7

TRANSFERENCIA DE REGISTROS Y OPERACIONES DE LA COMPUTADORA

7-1 INTRODUCCION

Un sistema digital es un sistema de lógica secuencial construido con flip-flops o multivibradores biestables y compuertas. En el capítulo 4 se demostró que los circuitos secuenciales se pueden especificar por medio de tablas de estados. Especificar un sistema digital grande con tablas de estados es muy difícil, si no imposible, porque el número de estados sería prohibitivamente grande. Para superar esta dificultad, los sistemas digitales se diseñan utilizando un enfoque modular. El sistema se divide en subsistemas modulares y cada uno de ellos realiza alguna tarea funcional. Los módulos se construyen a partir de dispositivos digitales como registros, contadores, decodificadores, multiplexores, elementos aritméticos y lógica de control. Los diversos módulos se interconectan a través de trayectorias de datos y de control comunes para formar el sistema de computación digital.

Los módulos digitales se definen de manera óptima por medio de los registros que contienen y las operaciones que se realizan en la información binaria almacenada en ellos. Ejemplos de operaciones de registros son corrimiento, conteo, puesta a ceros y carga. En esta configuración, se supone que los registros son los componentes básicos del sistema digital, y el flujo de información y las tareas de procesamiento entre los datos almacenados en los registros reciben el nombre de operaciones de *transferencia de registros*. Tales operaciones de transferencia de registros de los sistemas digitales se especifican por medio de los tres componentes básicos siguientes:

1. El conjunto de registros del sistema y su función.
2. Las operaciones que se efectúan con la información almacenada en los registros.
3. El control que supervisa la secuencia de operaciones en el sistema.

Un *registro*, según la notación de transferencia de registros, es un grupo de multivibradores biestables que almacenan información binaria y tienen la posibilidad de realizar una o más operaciones elementales. Un registro puede cargar información nueva o bien correr la información a la derecha o a la izquierda. Un contador se considera como un registro que efectúa la operación de incremento en uno. Un flip-flop solitario se considera como un registro de un bit que se puede iniciar, poner en ceros o complementar. De hecho, los flip-flops y compuertas asociadas de cualquier circuito secuencial reciben el nombre de registro por este método de designación.

Las operaciones que se realizan con la información almacenada en registros se denominan *microoperaciones*, esto es, una operación elemental que se puede realizar en paralelo en una cadena de bits durante un periodo de pulsos de reloj. El resultado de la operación puede reemplazar la información binaria previa en un registro, o bien puede transferirse a otro registro. Las funciones digitales que se presentaron en el capítulo 5 son registros que ejecutan microoperaciones. Un contador con carga en paralelo puede realizar las microoperaciones de incremento y carga. Un registro de corrimiento bidireccional puede efectuar las microoperaciones de corrimiento a la derecha y a la izquierda.

El control que inicia la secuencia de operaciones consta de señales de sincronización que dan sucesión a las operaciones en la forma prescrita. Ciertas condiciones que dependen de resultados de operaciones previas pueden determinar la secuencia de operaciones a futuro. Las salidas de la lógica de control son variables binarias que inician las diversas microoperaciones en los registros.

En este capítulo se presentan los componentes de la transferencia de registros con una notación simbólica para representar registros y especificar las operaciones con el contenido de los registros. El método de transferencia de registros se vale de un conjunto de expresiones e instrucciones que se parecen a las instrucciones utilizadas en los lenguajes de programación. Esta notación proporciona las herramientas necesarias para especificar el conjunto de interconexiones prescrito entre diversas funciones digitales.

En vez de tener registros individuales que efectúen las microoperaciones directamente, los sistemas de computación emplean un número de registros de almacenamiento junto con una unidad operacional común llamada *unidad de aritmética y lógica,* que se abrevia como ALU. Para efectuar una microoperación, el contenido de los registros especificados se coloca en las entradas de la ALU común. La ALU realiza una operación y, después, su resultado se transfiere a un registro destino. La ALU es un circuito combinatorio, por lo que toda la operación de transferencia de registros —de los registros fuente a la ALU y al registro destino— se puede efectuar durante un periodo de pulsos del reloj. Las microoperaciones de corrimiento se realizan a menudo en una unidad por separado. Esta unidad de corrimiento suele mostrarse por separado, pero a veces se considera como parte de la unidad de aritmética y lógica integral.

