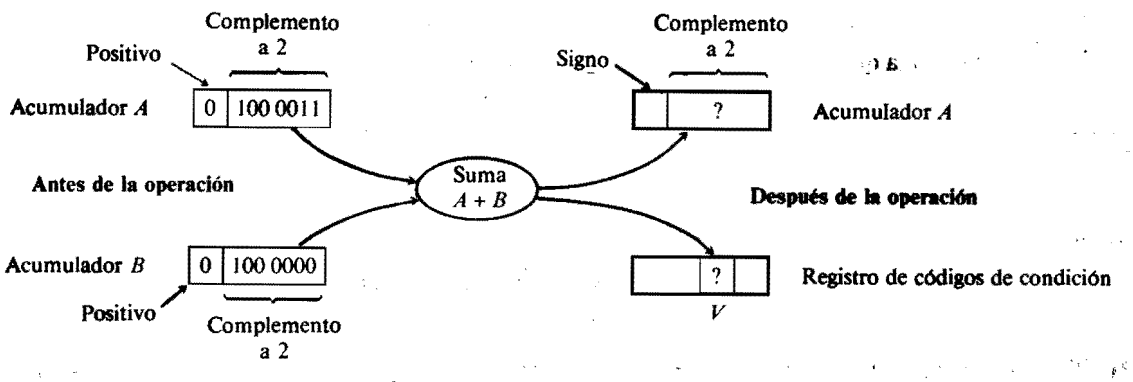


Figura 10.10.

Solución:

Al sumar los números en complemento a 2 01000011 + 01000000 = 10000011 (decimal +67 más +64 = +131). El señalizador *V* se pone a 1, indicando que un overflow en complemento a 2 ha provocado un error de rango. La «suma» del acumulador después de la operación de suma por tanto no es la suma correcta en complemento a 2.

10.14. Acudir a la Figura 10.11. El contenido del acumulador *A* después de la operación de resta es _____ (binario).

Solución:

Al restar en binario 00001111 − 00000001 = 00001110 (decimal 15 − 1 = 14). El contenido del acumulador *A* después de la operación de resta es 00001110_2.

10.15. Acudir a la Figura 10.11. El señalizador *Z* está a _____ (0, 1) después de la operación de resta.

Solución:

Al restar en binario 00001111 − 00000001 = 00001110. El señalizador *Z* está a 0 después de la operación de resta en la Figura 10.11 porque la diferencia no es cero.

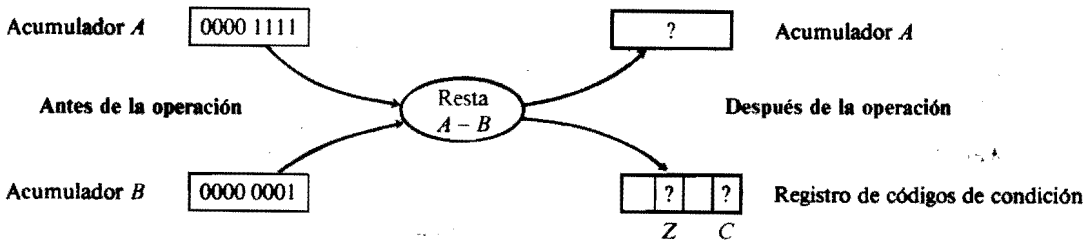


Figura 10.11.

10.16. Acudir a la Figura 10.11. El señalizador *C* está a _____ (0, 1) después de la operación de resta.

Solución:

Debido a que no existen préstamos, el señalizador *C* está a 0 después de la operación de resta de la Figura 10.11. El señalizador *C* de reinicialización indica que el número del acumulador *A* es mayor que el número del acumulador *B*.

10.17. Acudir a la Figura 10.12. El contenido del acumulador después de la operación de suma es _____ (8 bits).

Solución:

Sumar 10001001 + 10001001 = $1\ 0001\ 0010_2$. El contenido del acumulador después de la operación de suma de la Figura 10.12 es 00010010.

10.18. Acudir a la Figura 10.12. El señalizador *H* está a _____ (0, 1) después de la operación de suma.

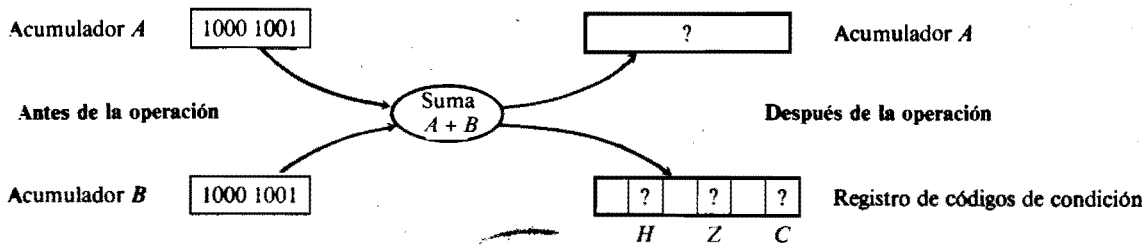


Figura 10.12.

Solución:

Al sumar 10001001 + 10001001 = 1 0001 0010. El señalizador H se pone a 1 después de la operación de suma de la Figura 10.12 porque había un arrastre de la posición B_3 a la B_4 del acumulador.

10.19. Acudir a la Figura 10.12. El señalizador Z está a _____ (0, 1) después de la operación de suma, mientras que el señalizador C está a _____ (0, 1).

Solución:

Sumar 10001001 + 10001001 = 1 0001 0010. El señalizador Z está a 0, mientras que el señalizador C está a 1 después de la operación de suma de la Figura 10.12.

10.20. Durante el ciclo de búsqueda del microprocesador, se recupera un código de op de memoria, vía el bus de datos y se deposita en el registro _____ del 6800. El código de op es entonces traducido por el _____ _____ del 6800, que indica a la sección de control que ejecute la instrucción.

Solución:

Acudir a la Figura 10.3. Durante el ciclo de búsqueda del microprocesador, se recupera un código de op de memoria, vía el bus de datos y se deposita en el registro de instrucciones del 6800. El código de operación es entonces traducido por el decodificador de instrucciones del 6800, que indica a la sección de control que ejecute la instrucción.

10.21. La MPU 6800 utiliza un reloj de _____ -fases _____ (sin solapamiento, con solapamiento). Generalmente, la fase 1 del reloj sincroniza las operaciones _____ (internas, externas) del procesador, mientras que los pulsos del reloj $\varnothing 2$ sincronizan los dispositivos externos y operaciones internas.

Solución:

Acudir a la Figura 10.7. La MPU 6800 utiliza un reloj sin solapamiento de dos fases. Generalmente, la fase 1 del reloj sincroniza las operaciones internas del procesador, mientras que los pulsos del reloj $\varnothing 2$ sincronizan las operaciones internas y los dispositivos externos.

10.22. Acudir a la Figura 10.13. El señalizador de la máscara de interrupción *(I)* estará a _____ (0, 1) después de la secuencia de la interrupción reinicialización mostrada en este ejemplo.

Solución:

Según la información de la Figura 10.8*a*, el señalizador de la máscara de interrupción *(I)* de la Figura 10.13 estará a 1 después de la secuencia de la interrupción reinicialización.

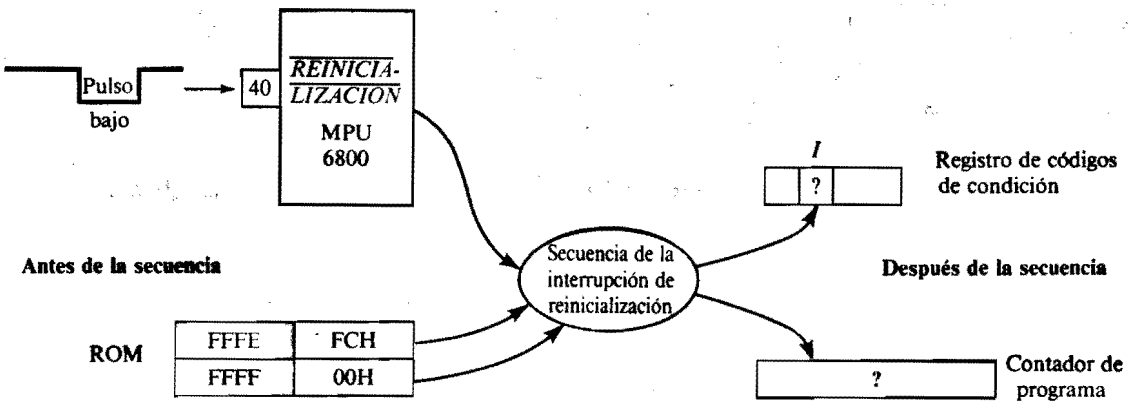


Figura 10.13.

10.23. Acudir a la Figura 10.13. El contenido del contador de programa después de la secuencia de la interrupción reinicialización será _____ (hex). Esta es la dirección de comienzo de la rutina de _____ _____ .

Solución:

Según la información de la Figura 10.8*a*, el contenido del contador de programa después de la secuencia de la interrupción reinicialización de la Figura 10.13 será FC00H. Esta es la dirección de comienzo de la rutina de servicio de reinicialización.

10.24. Acudir a la Figura 10.14. Listar el contenido de la pila después de la secuencia de interrupción no enmascarable mostrada en este ejemplo.

Solución:

Acudir al procedimiento mostrado en la Figura 10.8*b*. El contenido de la pila de la Figura 10.14 después de la secuencia de interrupción no enmascarable es el siguiente:

Posición de pila (hex)		Contenido
01F9	11001000	(Registro de códigos de condición)
01FA	11110000	(Acumulador *B*)
01FB	10000000	(Acumulador *A*)
01FC	00000000	(Registro índice – byte superior)
01FD	00000001	(Registro índice – byte inferior)
01FE	00000000	(Contador de programa – byte superior)
01FF	00010000	(Contador de programa – byte inferior)

10.25. Acudir a la Figura 10.14. El señalizador *I* estará a _____ (0, 1) después de la secuencia de interrupción no enmascarable mostrada en este ejemplo.

Solución:

Acudir a la Figura 10.8*b*. El señalizador *I* estará a 1 después de la secuencia de interrupción no enmascarable mostrada en la Figura 10.14.

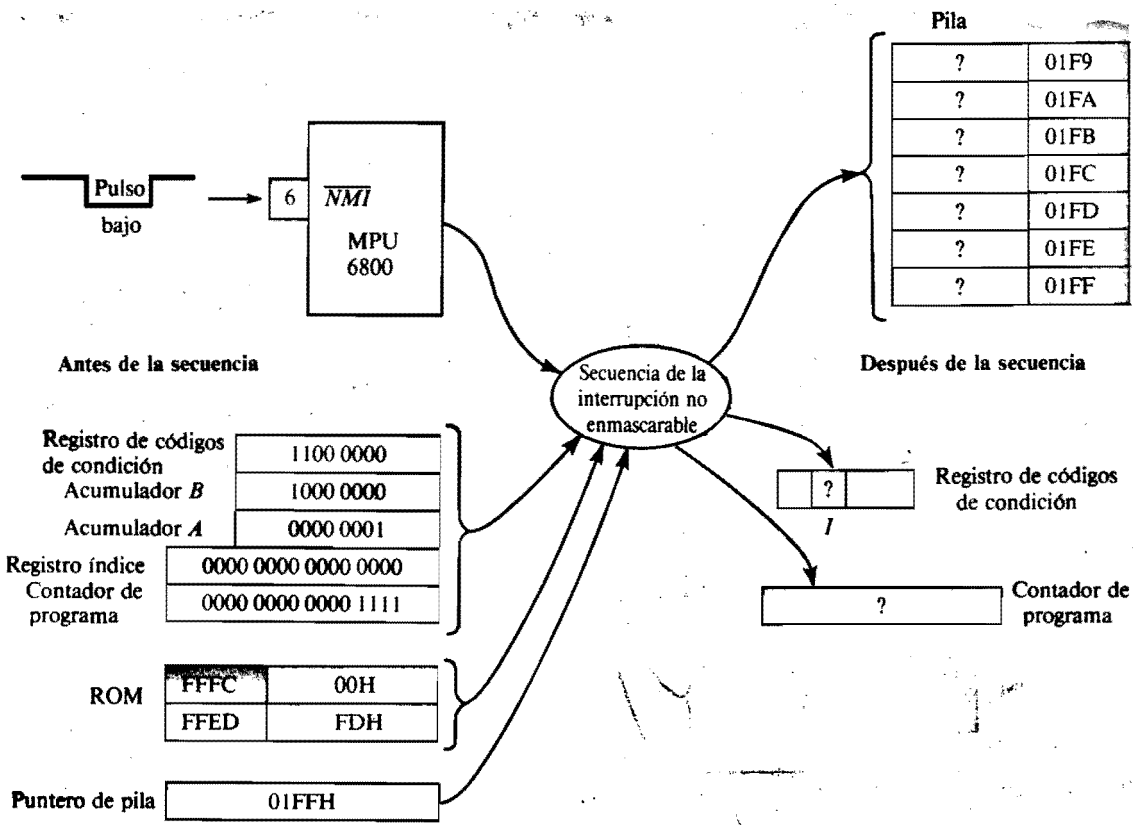


Figura 10.14.

10.26. Acudir a la Figura 10.14. El contenido del contador de programa después de la secuencia de interrupción no enmascarable será _____ (hex).

Solución:

Acudir a la Figura 10.8*b*. El contenido del contador de programa después de la secuencia de interrupción no enmascarable de la Figura 10.14 será 00FDH. Esta es la dirección de una rutina de servicio de la interrupción no enmascarable.

10.27. Acudir a la Figura 10.15. El señalizador de petición de interrupción *(I)* está a _____ (0, 1) *antes* de la secuencia de petición de interrupción.

Solución:

Acudir a la Figura 10.4 para las posiciones del señalizador en el *CCR*. El *CCR* contiene 11001000 antes de la secuencia mostrada en la Figura 10.15. Esto significa que el señalizador de petición de interrupción *(I)* está a 0 antes de la secuencia de petición de interrupción. Esto tendrá el efecto de permitir las entradas $\overline{IRQ}$ que sean aceptadas por la MPU 6800.

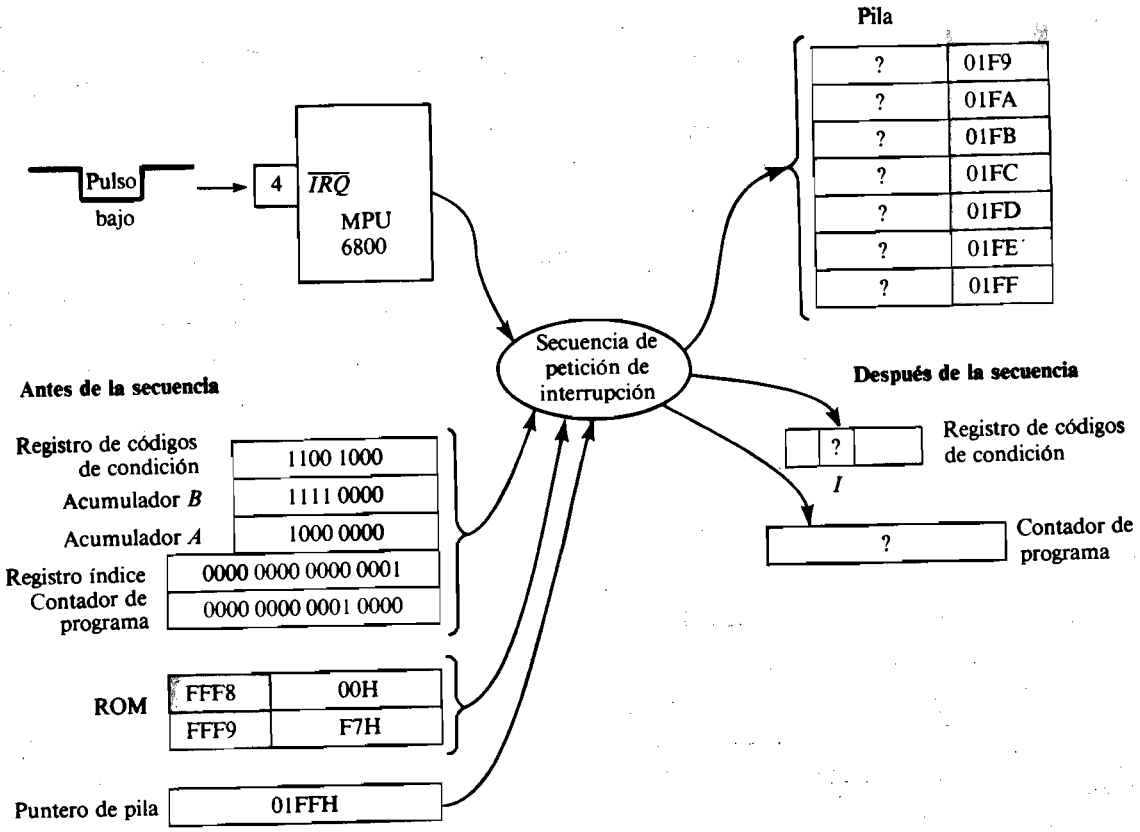


Figura 10.15.

10.28. Acudir a la Figura 10.15. Listar el contenido de la pila después de la secuencia de petición de interrupción.

Solución:

Acudir a la Figura 10.8*c*. El contenido de la pila de la Figura 10.15 después de la secuencia de petición de interrupción es el siguiente:

Posición de pila (hex)		Contenido
01F9	11001000	(Registro de códigos de condición)
01FA	11110000	(Acumulador *B*)
01FB	10000000	(Acumulador *A*)
01FC	00000000	(Registro índice – byte superior)
01FD	00000001	(Registro índice – byte inferior)
01FE	00000000	(Contador de programa – byte superior)
01FF	00010000	(Contador de programa – byte inferior)

10.29. Acudir a la Figura 10.15. El señalizador *I* estará a _____ (0, 1) después de la secuencia de petición de interrupción mostrada en este ejemplo.

Solución:

Acudir a la Figura 10.8*c*. El señalizador *I* estará a 1 después de la secuencia de petición de interrupción de la Figura 10.15.

10.30. Acudir a la Figura 10.15. El contenido del contador de programa después de la secuencia de petición de interrupción será _____ (hex). Esta es la dirección de la rutina _____ _____ _____ .

Solución:

Acudir a la Figura 10.8*c*. El contenido del contador de programa después de la secuencia de petición de interrupción en la Figura 10.15 será 00F7H. Esta es la dirección de la rutina de servicio de petición de interrupción.

10.4. MODOS DE DIRECCIONAMIENTO

Los modos de direccionamiento utilizados por el microprocesador Motorola 6800 son:

1. Inherente (implícito).
2. Inmediato.
3. Directo y extendido.
4. Indexado.
5. Relativo.

Direccionamiento inherente

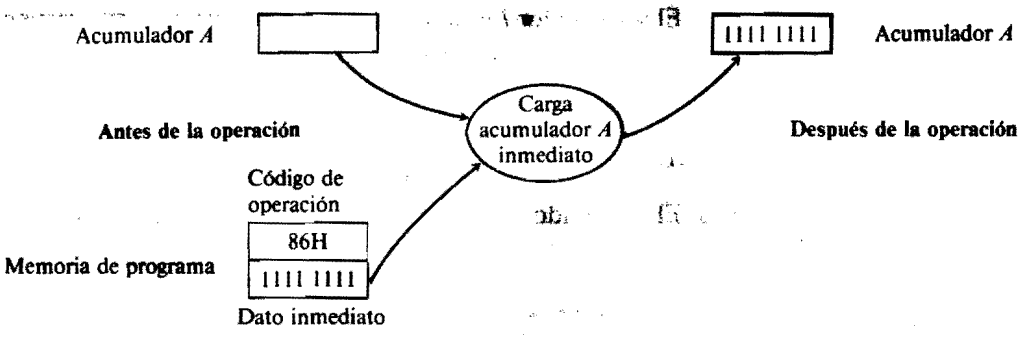
Muchas instrucciones del microprocesador 6800 utilizan *direccionamiento inherente*. Las instrucciones inherentes se utilizan cuando no es necesario buscar ningún operando en el exterior del procesador. A este tipo de direccionamiento se denominó direccionamiento implícito en los procesadores 8080/8085 y genérico. Las operaciones inherentes, o implícitas, son siempre instrucciones de 1 byte. Algunos ejemplos de instrucciones inherentes, o implícitas, son la de borrar (poner a cero), incrementar y decrementar los acumuladores.

Determinadas informaciones sobre la MPU 6800 de Motorola mencionan el direccionamiento de los acumuladores, que puede considerarse como un caso especial de direccionamiento inherente para los propósitos de este capítulo.

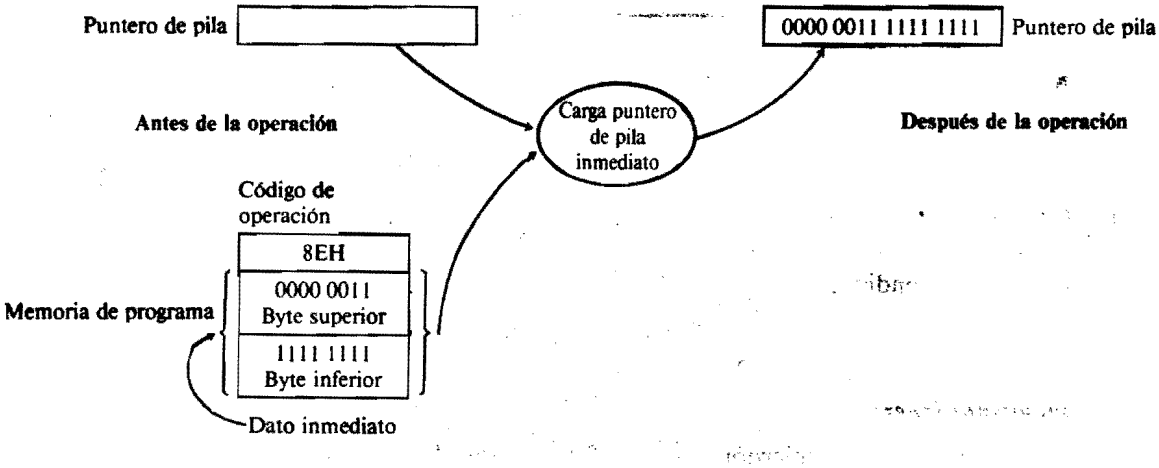
Direccionamiento inmediato

Igual que los procesadores genérico y 8080/8085, la MPU 6800 tiene instrucciones que utilizan *el modo de direccionamiento inmediato*. En las instrucciones inmediatas, el operando está localizado inmediatamente después del código de op en la memoria de programa.

La instrucción «cargar el acumulador *A* inmediato» se está ejecutando en la Figura 10.16*a*. En este ejemplo, el código de op del 6800 para la instrucción «cargar acumulador *A* inmediato» es 86H en la Figura 10.16*a*. El dato a ser cargado (11111111 en este ejemplo) es el segundo byte de la instrucción en la memoria de programa. El dato inmediato es entonces cargado en el acumulador *A*.



(a) Instrucción de 2 bytes con direccionamiento inmediato



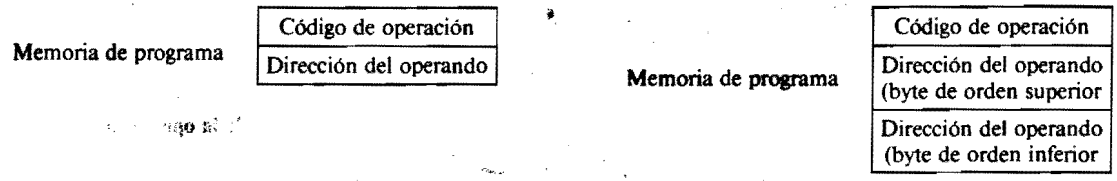
(b) Instrucción de 3 bytes con direccionamiento inmediato

Figura 10.16.

La mayoría de las operaciones inmediatas del 6800 son instrucciones de 2 bytes. Sin embargo, algunas son instrucciones de 3 bytes. Un ejemplo de una instrucción inmediata de 3 bytes se muestra en la Figura 10.16*b*. En este ejemplo, se está ejecutando la instrucción «cargar el puntero de pila inmediato». Observar que el segundo byte de la instrucción es el byte superior (00000011 en este ejemplo), mientras que el tercer byte es el byte inferior (11111111 en este caso).

Direccionamiento directo y extendido

Las operaciones que usan *direccionamiento directo* en la MPU 6800 se especifican mediante formatos de instrucción de 2 bytes. Este formato está representado en la Figura 10.17*a*. El byte 1 contiene el código de op para la instrucción de direccionamiento directo. El byte 2 contiene la *dirección* del operando. Debido a que la dirección es solamente de 8 bits, el rango de direcciones disponibles está limitado a las 256_{10} primeras posiciones de memoria. El direccionamiento directo



(a) Formato de instrucción para el direccionamiento directo del 6800

(b) Formato de instrucción para el direccionamiento extendido del 6800

Figura 10.17.

de Motorola es similar al direccionamiento de página base o de página cero en otros procesadores (como el 6502).

Las operaciones que utilizan el *direccionamiento extendido* en la MPU 6800 se especifican mediante instrucciones cuyo formato es de 3 bytes. Este es el mismo que el empleado en el direccionamiento directo en los microprocesadores genérico, 8080 y 8085. El formato de instrucción del 6800 para el direccionamiento extendido está representado en la Figura 10.17*b*. El byte 1 contiene el código de op para la instrucción de direccionamiento extendido. El byte 2 contiene el byte de orden superior de la dirección del operando. El byte 3 contiene el byte de orden inferior de la dirección del operando. Con el direccionamiento extendido, puede direccionarse el rango completo de 65.536_{10} posiciones de memoria.

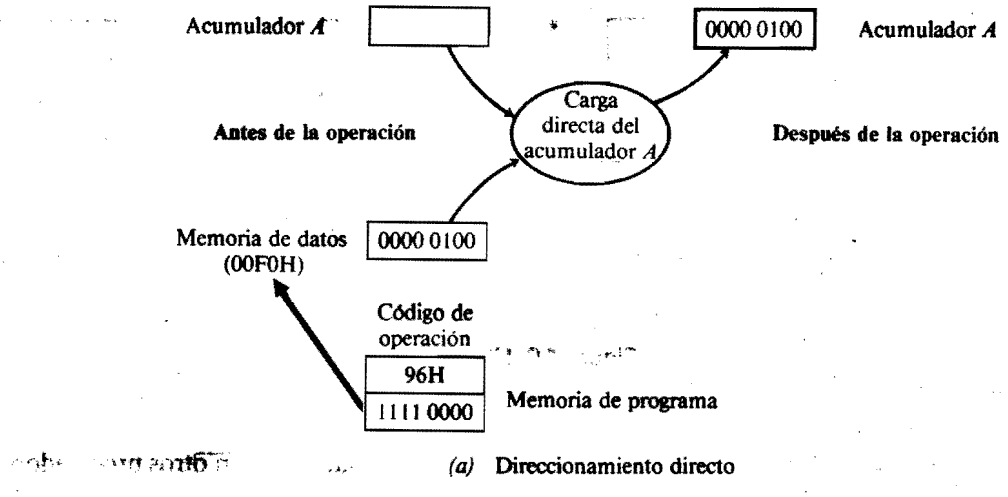
Un ejemplo de direccionamiento directo utilizando la MPU 6800 se detalla expresamente en la Figura 10.18*a*. La instrucción «carga el acumulador *A* directo» está ejecutándose en este ejemplo. El segundo byte del programa (11110000 en este ejemplo) es la dirección del operando. El operando (00000100 en este ejemplo) es buscado entonces en la posición de memoria 00F0H y depositado en el acumulador *A*.

Un ejemplo de direccionamiento extendido se muestra en la Figura 10.18*b*. El segundo y tercer bytes de la instrucción en la memoria del programa son ensamblados en una dirección de 16 bits (0000 0001 0000 0000 en este ejemplo). El contenido de esta dirección se busca en la memoria de datos y se deposita en el acumulador *A*.

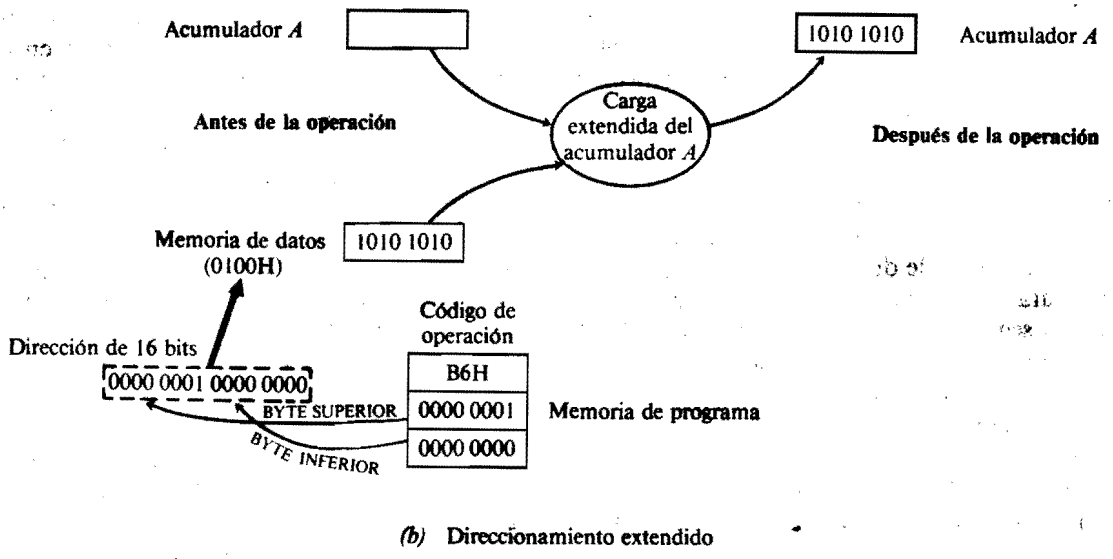
Direccionamiento indexado

El microprocesador 6800 contiene un registro índice de 16 bits *(IX)* que se utiliza en el *direccionamiento indexado*. El modo indexado requiere el uso de una instrucción de 2 bytes cuyo formato se muestra en la Figura 10.19*a*. El primer byte contiene el código de op habitual para la instrucción indexada y el segundo byte contiene el *desplazamiento*. El *desplazamiento se suma al contenido del registro índice* para determinar la dirección del operando.

Un ejemplo de una instrucción que utiliza el modo de direccionamiento indexado se detalla en la Figura 10.19*b*, donde se está ejecutando la instrucción «carga el acumulador *A* indexado». El direccionamiento indexado es como el direccionamiento extendido, ya que se genera una dirección de 16 bits. La dirección del operando se genera sumando el contenido del registro índice *(IX)* de 16 bits al desplazamiento localizado en la memoria de programa. Esta nueva dirección (0030H en este ejemplo) es la posición del operando. El operando (11110000 en este ejemplo) se carga entonces en el acumulador *A*.



(a) Direccionamiento directo



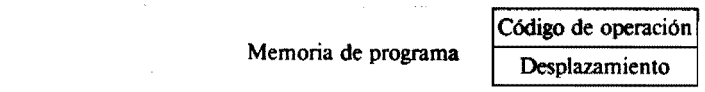
(b) Direccionamiento extendido

Figura 10.18.

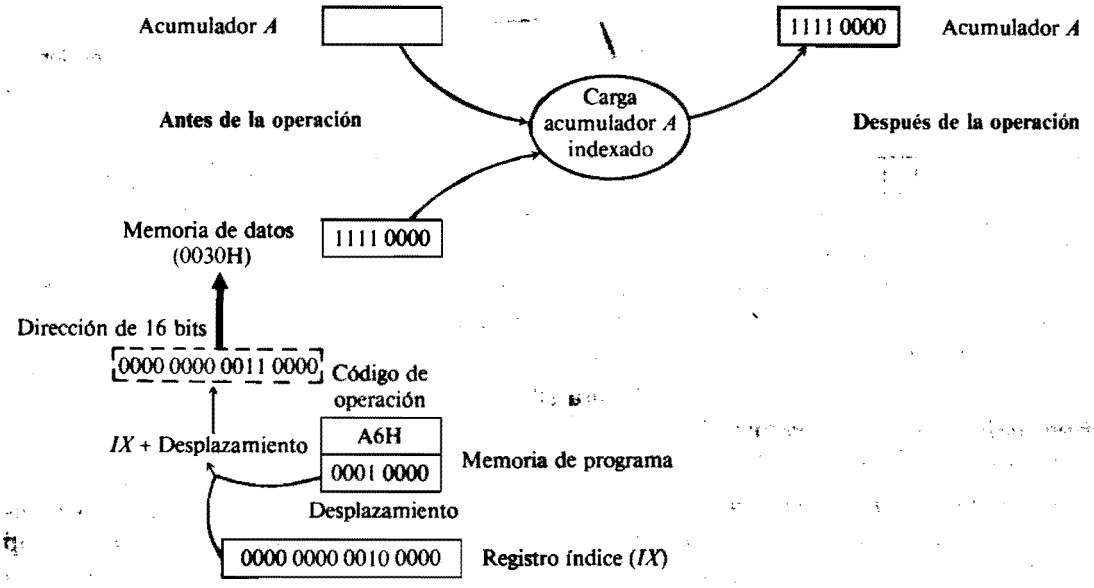
El valor del desplazamiento recuperado del segundo byte de la instrucción puede ser cualquier número sin signo comprendido entre 00H y FFH ($0–255_{10}$).

Direccionamiento relativo

El *modo de direccionamiento relativo* se utiliza en la mayoría de las instrucciones de bifurcación en el microprocesador 6800. El formato de las instrucciones de direccionamiento relativo se muestra en la Figura 10.20*a*. El primer byte de la instrucción es el código de op. El segundo byte de la instrucción se denomina *desplazamiento*, el cual se interpreta como un *número de 7 bits con*



(a) Formato de instrucción para el direccionamiento indexado

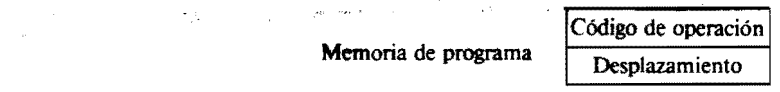


(b) Direccionamiento indexado

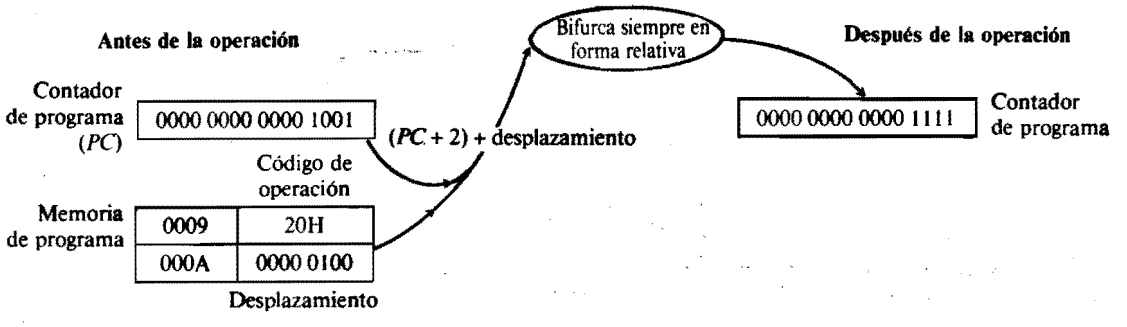
Figura 10.19.

signo. Si el MSB del desplazamiento es 0, el número es positivo, y esto significa que se va a ejecutar una bifurcación o salto hacia adelante. Si el MSB del desplazamiento es 1, el número es negativo, lo que significa que se va a ejecutar una bifurcación o salto hacia atrás.