A un grupo de registros conectados a una ALU común se le conoce como *unidad procesadora*, que es la parte de una computadora digital que ejecuta las operaciones de procesamiento de datos en el sistema. La unidad procesadora, cuando se combina con una unidad de control que supervisa la secuencia de operaciones, recibe el nombre de *unidad central de procesamiento*, que se abrevia como CPU. En la segunda parte de este capítulo se analizan la organización y el diseño de la unidad procesado-

ra. El diseño de una unidad de aritmética y lógica en particular se emprende con el fin de mostrar el proceso de diseño que implica la construcción de un circuito digital complejo. En el capítulo siguiente se estudia la organización y el diseño de la unidad de control. En el capítulo 10 se muestra el diseño detallado de una unidad central de procesamiento.

7-2 TRANSFERENCIA DE REGISTROS

Los registros de un sistema digital se designan por letras mayúsculas (a veces seguidas de numerales) que denotan la función del registro. Por ejemplo, el registro que contiene una dirección de la unidad de memoria suele denominarse registro de dirección de la memoria y está designado por el nombre *AR*. Otras designaciones de registros son *PC*, *IR*, *R*1 y *R*2. Los flip-flops individuales de un registro de *n* bits se numeran en secuencia de 0 a $n - 1$, comenzando desde 0 en la última posición a la derecha y aumentando hacia la izquierda. La figura 7-1 ilustra la representación de registros en forma de diagrama de bloque. La forma más común de representar un registro es a través de una figura rectangular con el nombre del registro en el interior, como se ve en la figura 7-1(a). Los bits individuales pueden distinguirse como en (b). La numeración de bits en un registro de 16 bits se puede marcar en la parte superior de la figura, como se indica en (c). Un registro de 16 bits se divide en dos partes en (d). A los bits del 0 al 7 se les asigna el símbolo *L* (de byte bajo, low) y a los bits del 8 al 15 se les asigna el símbolo *H* (de byte alto, high). El nombre del registro de 16 bits es *PC*. El símbolo *PC*(0-7) o *PC*(*L*) se refiere al byte de bajo orden y *PC*(8-15) o bien *PC*(*H*) al byte de alto orden.

La transferencia de información de un registro a otro se designa en forma simbólica por medio de un operador de reemplazo. La instrucción

$$R2 \leftarrow R1$$

denota una transferencia del contenido del registro *R*1 al registro *R*2; es decir, designa un reemplazo del contenido de *R*2 por el contenido de *R*1. Por definición, el contenido del registro fuente *R*1 no sufre alteración después de la transferencia.

Una instrucción que especifica una transferencia de registros implica que se dispone de circuitos de las salidas del registro fuente a las entradas del registro destino y que el registro destino tiene capacidad de carga en paralelo. Normalmente, no desea-