Un ejemplo de *salto hacia adelante* utilizando el direccionamiento relativo del 6800 se muestra en la **Figura** 10.20*b*. El contador de programa *(PC)* antes de la operación contiene 0009H. El código de op para la instrucción «bifurca siempre de forma relativa» (20H) es buscado en la posición 0009H de la memoria de programa. El contador de programa es incrementado a 000AH, y se busca el desplazamiento (00000100). El contador de programa se incrementa a la dirección de la siguiente instrucción (000BH) justo antes de buscar el operando real. El procesador 6800 suma internamente el desplazamiento (04H) al contenido del contador de programa después de la operación «bifurca siempre de forma relativa» será entonces 000B + 04 = 000FH ($0000\ 0000\ 0000\ 1111_2$). El procesador entonces saltará a esta nueva dirección y buscará una instrucción en la posición 000FH. Observar que el MSB del desplazamiento es 0. Esto significa que un desplazamiento positivo, realizará un salto o bifurcación hacia adelante.



(a) Formato de instrucción para direccionamiento relativo



(b) Direccionamiento relativo; salto hacia adelante

Figura 10.20.

Un ejemplo de *salto hacia atrás* utilizando direccionamiento relativo se ilustra en la Figura 10.21. El contador de programa *(PC)* antes de la operación contiene 001DH. El código de op para la instrucción «bifurcar siempre de forma relativa» es 20H y está en la posición 001DH de la memoria de programa. El desplazamiento de la posición 001EH de la memoria del programa es el número en complemento a 2 11111010 (−06H). El desplazamiento negativo significa que la bifurcación será hacia atrás. La dirección de destino de la operación «bifurca siempre de forma relativa» se calcula en dos pasos. Primero, suma +2 al contador de programa.

PC	0000 0000 0001 1101	001DH
+2	+ 0010	+ 2H
	0000 0000 0001 1111	001FH

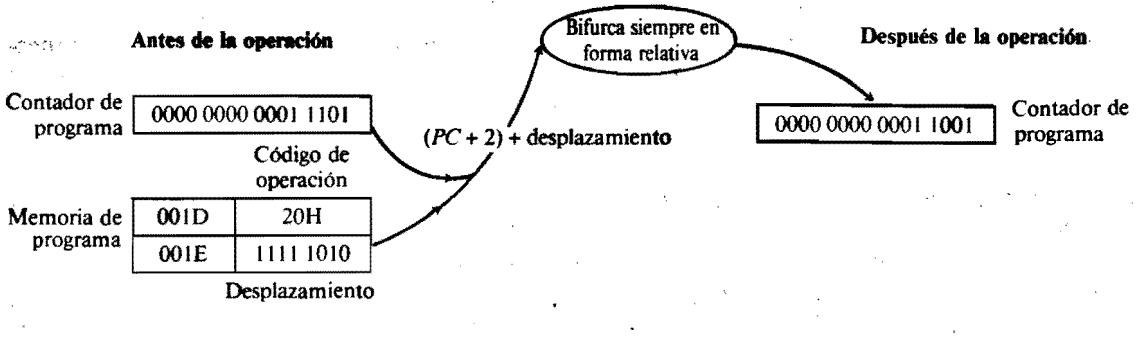
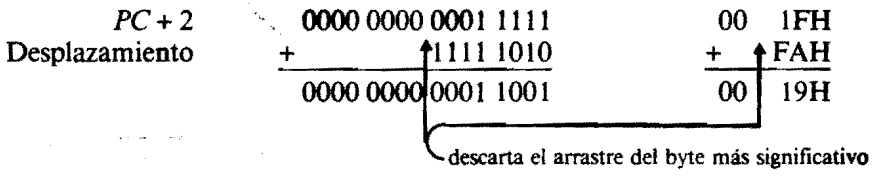


Figura 10.21. Direccionamiento relativo; salto hacia atrás.

Segundo, suma el desplazamiento a la suma parcial.



La nueva dirección es entonces 0019H. El contador de programa de la Figura 10.21 contiene la nueva dirección 0000 0000 0001 1001_2 después de la operación «bifurca siempre de forma relativa».

PROBLEMAS RESUELTOS

10.31. Cuando todos los operandos están localizados en el microprocesador 6800, se utilizan instrucciones que utilizan el modo de direccionamiento implícito o _____. Estas operaciones se especifican utilizando instrucciones de _____ (número) byte(s).

Solución:

Cuando todos los operandos están localizados en el microprocesador 6800, se utilizan instrucciones que utilizan el modo de direccionamiento implícito o inherente. Estas operaciones se especifican utilizando instrucciones de 1 byte.

10.32. Acudir a la Figura 10.22. Después de la operación «cargar registro índice inmediato», el registro índice contendrá _____ (hex).

Solución:

Acudir al problema ejemplo de la Figura 10.16*b*. Después de la operación «cargar registro índice inmediato» de la Figura 10.22, el registro índice contendrá 0011H (0000 0000 0001 0001_2).

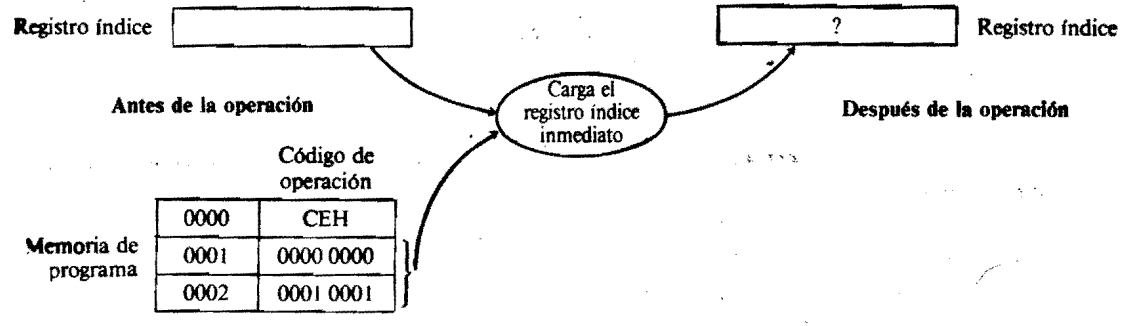


Figura 10.22.

10.33. Acudir a la Figura 10.23. La dirección de la memoria de datos en la izquierda es _____ (hex) como especifican el segundo y tercer bytes de la instrucción en _____ _____.

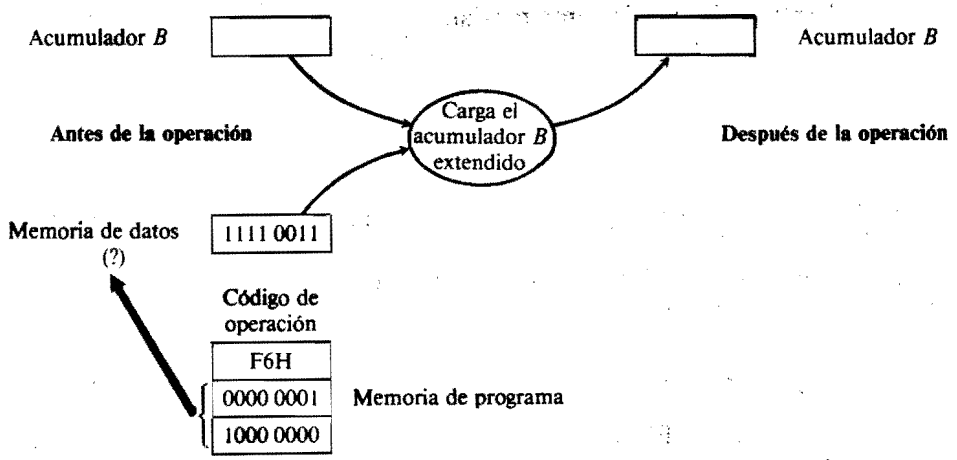


Figura 10.23.

Solución:

Acudir al problema ejemplo de la Figura 10.18*b*. La dirección de la posición de la memoria de datos en la izquierda de la Figura 10.23 es 0180H como especifican el segundo y tercer bytes de la instrucción en la memoria del programa.

10.34. Acudir a la Figura 10.23. El contenido del acumulador *B* después de la operación «cargar el acumulador *B* extendido» será _____ (8 bits).

Solución:

El contenido del acumulador *B* después de la operación «cargar el acumulador *B* extendido» de la Figura 10.23 será 11110011.

10.35. Acudir a la Figura 10.24. El código de op E6H especifica la instrucción « _____ _____ _____ _____ ».

Solución:

El código de op E6H de la Figura 10.24 especifica la instrucción «cargar el acumulador *B* indexado».

10.36. Acudir a la Figura 10.24. El contenido de la posición de la memoria de datos _____ (hex) se cargará en el acumulador *B*. Después de la operación de carga, el contenido del acumulador *B* será _____ (8 bits).

Solución:

Acudir al problema ejemplo de la Figura 10.19. El contenido de la posición de la memoria de datos 0041H de la Figura 10.24 se cargará en el acumulador *B*. Después de la operación de carga, el contenido del acumulador *B* será 11110000. La posición de la memoria de datos se calculó sumando el contenido del registro índice al desplazamiento (0030 + 11 = 0041H).

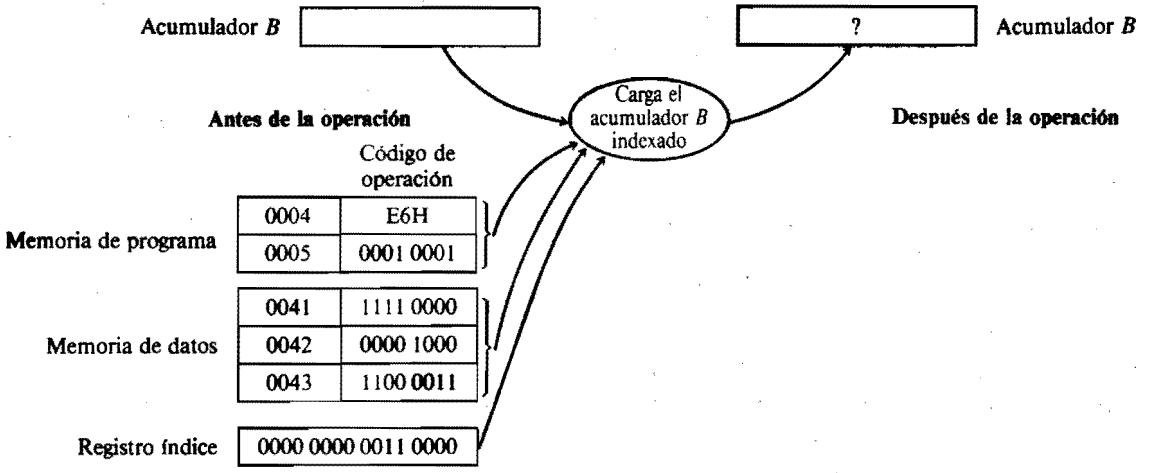


Figura 10.24.

10.37. Acudir a la Figura 10.25. La condición de bifurcación es _____ (falsa, cierta) en este problema.

Solución:

Acudir a la Figura 10.4 para la posición del señalizador de arrastre *(C)*. La condición de bifurcación (bifurcar si arrastre está a 0) es cierta en este problema. Por tanto, el procesador saltará a una nueva posición.

10.38. Acudir a la Figura 10.25. El desplazamiento de la memoria de programa comienza con un _____ (0, 1); por tanto, el microprocesador 6800 bifurcará hacia _____ (atrás, adelante).

Solución:

El desplazamiento de la memoria de programa en la Figura 10.25 comienza con 0; por tanto, el microprocesador 6800 bifurcará hacia adelante.

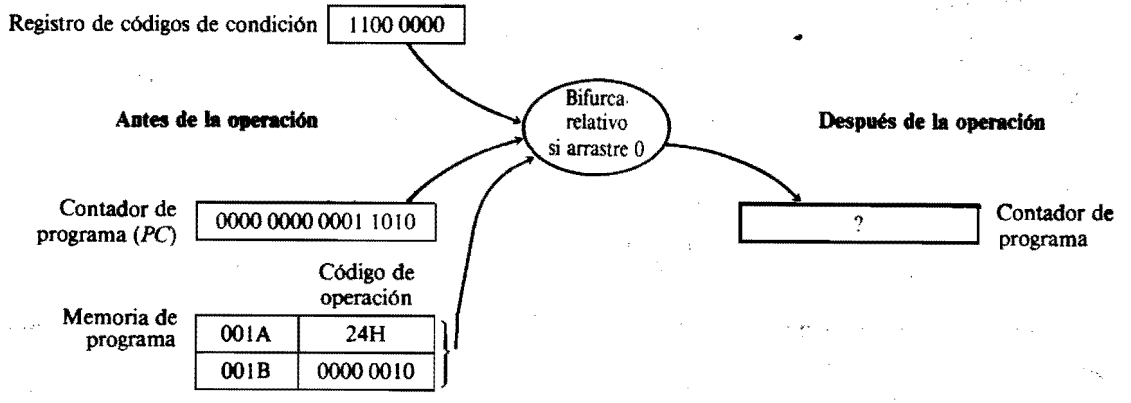


Figura 10.25.

10.39. Acudir a la Figura 10.25. El contenido del contador de programa después de la operación de bifurcación será _____ (hex) o _____ (binario).

Solución:

Ver el problema ejemplo de la Figura 10.20. El contenido del contador de programa después de la operación de bifurcar será 001EH o 0000 0000 0001 1110_2. Esto se calcula utilizando la fórmula ($PC + 2$) + desplazamiento = nueva dirección.

10.5. REPERTORIO DE INSTRUCCIONES DEL 6800

El microprocesador Motorola 6800 se utiliza en computadoras de programa almacenado. Las instrucciones del programa se almacenan como bytes de 8 bits en un área de almacenamiento denominada memoria del programa. La MPU del 6800 utiliza instrucciones de 1, 2 y 3 bytes. El primer byte de la instrucción es siempre el código de op (código de operación), que especifica una de las 197 posibles instrucciones que puede realizar la MPU 6800. Como en la mayoría de los microprocesadores de propósito general, el repertorio de instrucciones está fijado permanentemente en el diseño de la pastilla 6800.

Motorola agrupa las instrucciones del microprocesador **6800** en los siguientes grupos:

1. Instrucciones de acumulador y memoria.
2. Instrucciones del registro índice y puntero de pila.
3. Instrucciones de bifurcación y salto.
4. Instrucciones del registro de códigos de condición.

Cada uno de estos grupos será cubierto con mayor detalle en secciones posteriores de este mismo capítulo.

Un resumen del *repertorio de instrucciones de la MPU 6800* se detalla en la Figura 10.26. Este es un listado alfabético por nemotécnicos. Los nemotécnicos de Motorola se utilizan para los 197 códigos de op únicos. Los códigos de op están representados en notación hexadecimal en este resumen. También se da una breve descripción para cada uno de los nemotécnicos del 6800. De forma distinta a los procesadores genérico y 8080/8085, la MPU 6800 no utiliza un nemotécnico único para cada código de op. Algunos nemotécnicos tienen cuatro posibles códigos de op, dependiendo de los modos de direccionamiento. A causa de esto, en la Figura 10.26 se da también el modo de direccionamiento para cada una de las instrucciones del 6800.

Considerar la primera instrucción de la Figura 10.26. La instrucción «sumar el acumulador *B* al acumulador *A*» utiliza el nemotécnico ABA y un código de op hexadecimal de 1BH. De acuerdo con la Figura 10.26, utiliza direccionamiento inherente. La operación de la instrucción ABA se resume como

$$\text{Sumar } A \text{ a } B \rightarrow A$$

Esto significa sumar el acumulador *A* al acumulador *B* y el resultado se deposita en el acumulador *A*.

El resumen del repertorio de instrucciones del 6800 de la Figura 10.26 no da detalles sobre señalizadores, ciclos de reloj o número de bytes de memoria utilizados por cada instrucción. Sin embargo, da al programador una rápida visión general de las operaciones disponibles al utilizar el microprocesador Motorola 6800.

	Nemotécnico	Código de operación (hex)	Modo de direcciona-miento	Descripción
A	ABA	1B	Inherente	Suma A a $B \rightarrow A$
	ADC A	89 99 A9 B9	Inmediato Directo Indexado Extendido	Suma A a M con arrastre $\rightarrow A$
	ADC B	C9 D9 E9 F9	Inmediato Directo Indexado Extendido	Suma B a M con arrastre $\rightarrow B$
	ADD A	8B 9B AB BB	Inmediato Directo Indexado Extendido	Suma A a $M \rightarrow A$
	ADD B	CB DB EB FB	Inmediato Directo Indexado Extendido	Suma B a $M \rightarrow B$
	AND A	84 94 A4 B4	Inmediato Directo Indexado Extendido	AND de A con $M \rightarrow A$
	AND B	C4 D4 E4 F4	Inmediato Directo Indexado Extendido	AND de B con $M \rightarrow B$
	ASL	68 78	Indexado Extendido	Desplazamiento aritmético de M a la izquierda con el arrastre
	ASL A	48	Inherente	Desplazamiento aritmético de A a la izquierda con el arrastre
	ASL B	58	Inherente	Desplazamiento aritmético de B a la izquierda con el arrastre
	ASR	67 77	Indexado Extendido	Desplazamiento aritmético de M a la derecha con el arrastre
	ASR A	47	Inherente	Desplazamiento aritmético de A a la derecha con el arrastre
	ASR B	57	Inherente	Desplazamiento aritmético de B a la derecha con el arrastre
B	BCC	24	Relativo	Si señalizador $C = 0$, entonces bifurca
	BCS	25	Relativo	Si señalizador $C = 1$, entonces bifurca
	BEQ	27	Relativo	Si señalizador $Z = 1$, entonces bifurca
	BGE	2C	Relativo	Si número en complemento a $2 \geq 0$, entonces bifurca
	BGT	2E	Relativo	Si número en complemento a $2 > 0$, entonces bifurca
	BHI	22	Relativo	Si minuendo > sustraendo, entonces bifurca
	BIT A	85 95 A5 B5	Inmediato Directo Indexado Extendido	AND de A con M, señalizadores afectados (A y M no cambian)
	BIT B	C5 D5 E5 F5	Inmediato Directo Indexado Extendido	AND de B con M, señalizadores afectados (B y M no cambian)
	BLE	2F	Relativo	Si el número en complemento a $2 \leq 0$, entonces bifurca
	BLS	23	Relativo	Si sustraendo $\geq$ minuendo, entonces bifurca

Figura 10.26. Resumen del repertorio de instrucciones del microprocesador 6800.

	Nemotécnico	Código de operación (hex)	Modo de direcciona-miento	Descripción
B	BLT	2D	Relativo	Si el número en complemento a 2 < 0, entonces bifurca
	BMI	2B	Relativo	Si menos ($N = 1$), entonces bifurca
	BNE	26	Relativo	Si no igual a 0 ($Z = 0$), entonces bifurca
	BPL	2A	Relativo	Si más ($N = 0$), entonces bifurca
	BRA	20	Relativo	Bifurca siempre
	BSR	8D	Relativo	Bifurca a subrutina (*PC* a pila)
	BVC	28	Relativo	Si overflow borrado ($V = 0$), entonces bifurca
	BVS	29	Relativo	Si overflow a 1 ($V = 1$), entonces bifurca
C	CBA	11	Inherente	Compara *A* con *B* (*A* − *B*)
	CLC	0C	Inherente	Borra señalizador *C*
	CLI	0E	Inherente	Borra señalizador *I*
	CLR	6F 7F	Indexado Extendido	Borra posición de memoria
	CLR *A*	4F	Inherente	Borra *A*
	CLR *B*	5F	Inherente	Borra *B*
	CLV	0A	Inherente	Borra señalizador *V*
	CMP *A*	81 91 A1 B1	Inmediato Directo Indexado Extendido	Compara *A* con *M* (*A* − *M*)
	CMP *B*	C1 D1 E1 F1	Inmediato Directo Indexado Extendido	Compara *B* con *M* (*B* − *M*)
	COM	63 73	Indexado Extendido	Complementa *M* (complemento a 1)
	COM *A*	43	Inherente	Complementa *A* (complemento a 1)
	COM *B*	53	Inherente	Complementa *B* (complemento a 1)
	CPX	8C 9C AC BC	Inmediato Directo Indexado Extendido	Compara registro índice con *M* (X_H/X_L) − (*M*/*M* + 1)
D	DA *A*	19	Inherente	Ajuste decimal de *A* (convierte la suma binaria de caracteres de BCD al formato BCD)
	DEC	6A 7A	Indexado Extendido	Decrementa *M*
	DEC *A*	4A	Inherente	Decrementa *A*
	DEC *B*	5A	Inherente	Decrementa *B*
	DES	34	Inherente	Decrementa el puntero de pila
	DEX	09	Inherente	Decrementa el registro de índice
E	EOR *A*	88 98 A8 B8	Inmediato Directo Indexado Extendido	OR exclusiva de *A* con *M* → *A*
	EOR *B*	C8 D8 E8 F8	Inmediato Directo Indexado Extendido	OR exclusiva de *B* con *M* → *B*

Figura 10.26. *(continuación).*

	Nemotécnico	Código de operación (hex)	Modo de direcciona-miento	Descripción
I	INC	6C 7C	Indexado Extendido	Incrementa M
	INC A	4C	Inherente	Incrementa A
	INC B	5C	Inherente	Incrementa B
	INS	31	Inherente	Incrementa el puntero de pila
	INX	08	Inherente	Incrementa el registro de índice
J	JMP	6E 7E	Indexado Extendido	Salta a dirección
	JSR	AD BD	Indexado Extendido	Salta a subrutina (PC en pila)
L	LDA A	86 96 A6 B6	Inmediato Directo Indexado Extendido	Carga M en A
	LDA B	C6 D6 E6 F6	Inmediato Directo Indexado Extendido	Carga M en B
	LDS	8E 9E AE BE	Inmediato Directo Indexado Extendido	Carga M en SP, $M \rightarrow SP_H$, $(M+1) \rightarrow X_L$
	LDX	CE DE EE FE	Inmediato Directo Indexado Extendido	Carga M en registro índice, $M \rightarrow X_H$, $(M+1) \rightarrow X_L$
	LSR	64 74	Indexado Extendido	Desplazamiento lógico de M a la derecha en arrastre
	LSR A	44	Inherente	Desplazamiento lógico de A a la derecha en arrastre
	LSR B	54	Inherente	Desplazamiento lógico de B a la derecha en arrastre
N	NEG	60 70	Indexado Extendido	Complemento a 2 de $M \rightarrow M$
	NEG A	40	Inherente	Complemento a 2 de $A \rightarrow A$
	NEG B	50	Inherente	Complemento a 2 de $B \rightarrow B$
	NOP	01	Inherente	No operación (se utilizan 2 ciclos)
O	ORA A	8A 9A AA BA	Inmediato Directo Indexado Extendido	OR de A con $M \rightarrow A$
	ORA B	CA DA EA FA	Inmediato Directo Indexado Extendido	OR de B con $M \rightarrow B$
P	PSH A PSH B PUL A PUL B	36 37 32 33	Inherente Inherente Inherente Inherente	Introduce A en la pila Introduce B en la pila Saca A de la pila Saca B de la pila
R	ROL	69 79	Indexado Extendido	Rota M a la izquierda por medio del acarreo

Figura 10.26. *(continuación).*

	Nemotécnico	Código de operación (hex)	Modo de direcciona-miento	Descripción
R	ROL *A*	49	Inherente	Rota *A* a la izquierda por medio del acarreo
	ROL *B*	59	Inherente	Rota *B* a la izquierda por medio del acarreo
	ROR	66 76	Indexado Extendido	Rota *M* a la derecha por medio del acarreo
	ROR *A*	46	Inherente	Rota *A* a la derecha por medio del acarreo
	ROR *B*	56	Inherente	Rota *B* a la derecha por medio del acarreo
	RTI	3B	Inherente	Vuelve de la interrupción (los señalizadores *A, B, X, PC* son sacados de la pila)
	RTS	39	Inherente	Vuelve de la subrutina (*PC* es sacado de la pila)
S	SBA	10	Inherente	Resta *B* de $A \rightarrow A$
	SBC *A*	82 92 A2 B2	Inmediato Directo Indexado Extendido	Resta *M* de *A* con arrastre ($A - M - M$)
	SBC *B*	C2 D2 E2 F2	Inmediato Directo Indexado Extendido	Resta *M* de *B* con arrastre ($B - M - C \rightarrow B$)
	SEC	0D	Inherente	Pone señalizador *C* en 1
	SEI	0F	Inherente	Coloca la bandera *I* en 1 (coloca la mascarilla de interrupción
	SEV	0B	Inherente	Pone señalizador *V* en 1
	STA *A*	97 A7 B7	Directo Indexado Extendido	Almacena *A* en *M*
	STA *B*	D7 E7 F7	Directo Indexado Extendido	Almacena *B* en *M*
	STS	9F AF BF	Directo Indexado Extendido	Almacena *SP* en *M*, $SP_H \rightarrow M$, $SP_L \rightarrow (M + 1)$
	STX	DF EF FF	Directo Indexado Extendido	Almacena *X* en *M*, $X_H \rightarrow M$, $X_L \rightarrow (M + 1)$
	SUB *A*	80 90 A0 B0	Inmediato Directo Indexado Extendido	Resta *M* de $A \rightarrow A$
	SUB *B*	C0 D0 E0 F0	Inmediato Directo Indexado Extendido	Resta *M* de $B \rightarrow B$
	SWI	3F	Inherente	Interrupción software
T	TAB	16	Inherente	Transfiere *A* a *B*
	TAP	06	Inherente	Transfiere *A* al *CCR* del procesador
	TBA	17	Inherente	Transfiere *B* a *A*
	TPA	07	Inherente	Transfiere *CCR* del procesador a *A*
	TST	6D 7D	Indexado Extendido	Test de *M* por (*M*) – 00, (los señalizadores *N* y *Z* se ponen a 1 ó 0)

Figura 10.26. *(continuación).*

	Nemotécnico	Código de operación (hex)	Modo de direcciona-miento	Descripción
T	TST *A*	4D	Inherente	Test de *B* por (*B*) – 00, (los señalizadores *N* y *Z* se ponen a 1 ó 0)
	TST *B*	5D	Inherente	Test de *A* por (*A*) – 00, (los señalizadores *N* y *Z* se ponen a 1 ó 0)
	TSX	30	Inherente	Transfiere $SP + 1 \rightarrow X$
	TXS	35	Inherente	Transfiere $X - 1 \rightarrow SP$
W	WAI	3E	Inherente	Espera interrupción (*PC, I, Z, B, CCR* son introducidos en la pila)

Figura 10.26. *(continuación).*

PROBLEMAS RESUELTOS

10.40. Motorola subdivide las instrucciones del microprocesador 6800 en cuatro grupos. Listarlos.

Solución:

Motorola subdivide las instrucciones del microprocesador 6800 en instrucciones de acumulador y memoria, instrucciones de registro índice y puntero de pila, instrucciones de salto y bifurcación e instrucciones de registro de códigos de condición.

10.41. Acudir a la Figura 10.26. La instrucción ADD *A* puede ser utilizada en uno de _____ (número) modos de direccionamiento. La instrucción ADD *A* suma el contenido del acumulador _____ (*A, B*) al contenido de una posición _____. El resultado se coloca en _____ _____.

Solución:

La instrucción ADD *A* listada en la Figura 10.26 puede ser utilizada en uno de cuatro modos de direccionamiento. La instrucción ADD *A* suma el contenido del acumulador *A* al contenido de una posición de memoria. El resultado se coloca en el acumulador *A*.

10.42. Acudir a la Figura 10.27. El código de op para la instrucción ADD *B* del 6800 utilizado en este ejemplo es _____ (hex). Este código de op especifica que la instrucción «suma *B* a *M*» utiliza el modo de direccionamiento _____.

Solución:

Ver la instrucción ADD *B* en la Figura 10.26. El código de op para la instrucción ADD *B* del 6800 utilizado en el ejemplo de la Figura 10.27 es DBH. Este código de op especifica que la instrucción «suma *B* a *M*» utiliza el modo de direccionamiento directo.

10.43. Acudir a la Figura 10.27. El contenido del acumulador *B* después de la operación de sumar será _____.

Solución:

La suma binaria 00001111 + 00000001 = 00010000. Por tanto el contenido del acumulador *B* después de la operación de suma de la Figura 10.27 será 00010000_2.

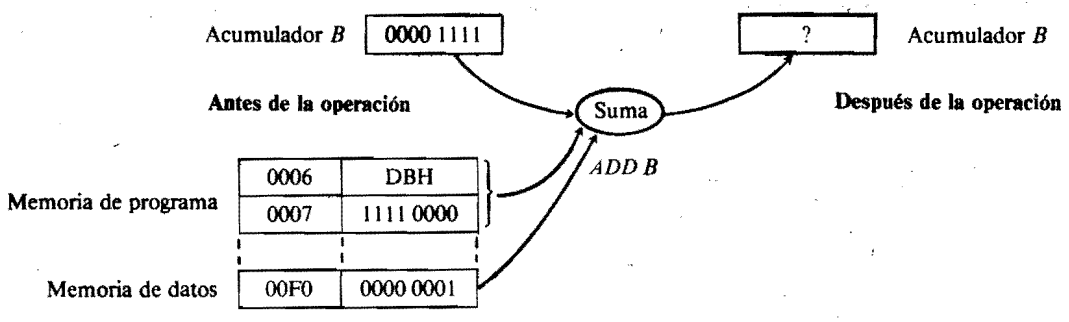


Figura 10.27.

10.44. Acudir a la Figura 10.28. La operación de borrado en este ejemplo utiliza el código de op _____ (hex). Esta instrucción utiliza el modo de direccionamiento _____ .

Solución:

Ver la instrucción CLR de la Figura 10.26. La operación de borrado mostrada en la Figura 10.28 utiliza el código de op 7FH. Esta instrucción utiliza el modo de direccionamiento extendido.

10.45. Acudir a la Figura 10.28. Listar el contenido de las posiciones de la memoria de datos 0080H, 0081H y 0082H después de la operación de borrado.

Solución:

El contenido de las posiciones de la memoria de datos mostradas en la Figura 10.28 después de la operación de borrado son:
Posición de memoria 0080 = 11111111 (no cambia)
Posición de memoria 0081 = 00000000 (borrada)
Posición de memoria 0082 = 00001111 (no cambia)

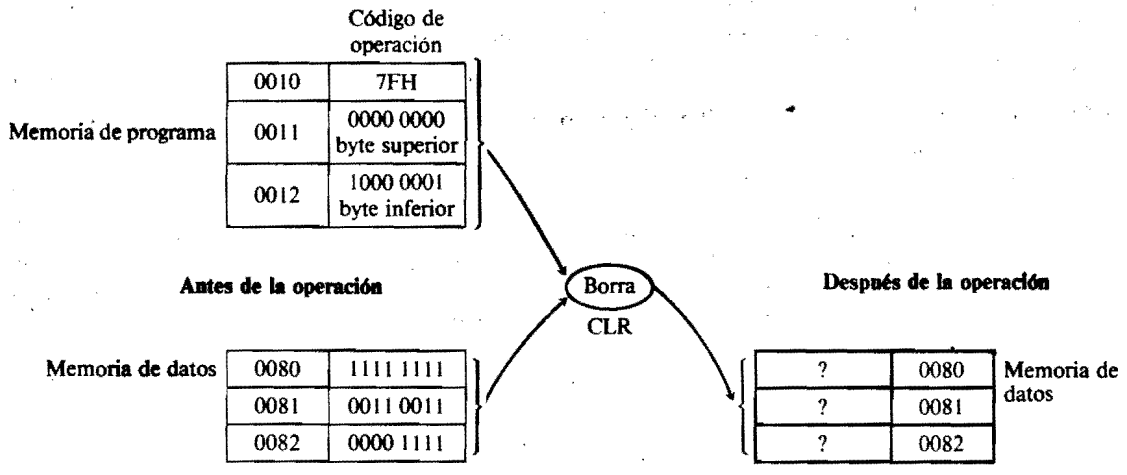


Figura 10.28.

10.46. Acudir a la Figura 10.29. El código de op para la instrucción DEC en este problema es _____ (hex). La operación «decrementar *M*» utilizada en este problema emplea el modo de direccionamiento _____.

Solución:

Ver la instrucción DEC de la Figura 10.26. El código de op para la instrucción DEC utilizada en la Figura 10.29 es 6AH. La operación «decrementar *M*» utilizada en este problema emplea el modo de direccionamiento indexado.

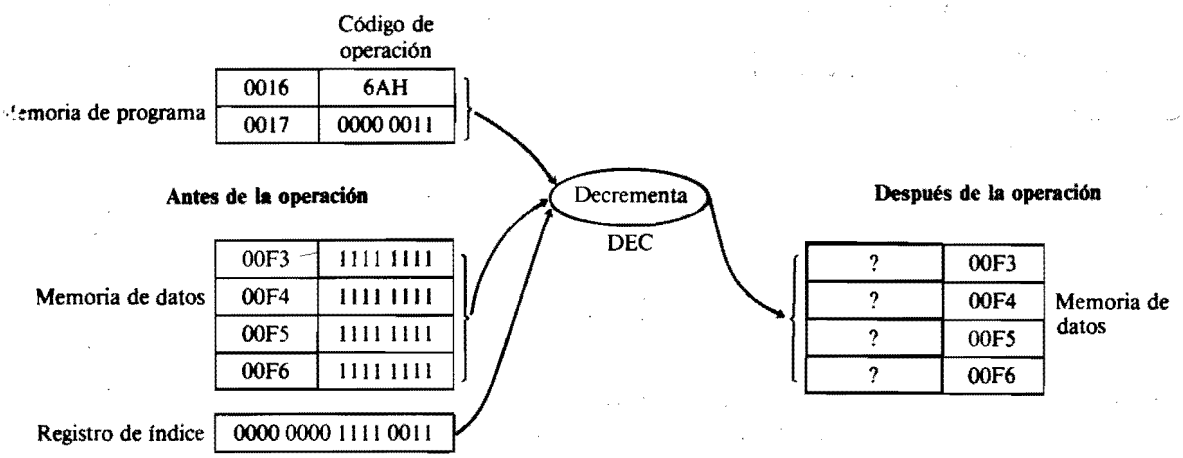


Figura 10.29.

10.47. Acudir a la Figura 10.29. Listar el contenido de las posiciones de la memoria de datos 00F3H, 00F4H, 00F5H y 00F6H después de la operación «decrementar *M*».