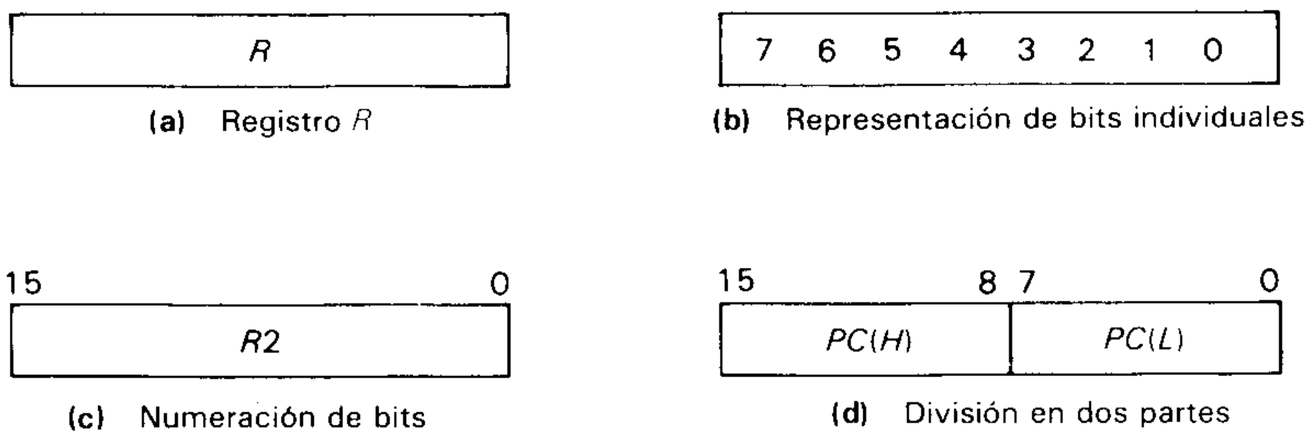
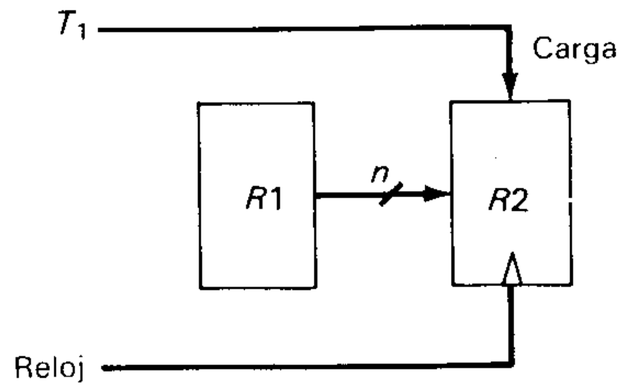
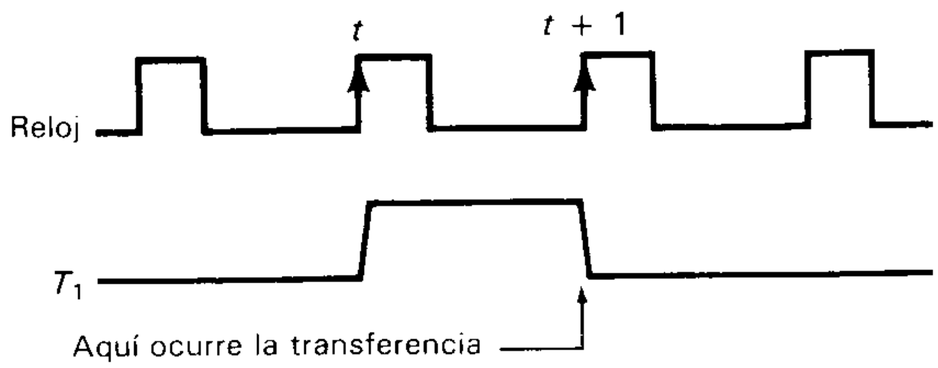


FIGURA 7—1
Diagrama de bloque de un registro



(a) Diagrama de bloques



(b) Diagrama de sincronización

FIGURA 7—2
Transferencia de $R1$ a $R2$ cuando $T_1 = 1$

mos que la transferencia ocurra con cada pulso del reloj, sino sólo en una condición predeterminada. Una instrucción condicional se simboliza con una instrucción *if-then* (si-entonces)

$$Si\ (T_1 = 1)\ entonces (R2 \leftarrow R1)$$

donde T_1 es una señal de sincronización generada en la sección de control. A veces conviene separar las variables de control de la operación de transferencia de registros especificando una *función de control,* que es una variable booleana que puede ser igual a 1 o a 0. La función de control se incluye en la instrucción de la manera siguiente:

$$T_1\colon R2 \leftarrow R1$$

La condición de control se termina con dos puntos y simboliza el requisito de que la operación de transferencia sea ejecutada por el *hardware* sólo si $T_1 = 1$.

Todas y cada una de las instrucciones que se escriben en una notación de transferencia de registros implican una construcción de *hardware* para ejecutar la transferencia. La figura 7-2 ilustra el diagrama de bloque que representa la transferencia de $R1$ a $R2$. Las n salidas del registro $R1$ se conectan a las n entradas del registro $R2$. La letra n se utilizará para indicar cualquier número de bits para el registro, y se reemplazará por un número real cuando se conozca la longitud del registro. El registro $R2$ tiene una entrada de control de carga que se activa por la variable de sincronización T_1. Se supone que la variable de sincronización está sincronizada con el mismo reloj que se aplica al registro. Como se indica en el diagrama de sincronización, T_1 es activada por el flanco ascendente de un pulso del reloj al tiempo t. La siguiente transición positiva del reloj al tiempo $t + 1$ nota que $T_1 = 1$ y las entradas de $R2$ se cargan en el registro en paralelo. T_1 puede volver a 0 al tiempo $t + 1$ mientras la variable de sincronización T_2 se vuelve 1 (véase la figura 5-22).