Solución:

Al utilizar el direccionamiento indexado, la nueva dirección a la que se apunta en la memoria de datos es 00F3 + 3 = 00F6H. Por tanto, el contenido de las posiciones de la memoria de datos después de la operación «decrementar *M*» será:

Posición de memoria 00F3 = 11111111 (no cambia)
Posición de memoria 00F4 = 11111111 (no cambia)
Posición de memoria 00F5 = 11111111 (no cambia)
Posición de memoria 00F6 = 11111110 (decrementadas)

10.6. INSTRUCCIONES DE MEMORIA Y ACUMULADOR DEL 6800

El resumen de las *instrucciones de memoria* y *acumulador* del microprocesador 6800 de Motorola está reproducido en la Figura 10.30. Estas incluyen las instrucciones aritméticas, lógicas, de prueba de datos y de transferencia de datos.

Considerar la primera operación listada en la Figura 10.30. La operación de suma regular puede ser realizada bien en el acumulador *A* o en el *B*. Por tanto, se listan dos nemotécnicos para

OPERACIONES	NEMOTECNICO	MODOS DE DIRECCIONAMIENTO: INMEDIATO OP	~	=	DIRECTO OP	~	=	INDEXADO OP	~	=	EXTENDIDO OP	~	=	IMPLICITO OP	~	=	OPERACION BOOLEANA/ARITMETICA (Todos los registros etiquetados se refieren a los contenidos)	REGISTRO DE CODIGOS DE CONDICION 5 H	4 I	3 N	2 Z	1 V	0 C
Suma	ADDA	8B	2	2	9B	3	2	AB	5	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Suma acumuladores	ABA													1B	2	1	A + B → A	↕	•	↕	↕	↕	↕
Suma con arrastre	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B + M + C → B	↕	•	↕	↕	↕	↕
AND	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				A · M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B · M → B	•	•	↕	↕	R	•
Test de bit	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A · M	•	•	↕	↕	R	•
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				B · M	•	•	↕	↕	R	•
Borra	CLR							6F	7	2	7F	6	3				00 → M	•	•	R	S	R	R
	CLRA													4F	2	1	00 → A	•	•	R	S	R	R
	CLRB													5F	2	1	00 → B	•	•	R	S	R	R
Compara	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3				A − M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				B − M	•	•	↕	↕	↕	↕
Compara acumuladores	CBA													11	2	1	A − B	•	•	↕	↕	↕	↕
Complementa a 1	COM							63	7	2	73	6	3				M̄ → M	•	•	↕	↕	R	S
	COMA													43	2	1	Ā → A	•	•	↕	↕	R	S
	COMB													53	2	1	B̄ → B	•	•	↕	↕	R	S
Complementa a 2 (Negar)	NEG							60	7	2	70	6	3				00 − M → M	•	•	↕	↕	①	②
	NEGA													40	2	1	00 − A → A	•	•	↕	↕	①	②
	NEGB													50	2	1	00 − B → B	•	•	↕	↕	①	②
Ajuste decimal, *A*	DAA													19	2	1	Convierte la suma binaria de caracteres BCD a formato BCD	•	•	↕	↕	↕	③
Decrementa	DEC							6A	7	2	7A	6	3				M − 1 → M	•	•	↕	↕	4	•
	DECA													4A	2	1	A − 1 → A	•	•	↕	↕	4	•
	DECB													5A	2	1	B − 1 → B	•	•	↕	↕	4	•
OR exclusiva	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3				A⊕M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				B⊕M → B	•	•	↕	↕	R	•
Incrementa	INC							6C	7	2	7C	6	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	2	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	2	1	B + 1 → B	•	•	↕	↕	⑤	•
Carga acumulador	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	•	•	↕	↕	R	•
OR, inclusiva	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Introduce datos	PSHA													36	4	1	A → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
	PSHB													37	4	1	B → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
Extrae datos	PULA													32	4	1	SP + 1 → SP, M_{SP} → A	•	•	•	•	•	•
	PULB													33	4	1	SP + 1 → SP, M_{SP} → B	•	•	•	•	•	•
Desplazamiento circular a la izquierda	ROL							69	7	2	79	6	3				M	•	•	↕	↕	⑥	↕
	ROLA													49	2	1	A	•	•	↕	↕	⑥	↕
	ROLB													59	2	1	B	•	•	↕	↕	⑥	↕
Desplazamiento circular a la derecha	ROR							66	7	2	76	6	3				M	•	•	↕	↕	⑥	↕
	RORA													46	2	1	A	•	•	↕	↕	⑥	↕
	RORB													56	2	1	B	•	•	↕	↕	⑥	↕
Desplazamiento a la izquierda aritmético	ASL							68	7	2	78	6	3				M	•	•	↕	↕	⑥	↕
	ASLA													48	2	1	A	•	•	↕	↕	⑥	↕
	ASLB													58	2	1	B	•	•	↕	↕	⑥	↕
Desplazamiento a la derecha, aritmético	ASR							67	7	2	77	6	3				M	•	•	↕	↕	⑥	↕
	ASRA													47	2	1	A	•	•	↕	↕	⑥	↕
	ASRB													57	2	1	B	•	•	↕	↕	⑥	↕
Desplazamiento a la derecha, lógico	LSR							64	7	2	74	6	3				M	•	•	R	↕	⑥	↕
	LSRA													44	2	1	A	•	•	R	↕	⑥	↕
	LSRB													54	2	1	B	•	•	R	↕	⑥	↕
Almacena acumulador	STAA				97	4	2	A7	6	2	B7	5	3				A → M	•	•	↕	↕	R	•
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	•	•	↕	↕	R	•
Resta	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A − M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B − M → B	•	•	↕	↕	↕	↕
Resta acumuladores	SBA													10	2	1	A − B → A	•	•	↕	↕	↕	↕
Resta con arrastre	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A − M − C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B − M − C → B	•	•	↕	↕	↕	↕
Transfiere acumuladores	TAB													16	2	1	A → B	•	•	↕	↕	R	•
	TBA													17	2	1	B → A	•	•	↕	↕	R	•
Test, Cero o Menos	TST							6D	7	2	7D	6	3				M − 00	•	•	↕	↕	R	R
	TSTA													4D	2	1	A − 00	•	•	↕	↕	R	R
	TSTB													5D	2	1	B − 00	•	•	↕	↕	R	R
																		H	I	N	Z	V	C

Leyenda:

OP Código de operación (hexadecimal);
~ Números de ciclos de la MPU;
* Número de bytes de programa;
+ Más aritmético;
- Menos aritmético;
· AND booleano;
+ OR booleana inclusiva;
⊕ OR booleana exclusiva;
M Complemento de *M*;
→ Transferir a;
0 Bit = cero;
00 Byte = cero;
M_{SP} Contenido de la posición señalada por el puntero de pila;

Nota - Las instrucciones del modo de direccionamiento del acumulador se incluyen en la columna de direccionamiento implícito.

Figura 10.30. Instrucciones del acumulador y memoria. *(Cortesía de Motorola, Inc.)*

Símbolos de códigos de condición:

H	Arrastre intermedio del bit 3
I	Máscara de interrupción
N	Negativo (bit de signo)
Z	Cero (byte)
V	Overflow, complemento a 2
C	Arrastre del bit 7
R	Reinicialización siempre
S	Pone a 1 siempre
↕	Test y pone a 1 si es verdadero, borra de otra forma
•	No afectado

Notas del registro de códigos de condición:

(Se coloca el bit si el test es cierto y se borra en otro caso)

① (Bit V) Test: ¿resultado = 10000000?
② (Bit C) Test: ¿resultado = 00000000?
③ (Bit C) Test: ¿valor decimal del carácter BCD más significativo es menor que 9? (No borrado si anteriormente a 1)
④ (Bit) V) Test: ¿operando = 10000000 antes de la ejecución?
⑤ (Bit V) Test: ¿operando = 01111111 antes de la ejecución?
⑥ (Bit V) Test: coloca igual al resultado de N ⊕ después de que ha ocurrido el desplazamiento

Figura 10.30. Instrucciones del acumulador y memoria *(continuación).*

la operación de suma. Son ADD *A* y ADD *B*. Observar que la operación ADD *A* puede ser realizada utilizando direccionamiento inmediato, directo, indexado o extendido. El código de op para la instrucción ADD *A* inmediato será 8BH, mientras que ADD *A* directo utiliza el código de op único de 9BH, etc. También se puede determinar de la Figura 10.30 que la operación ADD *A* inmediato se especifica con una instrucción de 2 bytes (notar el 2 bajo el signo #). Además, la operación ADD *A* inmediato utiliza dos ciclos de reloj, o máquina, para su ejecución (observar el 2 bajo el signo ~).

La operación simbólica de la instrucción ADD *A* aparece como $A + M \rightarrow A$ en la Figura 10.30. Esto significa que el contenido del acumulador *A* se suma al contenido de la posición de memoria *M*. El resultado de la operación (la suma en este caso) se coloca en el acumulador *A*, como indica la flecha.

Las columnas derecha de la Figura 10.30 indican que los señalizadores *H*, *N*, *Z*, *V* y *C* son afectados por la operación ADD *A*. Sin embargo, el señalizador *I* no está afectado por esta instrucción. Una leyenda y notas especiales están colocadas al final de la tabla de la Figura 10.30 para explicar muchos de los símbolos.

PROBLEMAS RESUELTOS

10.48. Acudir a la Figura 10.30. La instrucción LDA *B* transfiere el contenido de una posición de _____ al acumulador _____. Los señalizadores _____, _____ y _____ son afectados por la operación LDA *B*.

Solución:

Ver la instrucción LDA *B* de la Figura 10.30. La instrucción LDA *B* transfiere el contenido de una posición de memoria *(M)* al acumulador *B*. Los señalizadores *N*, *Z* y *V* son afectados por la operación LDA *B*. El señalizador *V* está siempre en reinicialización (ver la R de la Figura 10.30) por la operación LDA *B*.

10.49. Acudir a la Figura 10.31. La instrucción LDA *B* de 3 bytes utilizada en este problema tiene el código de op _____ (hex). Esta instrucción utiliza direccionamiento _____.

Solución:

Ver la instrucción LDA *B* en la Figura 10.30. La instrucción LDA *B* utilizada en la Figura 10.31 tiene el código de op F6H. Esta instrucción utiliza direccionamiento extendido.

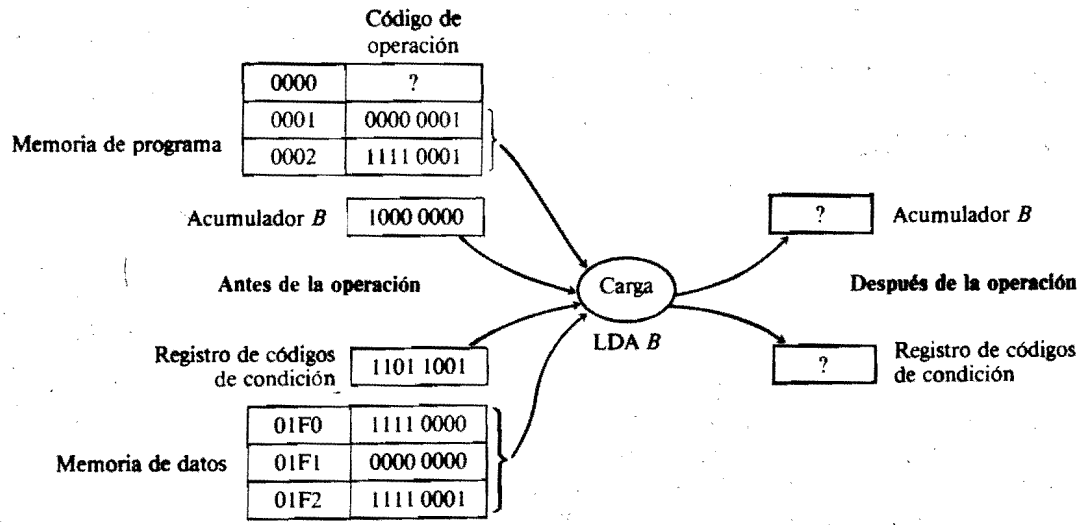


Figura 10.31.

10.50. Acudir a la Figura 10.31. El contenido del acumulador *B* después de la operación de carga será _____ (8 bits).

Solución:

El direccionamiento extendido se utiliza en la instrucción LDA *B* de la Figura 10.31, que significa que el dato en el acumulador *B* de la posición 01F1H de la memoria de datos será copiado. Por tanto, el contenido del acumulador después de la operación de carga será 00000000.

10.51. Acudir a la Figura 10.31. El contenido del registro de códigos de condición después de la operación de carga será _____ (8 bits).

Solución:

Ver la instrucción LDA *B* en la Figura 10.30. El contenido del registro de códigos de condición después de la operación de carga de la Figura 10.31 será 11010101. Solamente los señalizadores *N* y *Z* han sido cambiados durante la operación de carga.

10.52. Acudir a la Figura 10.32. La instrucción «almacenar acumulador *A*» utilizada en este problema utiliza direccionamiento _____ .

Solución:

Ver la instrucción STA *A* de la Figura 10.30. El código de op A7H utilizado en la Figura 10.32 representa la instrucción «almacenar acumulador *A*» y utiliza direccionamiento indexado.

10.53. Acudir a la Figura 10.32. El contenido de la posición de la memoria de datos _____ (hex) será _____ (8 bits) después de la operación de almacenar.

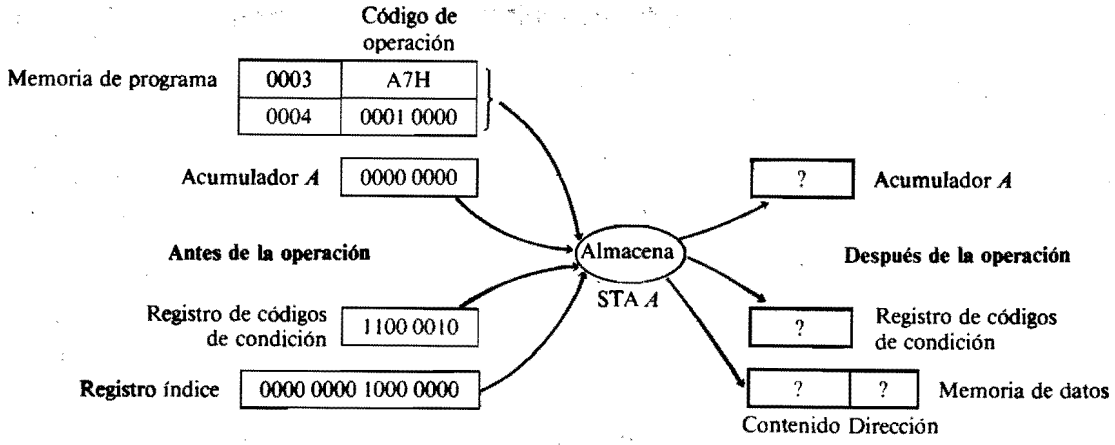


Figura 10.32.

Solución:

Ver la instrucción STA *A* de la Figura 10.30. La posición de la memoria de datos se calcula sumando el contenido del registro índice al desplazamiento (0080 + 10 = 0090H en este ejemplo). Por tanto, el contenido de la memoria de datos 0090H será 00000000 después de la operación de almacenar, en la Figura 10.32.

10.54. Acudir a la Figura 10.32. El contenido del registro de códigos de condición después de la operación de almacenar será _____ (8 bits).

Solución:

Los señalizadores *N*, *Z* y *V* son afectados por la operación STA *A* de acuerdo con la tabla de la Figura 10.30. El contenido del registro de códigos de condición después de la operación de almacenar de la Figura 10.32 será 11000100.

10.55. Acudir a la Figura 10.33. El código de op para la instrucción TST *A* es _____ (hex). Esta instrucción «examinar *A*» utiliza direccionamiento _____ .

Solución:

Ver la instrucción TST *A* de la Figura 10.30. El código de op para la instrucción TST *A* es 4DH. Esta instrucción «examinar *A*» utiliza direccionamiento inherente, o implícito.

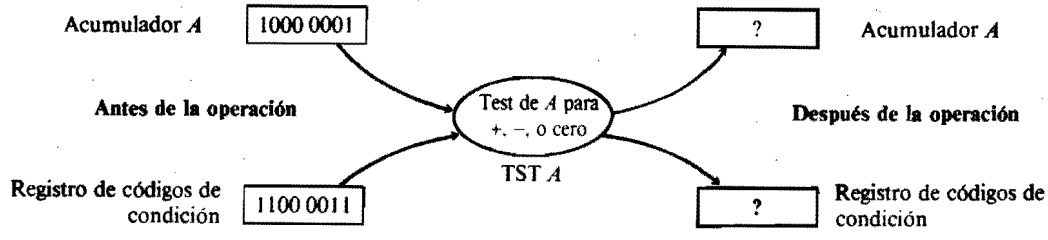


Figura 10.33.

10.56. Acudir a la Figura 10.33. El contenido del acumulador *A* después de la operación TST *A* será _____ (8 bits).

Solución:

La operación de examinar no cambia el valor del acumulador *A*. Por tanto, el contenido del acumulador *A* en la Figura 10.33 después de la operación TST *A* será 10000001.

10.57. Acudir a la Figura 10.33. El contenido del registro de códigos de condición después de la operación TST *A* será _____ (8 bits).

Solución:

De acuerdo con la Figura 10.33, solamente los señalizadores *N*, *Z*, *V* y *C* están afectados por la instrucción TST *A*. El contenido del registro de códigos de condición después de la operación TST *A* será 11001000. Los señalizadores *V* y *C* son siempre puestos a 0 por esta operación. El señalizador *N* se pondrá a 1, indicando que el acumulador *A* contiene un número negativo. El señalizador *Z* estará a 0, indicando que el acumulador *A* no contiene cero.