TABLA 7—1
Simbolos básicos de transferencias de registros

Simbolo	Descripción	Ejemplos
Letras (y numerales)	Denota un registro	AR, $R2$
Paréntesis ( )	Denota una parte de un registro	$R2(0\text{–}7)$, $R2(L)$
Flecha ←	Denota transferencia de información	$R2 \leftarrow R1$
Coma ,	Separa dos microoperaciones	$R2 \leftarrow R1$, $R1 \leftarrow R2$
Corchetes o paréntesis cuadrados [ ]	Especifican una dirección de la memoria	$DR \leftarrow M[AR]$

Nótese que el reloj no está incluido como una variable en las instrucciones de transferencia de registros. Se supone que todas las transferencias se llevan a cabo durante una transición de flanco del reloj. Aunque la condición de control como T_1 se vuelva activa al tiempo t, la transferencia real no ocurre sino hasta que el registro es activado por la siguiente transición positiva del reloj.

Los símbolos básicos de la notación de transferencia de registros se presentan en la tabla 7-1. Los registros se denotan por letras mayúsculas y después de las letras puede haber numerales. Se utilizan paréntesis para denotar una parte de un registro mediante la especificación del intervalo o escala de bits o dando un nombre de símbolo a una parte de un registro. La flecha denota una transferencia de información y la dirección de dicha transferencia. Se utiliza una coma para separar dos o más operaciones que se ejecutan al mismo tiempo. La instrucción

$$T_3\colon R2 \leftarrow R1,\ R1 \leftarrow R2$$

denota una operación que intercambia el contenido de dos registros durante un pulso de reloj común siempre que $T_3 = 1$. Esta operación simultánea es posible con registros que tengan flip-flops activados por flanco.

Los paréntesis cuadrados (corchetes) se utilizan junto con la transferencia de memoria. La letra M designa una palabra de memoria y el registro encerrado en corchetes da la dirección de la palabra en la memoria. Esto se explica más a fondo en la sección 7-4.

Selección del multiplexor

Hay ocasiones en las que un registro recibe información de dos fuentes diferentes en tiempos distintos. Considérese la siguiente instrucción condicional:

Si $(T_1 = 1)$ entonces $(R0 \leftarrow R1)$ de lo contrario si $(T_2 = 1)$ entonces $(R0 \leftarrow R2)$

El contenido del registro $R1$ se transferirá al registro $R0$ cuando ocurra la variable de sincronización T_1; en caso contrario, el contenido del registro $R2$ se transfiere a $R0$ cuando ocurre T_2. La instrucción condicional se puede dividir en dos partes utilizando funciones de control.

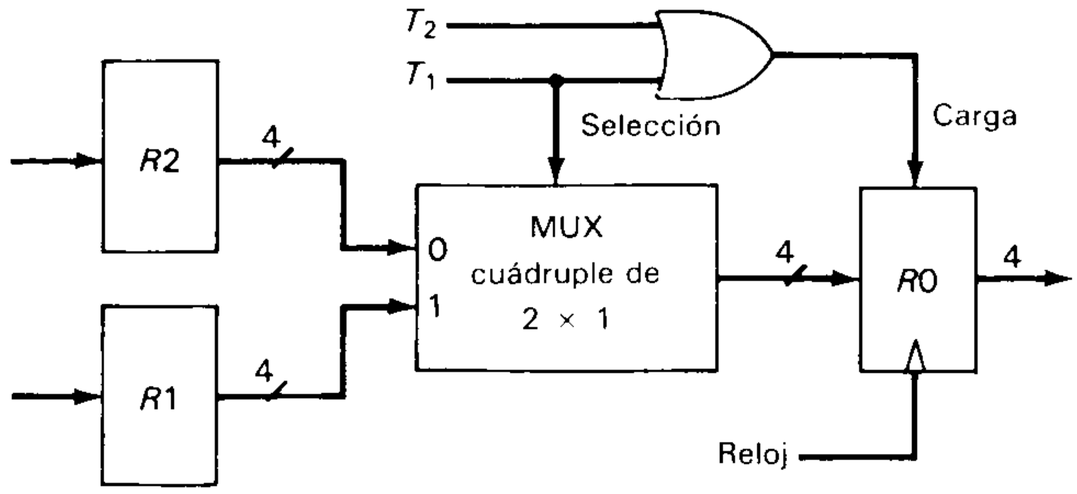
$$T_1\colon R0 \leftarrow R1$$

$$\overline{T}_1 T_2\colon R0 \leftarrow R2$$

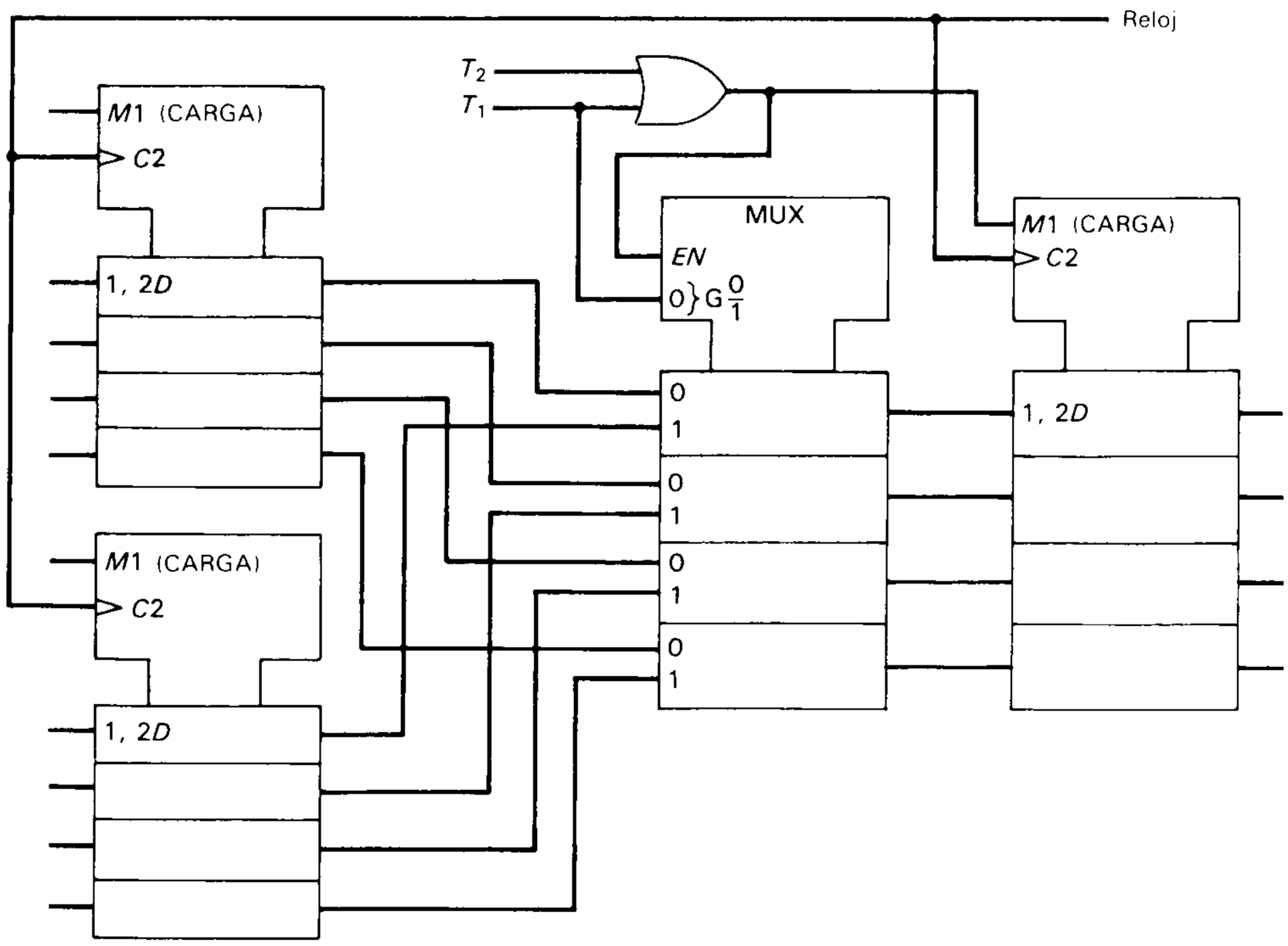
Esto especifica una conexión de *hardware* a partir de dos registros, $R1$ y $R2$ a un registro destino común $R0$. Este tipo de operación requiere un multiplexor para seleccionar entre los dos registros fuente de acuerdo con los valores de las variables de sincronización.