10.7. INSTRUCCIONES DEL PUNTERO DE PILA Y REGISTRO INDICE DEL 6800

El siguiente grupo de las operaciones del microprocesador 6800 son las *instrucciones del puntero de pila* y *registro índice*, detalladas en la Figura 10.34. Estas incluyen las instrucciones de incrementar y decrementar, cargar y almacenar y transferencia utilizadas con el registro índice y puntero de pila. Recordar que el registro índice, de 16 bits, del 6800 se utiliza cuando se ejecutan las instrucciones en el modo de direccionamiento indexado. El puntero de pila de 16 bits se utiliza para localizar el tope de la pila de la RAM.

OPERACIONES DEL PUNTERO	NEMOTECNICO	MODOS DE DIRECCIONAMIENTO: INMEDIATO OP	~	#	DIRECTO OP	~	#	INDEXADO OP	~	#	EXTENDIDO OP	~	#	IMPLICITO OP	~	#	OPERACION BOOLEANA/ARITMETICA	REGISTRO DE CODIGOS DE CONDICION: 5 H	4 I	3 N	2 Z	1 V	0 C
Compara Reg. Indice	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				$X_H - M, X_L - (M + 1)$	•	•	①	↕	②	•
Decrementa Reg. Indice	DEX													09	4	1	$X - 1 \rightarrow X$	•	•	•	↕	•	•
Decrementa Punt. de pila	DES													34	4	1	$SP - 1 \rightarrow SP$	•	•	•	•	•	•
Incrementa Reg. Indice	INX													08	4	1	$X + 1 \rightarrow X$	•	•	•	↕	•	•
Incrementa Punt. de pila	INS													31	4	1	$SP + 1 \rightarrow SP$	•	•	•	•	•	•
Carga Reg. Indice	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				$M \rightarrow X_H, (M + 1) \rightarrow X_L$	•	•	③	↕	R	•
Carga Punt. de pila	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				$M \rightarrow SP_H, (M + 1) \rightarrow SP_L$	•	•	③	↕	R	•
Almacena Reg. Indice	STX				DF	5	2	EF	7	2	FF	6	3				$X_H \rightarrow M, X_L \rightarrow (M + 1)$	•	•	③	↕	R	•
Almacena Punt. de pila	STS				9F	5	2	AF	7	2	BF	6	3				$SP_H \rightarrow M, SP_L \rightarrow (M + 1)$	•	•	③	↕	R	•
Reg. Indice → Punt. de pila	TXS													35	4	1	$X - 1 \rightarrow SP$	•	•	•	•	•	•
Punt. de pila → Reg. Indice	TSX													30	4	1	$SP + 1 \rightarrow X$	•	•	•	•	•	•

① (Bit N) Test: ¿Es bit de signo del byte más significativo (MS) del resultado = 1?
② (Bit V) Test: ¿Rebose de complemento a 2 de la resta de los bytes ms?
③ (Bit N) Test: ¿Resultado menor que cero?

Figura 10.34. Instrucciones del puntero de pila y registro índice. *(Cortesía de Motorola, Inc.)*

Considerar la instrucción «comparar registro índice» en la primera línea de la tabla de la Figura 10.34. El nemotécnico es CPX. La instrucción CPX puede acceder a memoria utilizando direccionamiento extendido, indexado, directo o inmediato. Las instrucciones de comparación son, esencialmente, operaciones de resta quedando inalteradas las posiciones de memoria y registros. El señalizador Z está a 1 o a 0 y puede ser utilizado para una bifurcación condicional posterior. Los señalizadores N y V también son afectados, pero no están pensados para realizar bifurcaciones condicionales después de esta operación.

PROBLEMAS RESUELTOS

10.58. Acudir a la Figura 10.35. La operación LDX mostrada en este ejemplo utiliza un código de op de _____ (hex). Esta instrucción utiliza direccionamiento _____ .

Solución:

La operación LDX de la Figura 10.35 utiliza un código de op de CEH. Esta instrucción utiliza direccionamiento inmediato.

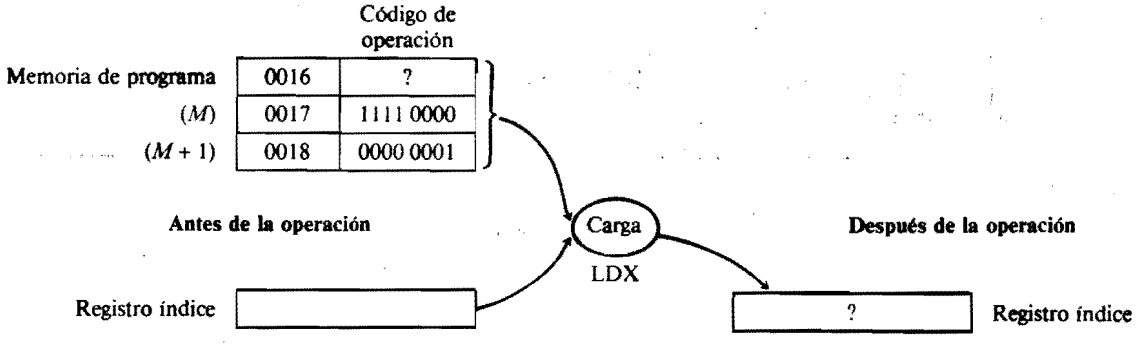


Figura 10.35.

10.59. Acudir a la Figura 10.35. El contenido del registro índice después de la operación de carga será _____ (hex).

Solución:

Ver la instrucción LDX de la Figura 10.34. El contenido del registro índice de la Figura 10.35 después de la operación de carga será F001H.

10.60. Acudir a la Figura 10.36. La posición de la memoria de programa 001CH contendrá el código de op _____ (hex) para especificar la operación «cargar puntero de pila extendido».

Solución:

Ver la instrucción LDS en la Figura 10.34. La posición 001CH del programa de memoria de la Figura 10.36 deberá contener el código de op BEH para especificar la operación «cargar puntero de pila extendido».

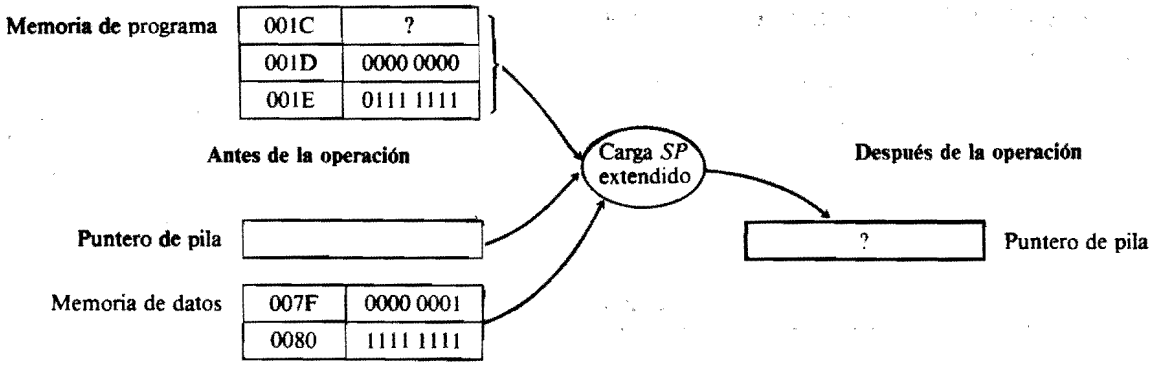


Figura 10.36.

10.61. Acudir a la Figura 10.36. El contenido del **puntero de pila** después de la operación de carga será _____ (hex).

Solución:

Ver la instrucción LDS de la Figura 10.34. El contenido del puntero de pila después de la operación de carga en la Figura 10.36 será 01FFH.

10.8. INSTRUCCIONES DE BIFURCACION Y SALTO DEL 6800

Las *instrucciones de bifurcación* y *salto* para el microprocesador 6800 están detalladas en la Figura 10.37. Estas instrucciones se utilizan para controlar la transferencia de operaciones de un punto a otro del programa. Observar que todas las operaciones de bifurcación utilizan el modo de direccionamiento relativo. La instrucción de «no operación» (NOP) solamente hace avanzar una unidad el contador de programa y no tiene ningún efecto sobre señalizadores, registros o posiciones de memoria.

Considerar la instrucción «saltar a subrutina» (JSR) de la Figura 10.37. La instrucción JSR utiliza direccionamiento indexado o extendido. Detalles de la operación JSR se dan en la Figura 10.38. Las instrucciones «bifurcar a subrutina» (BSR) tienen la misma función que la operación JSR. Sin embargo, utiliza direccionamiento relativo. La instrucción «bifurcar a subrutina» también se explica con mayor detalle en la Figura 10.38.

Las instrucciones de saltar (JMP) y «volver de subrutina» (RTS) se listan en la tabla de la Figura 10.37. Mayor detalle sobre su funcionamiento se da en la Figura 10.38. La «interrupción software» (SWI) y «espera interrupción» (WAI) están también listadas en la tabla de la Figura 10.37. De nuevo, se da mayor detalle sobre estas dos instrucciones en el apartado de interrupciones de la Figura 10.38. Finalmente, la instrucción «vuelta de interrupción» (RTI) se lista y detalla en las Figuras 10.37 y 10.38. La instrucción RTS se utiliza como la última instrucción de una subrutina, mientras que la instrucción RTI realiza el mismo propósito en una rutina de servicio de interrupción.

OPERACIONES	NEMOTECNICO	RELATIVO OP	RELATIVO ~	RELATIVO #	INDEXADO OP	INDEXADO ~	INDEXADO #	EXTENDIDO OP	EXTENDIDO ~	EXTENDIDO #	IMPLICITO OP	IMPLICITO ~	IMPLICITO #	TEST DE BIFURCACION	REGISTRO DE CODIGOS DE CONDICION 5 H	4 I	3 N	2 Z	1 V	0 C
Bifurcación	BRA	20	4	2										Ninguno	•	•	•	•	•	•
Bifurca si arrastre a 0	BCC	24	4	2										C = 0	•	•	•	•	•	•
Bifurca si arrastre a 1	BCS	25	4	2										C = 1	•	•	•	•	•	•
Bifurca si = 0	BEQ	27	4	2										Z = 1	•	•	•	•	•	•
Bifurca si ≥ 0	BGE	2C	4	2										N ⊕ V = 0	•	•	•	•	•	•
Bifurca si > 0	BGT	2E	4	2										Z + (N ⊕ V) = 0	•	•	•	•	•	•
Bifurca si es mayor	BHI	22	4	2										C + Z = 0	•	•	•	•	•	•
Bifurca si ≤ 0	BLE	2F	4	2										Z + (N ⊕ V) = 1	•	•	•	•	•	•
Bifurca si es menor 0 =	BLS	23	4	2										C + Z = 1	•	•	•	•	•	•
Bifurca si < 0	BLT	20	4	2										N ⊕ V = 1	•	•	•	•	•	•
Bifurca si es menos	BMI	2B	4	2										N = 1	•	•	•	•	•	•
Bifurca si no es = a cero	BNE	26	4	2										Z = 0	•	•	•	•	•	•
Bif. si overflow a 0	BVC	28	4	2										V = 0	•	•	•	•	•	•
Bif. si overflow a 1	BVS	29	4	2										V = 1	•	•	•	•	•	•
Bifurca si es más	BPL	2A	4	2										N = 0	•	•	•	•	•	•
Bifurca a subrutina	BSR	8D	8	2											•	•	•	•	•	•
Salta	JMP				6E	4	2	7E	3	3				Ver operaciones especiales	•	•	•	•	•	•
Salta a subrutina	JSR				AD	8	2	BD	9	3					•	•	•	•	•	•
No operación	NOP										01	2	1	Sólo avanza el control del programa	•	•	•	•	•	•
Vuelta de interrupción	RTI										3B	10	1		—	—	—	①	—	—
Vuelta de subrutina	RTS										39	5	1		•	•	•	•	•	•
Interrupcion software	SWI										3F	12	1	Ver operaciones especiales	•	•	•	•	•	•
Espera la interrupción	WAI										3E	9	1		•	②	•	•	•	•

* WAI pone el bus de direcciones R/W y el bus de datos en modo de tres estados mientras VMA está en el nivel bajo.

① (All) Carga el registro de códigos de condición de la pila. (Ver operaciones especiales).
② (Bit 1) Se pone a 1 cuando la interrupción ocurre. Si está previamente a 1, se requiere una interrupción no enmascarable para sacarlo del estado de espera.

Figura 10.37. Instrucciones de bifurcación y salto. *(Cortesía de Motorola, Inc.)*

WAI es una instrucción utilizada comúnmente. Se utiliza al final de un programa con más frecuencia que la instrucción de alto (HLT) en la MPU 8080/8085. De acuerdo con la Figura 10.38, la instrucción WAI coloca primero el contenido de los registros del procesador en la pila. Después introduce un *ciclo de espera* «de no hacer nada». La MPU 6800 continúa en este lazo hasta que el procesador recibe una interrupción hardware (*IRQ, NMI, REINICIALIZACION*).

PROBLEMAS RESUELTOS

10.62. Todas las instrucciones de bifurcación de la MPU 6800 utilizan el modo de direccionamiento _____ .

Solución:

De acuerdo con la Figura 10.37, todas las instrucciones de bifurcación de la MPU 6800 utilizan el modo de direccionamiento relativo.

10.63. Todas las operaciones de bifurcación de la MPU 6800 están especificadas con instrucciones de _____ (número) bytes. El primer byte de la instrucción siempre es _____ _____ . El segundo byte de la instrucción se denomina _____ .

Solución:

De acuerdo con la Figura 10.37, todas las operaciones de bifurcación de la MPU 6800 se especifican con instrucciones de 2 bytes. El primer byte de la instrucción siempre es el código de op. El segundo byte de la instrucción se denomina desplazamiento (offset).

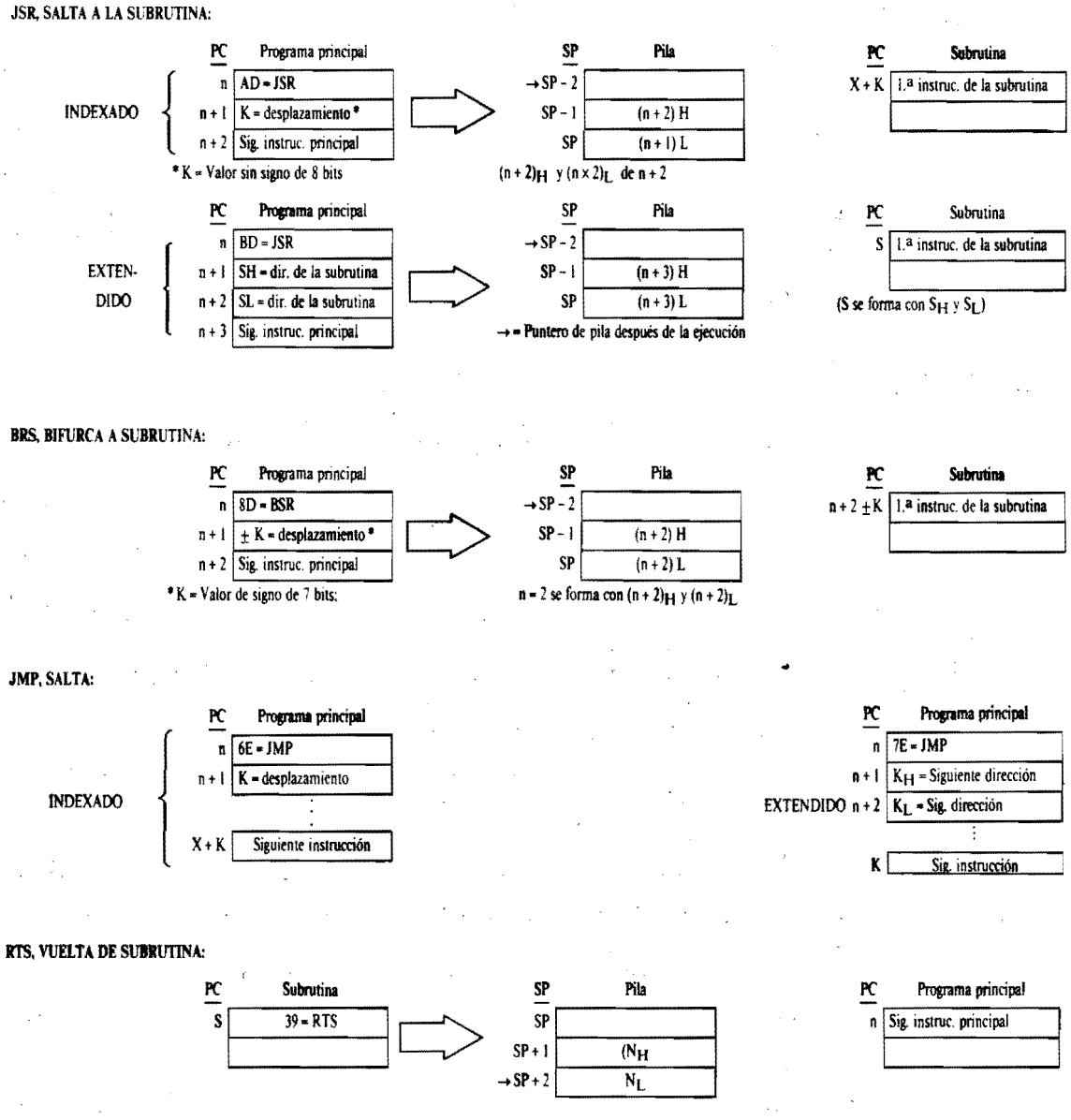


Figura 10.38. Explicación de las operaciones especiales de bifurcación y salto.

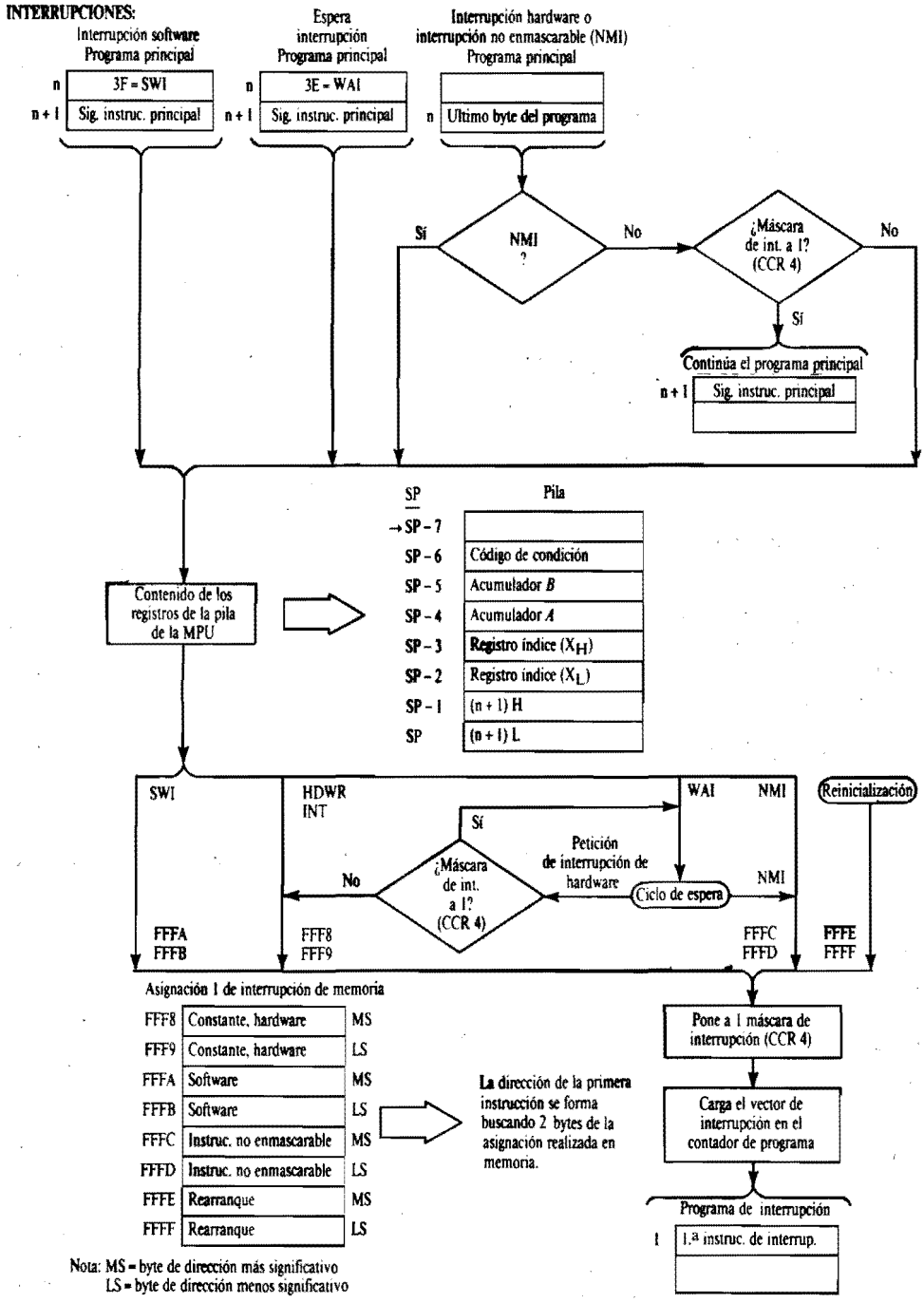


Figura 10.38. *(continuación).*

RTI, VUELTA DE INTERRUPCION:

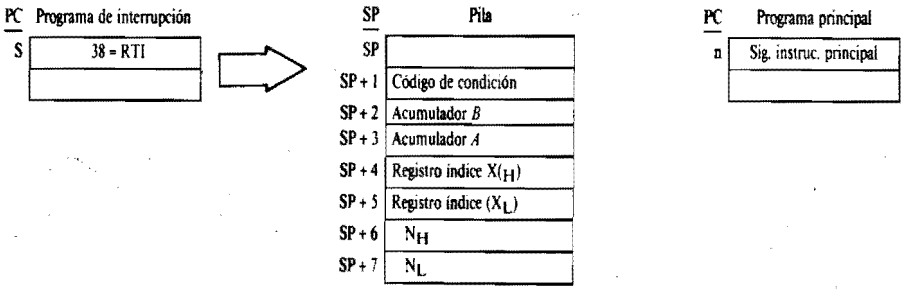


Figura 10.38. *(continuación).*

10.64. Acudir a la Figura 10.39. El código de op para la operación «bifurca si cero» es _____ (hex). El contenido de la posición de memoria del programa 0028H es igual _____ (a la dirección, al desplazamiento), que es _____ (−, +) 7H.

Solución:

Ver la instrucción BEQ de la tabla de la Figura 10.37. El código de op para la operación «bifurca si cero» de la Figura 10.39 es 27H. El contenido de la posición 0028H de la memoria de programa es igual al desplazamiento, que es −7H. El desplazamiento de −7H sugiere una bifurcación hacia atrás.

10.65. Acudir a la Figura 10.39. La condición de bifurcación es _____ (falsa, cierta) en este problema.

Solución:

La condición de bifurcación para la instrucción BEQ es bifurcar si $Z = 1$ de acuerdo con la tabla de la Figura 10.37. Por tanto, la condición de bifurcación es cierta en el problema de la Figura 10.39.

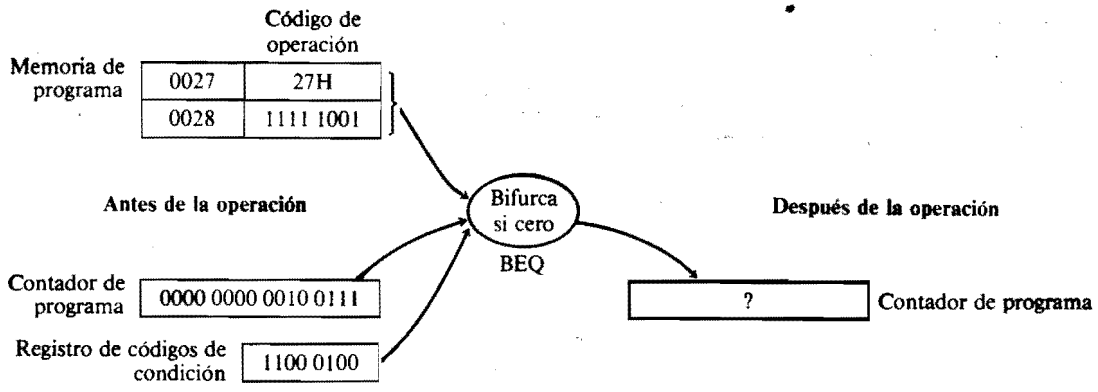


Figura 10.39.

10.66. Acudir a la Figura 10.39. El contenido del contador de programa después de la operación «bifurca si cero» será _____ (hex).

Solución:

La condición de bifurcación es verdadera en la Figura 10.39. El desplazamiento −7H dicta una bifurcación hacia atrás. Por tanto, el contenido del contador de programa después de la operación «bifurca si cero» de la Figura 10.39 será 0022H [(0027 + 2) −7 = 0022H].

10.67. Acudir a la Figura 10.40. La operación de «saltar a subrutina» en este problema está especificada por el código de op _____ (hex). Esta operación utiliza direccionamiento _____.

Solución:

Ver la instrucción JSR de la Figura 10.37. La operación de «saltar a subrutina» de la Figura 10.40 está especificada por el código de op BDH. Esta operación utiliza direccionamiento extendido.

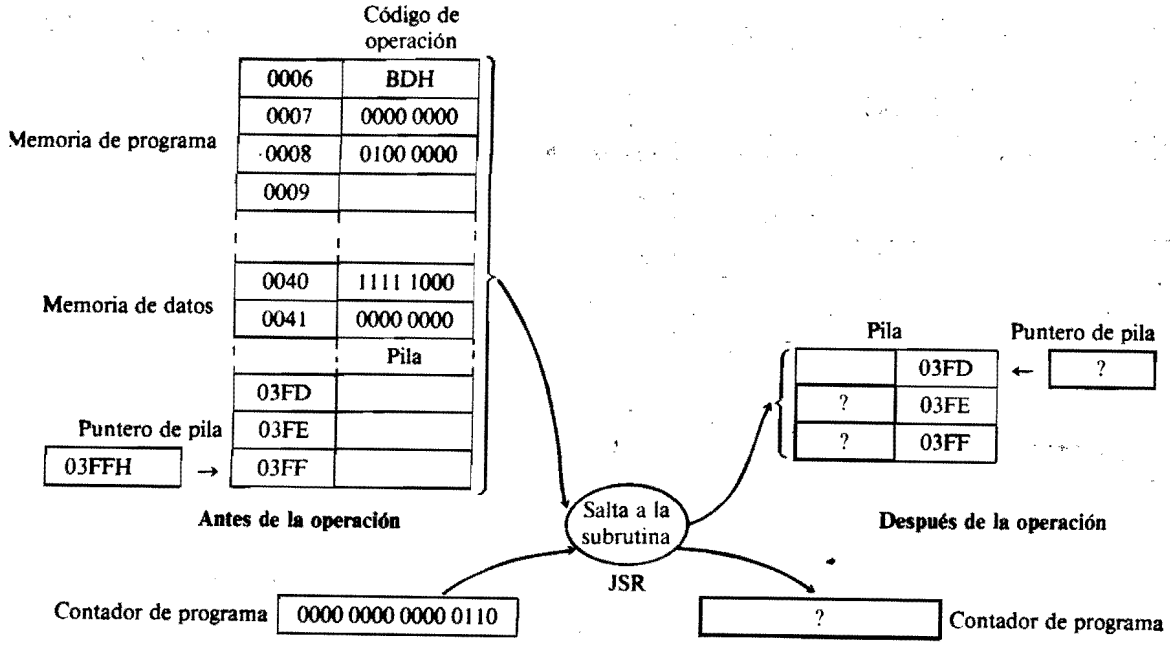


Figura 10.40.

10.68. Acudir a la Figura 10.40. Listar los contenidos del puntero de pila y las posiciones de la pila 03FEH y 03FFH después de la operación JSR.

Solución:

Ver los detalles de «JSR extendido» en la Figura 10.38. Después de la operación «JSR extendido» mostrada en la Figura 10.40, los contenidos son:
Puntero de pila = 03FDH
Posición de pila 03FEH = 00000000 (PC_H)
Posición de pila 03FFH = 00001001 (PC_L) ($PC + 3$)

10.69. Acudir a la Figura 10.40. El contenido del contador de programa después de la operación JSR será _____ (hex). Esta es la dirección de _____ .

Solución:

Ver los detalles de «JSR extendido» de la Figura 10.38. El contenido del contador de programa de la Figura 10.40 después de la operación JSR será F800H. Esta es la dirección de la subrutina.

10.70. Acudir a la Figura 10.41. El código de op para la instrucción SWI es _____ (hex). SWI significa la operación _____ _____ .

Solución:

Ver la instrucción SWI de la Figura 10.37. El código de op para la instrucción SWI de la Figura 10.41 es 3FH. SWI significa operación de interrupción software.

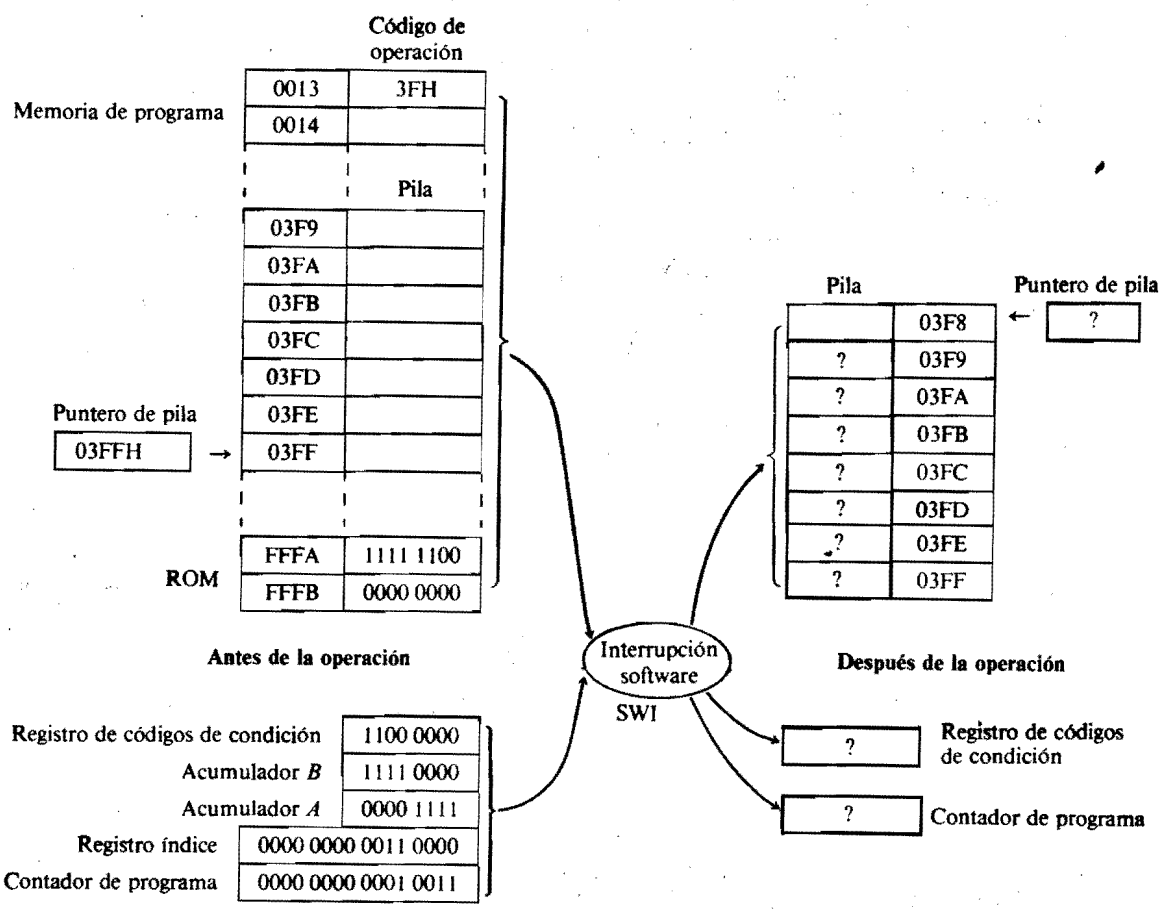


Figura 10.41.

10.71. Acudir a la Figura 10.41. Listar el contenido del puntero de pila y pila después de la operación de interrupción software.

Solución:

Después de la operación SWI de la Figura 10.41, el contenido del puntero de pila es 03F8H, y el contenido de la pila es:

Posición de pila (hex)		Contenido
03F9	11000000	(Registro de códigos de condición)
03FA	11100000	(Acumulador *B*)
03FB	00001111	(Acumulador *A*)
03FC	00000000	(Registro índice – byte superior)
03FD	00110000	(Registro índice – byte inferior)
03FE	00000000	(*PC* – byte superior)
03FF	00010100	(*PC* – byte inferior) (*PC* + 1)

10.72. Acudir a la Figura 10.41. El contenido del registro de código de condición después de la operación SWI será _____ (8 bits).

Solución:

Ver la secuencia de interrupción software mostrada en la Figura 10.38. El señalizador *I* estará a 1. Por tanto, el contenido del registro de código de condición de la Figura 10.41 después de la operación SWI será 11010000.

10.73. Acudir a la Figura 10.41. El contenido del contador de programa después de la operación SWI será _____ (hex). Esta dirección se denomina interrupción software _____ y se encontró en las posiciones _____ (hex) y _____ (hex) de la ROM.

Solución:

Ver la secuencia de interrupciones software mostradas en la Figura 10.38. El contenido del contador de programa después de la operación SWI será FC00H. Esta dirección es llamada vector de interrupción software y se encuentra en las posiciones FFFAH y FFFBH de la ROM. Estas son las posiciones estándar para encontrar el vector de interrupción software en un sistema 6800.

10.9. INSTRUCCIONES DEL REGISTRO DE CODIGOS DE CONDICION DEL 6800

Las *instrucciones del registro de códigos de condición* para el microprocesador Motorola 6800 se muestran en la Figura 10.42*a*. Estas instrucciones están disponibles para que el usuario realice manipulaciones *directas* de los señalizadores del registro de códigos de condición (*CCR*). Por conveniencia, la posición y nombres de cada uno de los señalizadores en el *CCR* están reproducidas en la Figura 10.42*b*. Algunos estudiantes recuerdan el orden de los señalizadores del 6800 recordando las palabras «HINZ Vitamina C».

OPERACIONES	NEMOTECNICO	IMPLICITO OP	~	#	OPERACION BOOLEANA	REGISTRO DE CODIGOS DE CONDICION 5 H	4 I	3 N	2 Z	1 V	0 C
Borra arrastre	CLC	0C	2	1	0 → C	•	•	•	•	•	R
Borra masc. de interrup.	CLI	0E	2	1	0 → I	•	R	•	•	•	•
Borra overflow	CLV	0A	2	1	0 → V	•	•	•	•	R	•
Pone a 1 arrastre	SEC	0D	2	1	1 → C	•	•	•	•	•	S
Pone a 1 masc. de interrup.	SEI	0F	2	1	1 → I	•	S	•	•	•	•
Pone overflow	SEV	0B	2	1	1 → V	•	•	•	•	S	•
Acumulador *A* → *CCR*	TAP	06	2	1	A → CCR	①					
CCR → *A*	TPA	07	2	1	CCR → A	•	•	•	•	•	•

① (Todos) ponen a 1 según el contenido del acumulador *A*.

LEYENDA

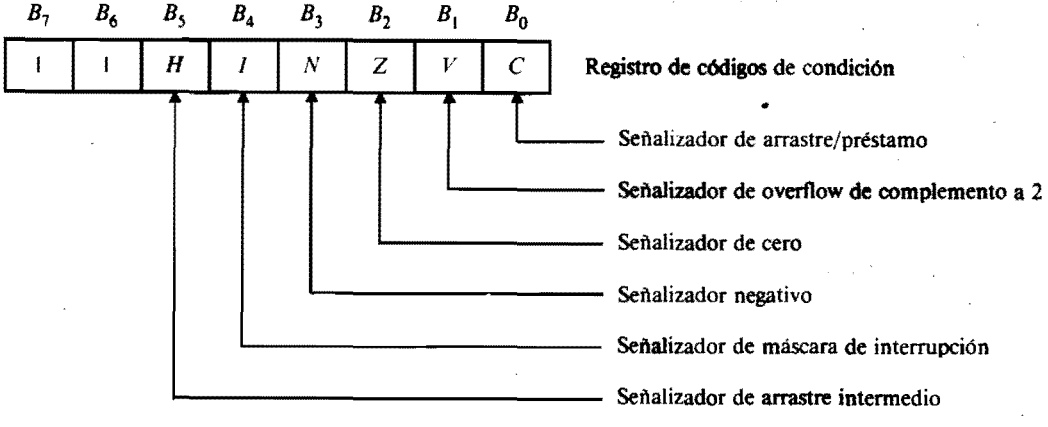
OP	Código de operación (hexadecimal);	+	OR inclusiva booleana;
~	Número de ciclos de la MPU;	⊕	OR exclusiva booleana;
#	Número de bytes del programa;	M	Complemento de *M*;
+	Más aritmético;	→	Transfiere a;
–	Menos aritmético;	0	Bit = 0
•	AND booleano;	00	Byte = 0;
M_{SP}	Contenido de la posición de memoria apuntada por el puntero de pila;		

SIMBOLOS DE CODIGOS DE CONDICION

H	Arrastre intermedio del bit 3;
I	Máscara de interrupción;
N	Negativo (bit de signo)
Z	Cero (byte):
V	Overflow, complemento a 2
C	Arrastre del bit 7;
R	Reinicialización siempre
S	Pone a 1 siempre
↕	Test y pone a 1 si es verdad, borra de otra forma
•	No se afecta

Nota: En las instrucciones del modo de direccionamiento del acumulador se incluye la columna de direccionamiento implícito

(a) Instrucciones del registro de código de condición



(b) Señalizadores en el registro de código de condición

Figura 10.42. *(Cortesía de Motorola, Inc.).*

PROBLEMAS RESUELTOS

10.74. Acudir a la Figura 10.43. La operación TAP transfiere el contenido del acumulador _____ (*A*, *B*) al _____ _____ _____ .

Solución:

Ver la instrucción TAP de la Figura 10.42*a*. La operación TAP transfiere el contenido del acumulador *A* al registro de códigos de condición.

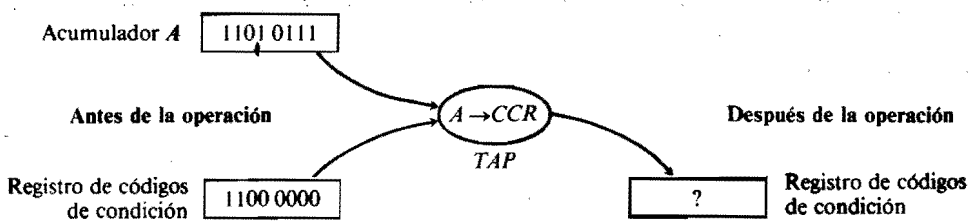


Figura 10.43.

10.75. Acudir a la Figura 10.43. Listar las condiciones de los señalizadores del *CCR* después de la operación TAP.

Solución:

Las condiciones de los señalizadores después de la operación TAP en la Figura 10.43 son:

Señalizador $H = 0$	Señalizador $N = 0$	Señalizador $V = 1$
Señalizador $I = 1$	Señalizador $Z = 1$	Señalizador $C = 1$

10.76. Listar los nemotécnicos de las instrucciones que se utilizan para poner a 1 y a 0 el señalizador de la máscara de interrupción.

Solución:

Ver la Figura 10.42*a*. Los nemotécnicos utilizados para poner a 1 y a 0 el señalizador de la máscara de interrupción (*I*) son SEI y CLI.

10.77. La instrucción «borrar arrastre» utiliza el nemotécnico _____, el código de op _____ (hex), direccionamiento _____ y _____ (número) de ciclos de reloj para su ejecución. Pone a _____ (0, 1) el señalizador de _____ en el _____ _____ _____ .

Solución:

Ver la instrucción «borrar arrastre» de la Figura 10.42. La instrucción «borrar arrastre» utiliza el nemotécnico CLC, el código de op 0CH y direccionamiento implícito (o inherente) y dos ciclos de reloj para su ejecución. Pone a 0 el señalizador *C* en el registro de códigos de condición.

PROBLEMAS SUPLEMENTARIOS

10.78. El microprocesador 6800 fue introducido por Motorola en _____ (1968, 1974).

Res. 1974.

10.79. Un sistema microcomputador 6800 mínimo debe incluir la MPU 6800, reloj, fuente de alimentación, conmutador de rearranque, memorias _____ y _____ y una _____ 6820 para realizar la interfaz con los dispositivos de E/S.

Res. RAM, ROM, PIA (adaptador de interfaz de periféricos).

10.80. Las patillas $\overline{REINICIALIZACION}$, $\overline{IRQ}$ y $\overline{NMI}$ de la MPU 6800 son _____ (entradas, salidas), y se clasifican como _____ hardware.

Res. entradas, interrupciones.

10.81. La patilla $\overline{HALT}$ de la MPU 6800 es una _____ (entrada, salida). Cuando la patilla $\overline{HALT}$ está activada por un nivel _____ (ALTO, BAJO), el procesador entra en el modo de alto.

Res. entrada, BAJO.

10.82. La $R/\overline{W}$ de la MPU 6800 es una _____ (entrada, salida). Cuando la patilla $R/\overline{W}$ va al nivel BAJO, es una señal de que el procesador está _____ (leyendo, escribiendo).

Res. salida, escribiendo.

10.83. Listar los tres registros de 16 bits del microprocesador 6800.

Res. contador de programa, puntero de pila, registro índice.

10.84. Listar tres registros de 8 bits del 6800 que puedan ser accedidos por el programador.

Res. acumulador *A*, acumulador *B*, registro de códigos de condición.

10.85. La MPU 6800 utiliza E/S de _____ - _____ ya que las entradas y salidas son tratadas igual que posiciones de memoria. La MPU 6800 _____ (utiliza, no utiliza) instrucciones especiales IN y OUT para operaciones de E/S.

Res. mapa de memoria, no utiliza.

10.86. En los sistemas basados en la MPU 6800, la _____ (RAM, ROM) está habitualmente localizada en las direcciones inferiores comenzando en 0000H, mientras que la _____ (RAM, ROM) está localizada en las direcciones superiores finalizando en FFFFH.

Res. RAM, ROM.

10.87. Los vectores de interrupción están localizados en las posiciones de memoria _____ (más altas, más bajas) en los sistemas basados en el 6800.

Res. más altas (localizadas en FFF8 a FFFFH).

10.88. Solamente es enmascarable la interrupción hardware _____ ($\overline{IRQ}$, $\overline{NMI}$, $\overline{REINICIALIZACION}$).

Res. $\overline{IRQ}$.

10.89. La secuencia de interrupción de reinicialización detallada en la Figura 10.8*a* hace que el señalizador _____ se ponga a 1, y que el vector de interrupción de reinicialización se encuentre en las posiciones de memoria _____ (hex) y _____ (hex).

Res. *I*, FFFEH, FFFFH.

10.90. De acuerdo con la Figura 10.9, las posiciones de la ROM FFF8H y FFF9H en los sistemas basados en el 6800 están siempre reservadas para el _____ _____ _____.

Res. vector de petición de interrupción (dirección).

10.91. Listar los seis modos de direccionamiento utilizados por las instrucciones de la MPU 6800.

Res. inherente (o implícito), inmediato, directo, extendido, indexado y relativo.

10.92. En el direccionamiento inmediato, el operando se encuentra en la memoria de _____ (datos, programa).

Res. programa.

10.93. Las operaciones de direccionamiento directo del microprocesador 6800 se especifican por medio de instrucciones de _____ (número) bytes.

Res. 2.

10.94. Las operaciones de direccionamiento extendido del microprocesador 6800 se especifican por medio de instrucciones de _____ (número) bytes.

Res. 3.

10.95. En el direccionamiento indexado del 6800, la dirección del operando se calcula sumando el valor del _____ encontrado en el segundo byte de la instrucción al contenido del registro _____.

Res. desplazamiento, índice.

10.96. En el direccionamiento relativo del 6800, la dirección de destino de la bifurcación se calcula sumando _____ (número) al contenido del contador de programa y sumando después el valor del _____ de la memoria del programa.

Res. 2, desplazamiento.

10.97. En el direccionamiento relativo, un desplazamiento de 11111100 indica una bifurcación _____ (hacia atrás, hacia adelante).

Res. hacia atrás (MSB = 1).

10.98. La instrucción AND *A* de la Figura 10.26 puede ser utilizada en uno de _____ (número) modos de direccionamiento. La instrucción AND *A* realiza la operación AND del contenido de _____ _____ con el contenido de una posición _____. El resultado de la operación se coloca en _____ _____.

Res. cuatro, acumulador *A*, de memoria *(M)*, acumulador *A*.

10.99. Aunque todas las instrucciones de bifurcación utilizan direccionamiento _____, las instrucciones de salto utilizan direccionamiento _____ y _____.

Res. relativo, indexado, extendido.

10.100. La instrucción COM *B* de la Figura 10.26 hace que el contenido del acumulador _____ sea cambiado a su forma en complemento a _____, mientras que la instrucción NEG *B* hace que el contenido de _____ _____ se cambie a su forma en _____ _____.

Res. *B*, 1, acumulador *B*, complemento a 2.

10.101. Acudir a la Figura 10.44. El código de op para la instrucción ROL *A* es _____ (hex).

Res. 49H.

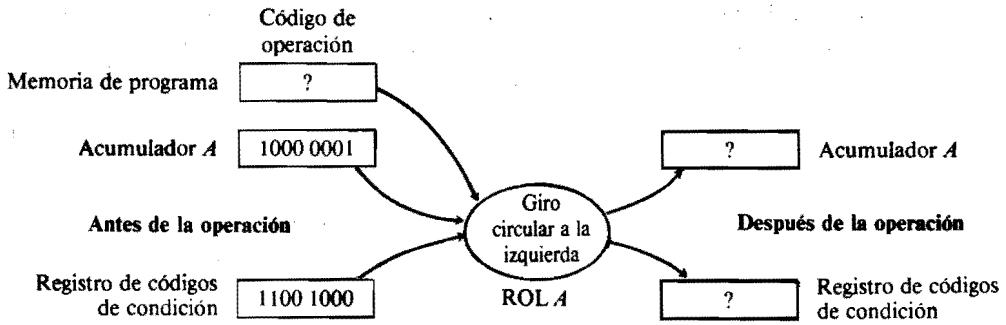


Figura 10.44.

10.102. Acudir a la Figura 10.44. El contenido del acumulador *A* después de la operación ROL *A* será _____ (8 bits).

Res. 00000010.

10.103. Acudir a la Figura 10.44. El contenido del registro de códigos de condición después de la operación ROL *A* será _____ (8 bits).

Res. 11000011.

10.104. Acudir a la Figura 10.45. El código de op para la instrucción SBA _____ (hex). Esta instrucción de 1 byte utiliza direccionamiento _____.

Res. 10H, inherente (o implícito).

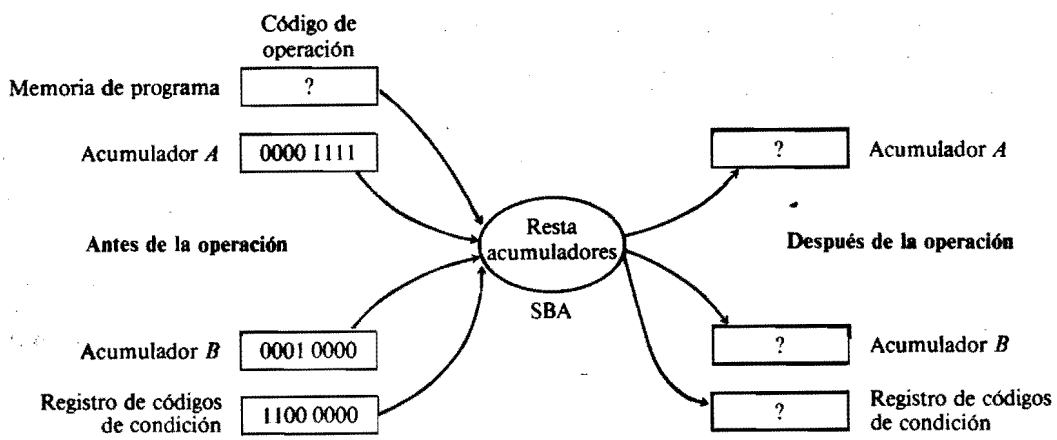


Figura 10.45.

10.105. Acudir a la Figura 10.45. El contenido del acumulador *A* después de la operación SBA es _____ (8 bits).

Res. 11111111 (representación de −1 en complemento a 2).

10.106. Acudir a la Figura 10.45. El contenido de *CCR* después de la operación «restar acumuladores» es de _____ (8 bits).

Res. 11001001 (cambiados señalizadores *N* y *C*).

10.107. La instrucción de «vuelta de subrutina» está especificada por el código de op _____ (hex) o por el nemotécnico _____. Esta instrucción se utiliza al final de todas las _____.

Res. 39H, RTS, subrutinas.

10.108. Acudir a la Figura 10.46. La operación «bifurca si más» está especificada por el código de op 2AH o por el nemotécnico _____. La operación de bifurcación utiliza direccionamiento _____.

Res. BPL, relativo.

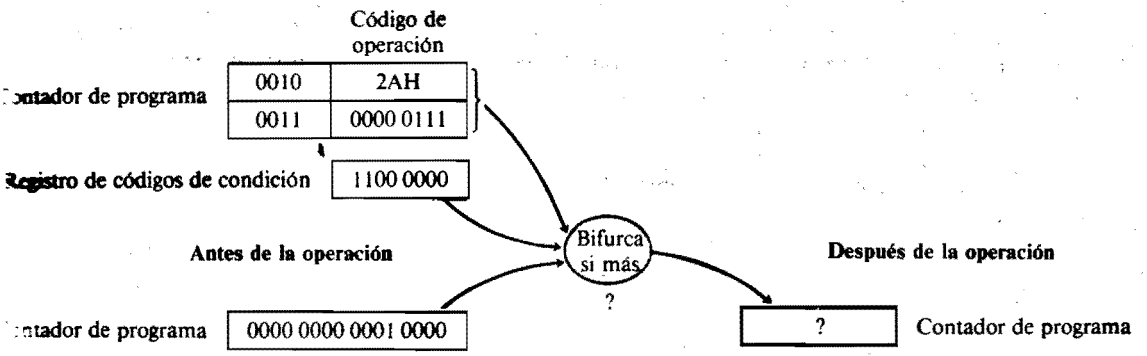


Figura 10.46.

10.109. Acudir a la Figura 10.46. La condición «bifurca si más» es _____ (falsa, cierta), según los señalizadores de condición.

Res. cierta ($N = 0$).

10.110. Acudir a la Figura 10.46. El contenido del contador de programa después de la operación «bifurca si más» _____ (hex).

Res. 0019H [(0010 + 2) + 7 = 0019H].

10.111. El desplazamiento en la instrucción indexada es un valor _____ de 8 bits. Sin embargo, el desplazamiento en la instrucción relativa es un valor con signo de _____ (número) bits.

Res. sin signo, 7.

10.112. La instrucción WAI de la Figura 10.38 introduce todos los registros del procesador en la _____ y después entra en el ciclo de _____.

Res. pila, espera.

10.113. Acudir a la Figura 10.47. La instrucción CLI se especifica por el código de op _____ (hex).

Res. 0EH.

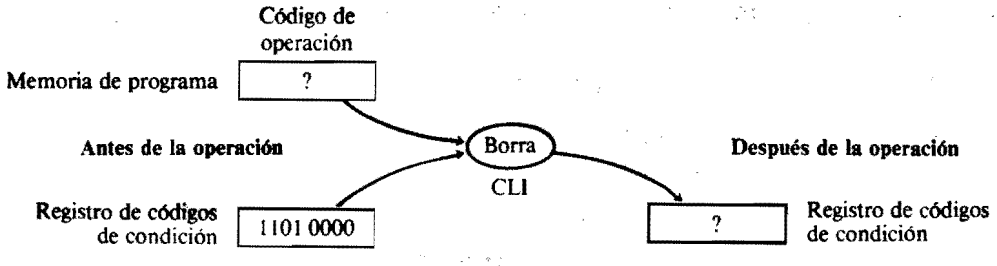


Figura 10.47.

10.114. Acudir a la Figura 10.47. La instrucción CLI especifica la operación de «borrar _____ _____». Pone a _____ (0, 1) el señalizador *I* del registro de códigos de condición.

Res. máscara de interrupción, 0.

10.115. Acudir a la Figura 10.47. El contenido del registro de códigos de condición después de la operación CLI es _____ (8 bits).

Res. 11000000 (señalizador *I* borrado).