El diagrama de bloques del circuito que ejecuta las dos instrucciones anteriores utilizando registros de 4 bits se muestra en la figura 7-3(a). El multiplexor cuádruple 2×1 selecciona entre los dos registros fuente. Cuando $T_1 = 1$ se selecciona el registro $R1$ y cuando $T_1 = 0$, el multiplexor selecciona el registro $R2$. Por lo tanto, cuando $T_1 = 1$, $R1$ se carga en $R0$ sin que importe el valor de T_2. Cuando $T_2 = 1$ y $T_1 = 0$, $R2$ se carga en $R0$. Cuando T_1 y T_2 son iguales a 0, el multiplexor selecciona $R2$ para las entradas de $R0$ pero las entradas no se cargan en el registro porque la entrada de control de carga es igual a 0.

El diagrama de lógica detallado de la aplicación de *hardware* se muestra en la figura 7-3(b). El diagrama utiliza símbolos gráficos estándar, como se presentaron en



(a) Diagrama de bloques



(b) Diagrama de lógica con símbolos gráficos estándar

FIGURA 7—3
Uso de multiplexores para seleccionar entre dos registros

los capítulos anteriores. El símbolo gráfico de los registros se toma de la figura 5-3(b) y, para el multiplexor, de la figura 3-31(b). La entrada habilitadora *EN* del multiplexor se activa con la misma condición de la entrada de carga del registro destino.

Es importante poder relacionar la información dada en el diagrama de bloques con las conexiones alámbricas detalladas del diagrama de lógica correspondiente. En secciones subsiguientes presentaremos otros diagramas de bloque de diversos sistemas digitales. A fin de ahorrar espacio, el diagrama de lógica detallado puede omitirse. Sin embargo, debe ser un procedimiento directo para obtener el diagrama de lógica con la conexión alámbrica detallada, a partir de la información que se presenta en el diagrama de bloques.

7-3 MICROOPERACIONES

Una microoperación es una operación elemental que se realiza con los datos almacenados en registros. El tipo de microoperaciones que se encuentran más frecuentemente en las computadoras digitales se clasifican en cuatro categorías.

1. Las microoperaciones de transferencia de registros transfieren información binaria de un registro a otro.
2. Las microoperaciones aritméticas efectúan operaciones aritméticas con números almacenados en registros.
3. Las microoperaciones lógicas efectúan operaciones de manipulación de bits con datos no numéricos almacenados en registros.
4. Las microoperaciones de corrimiento realizan operaciones de corrimiento del contenido de los registros.

La microoperación de transferencia de registros se presentó en la sección anterior. Este tipo de microoperación no altera el contenido de información cuando la información binaria pasa del registro fuente al registro destino. Los otros tres tipos de microoperaciones cambian el contenido de información durante la transferencia. Entre todas las operaciones posibles que pueden existir en sistemas digitales, existe un conjunto básico del cual se pueden obtener todas las otras operaciones. En esta sección se define un conjunto de microoperaciones básicas, su notación simbólica y el *hardware* digital que las ejecuta.

Microoperaciones aritméticas

Las microoperaciones aritméticas básicas son la adición, sustracción, incremento, disminución y corrimiento. Los corrimientos aritméticos se explican más adelante junto con las microoperaciones de corrimiento. La microoperación aritmética definida por la instrucción siguiente:

$$R0 \leftarrow R1 + R2$$

especifica una microoperación de *suma* e indica que el contenido del registro $R1$ se sumará al contenido del registro $R2$ y que la suma se transferirá al registro $R0$. Para ejecutar esta instrucción con *hardware* necesitamos tres registros y el componente digital que realiza la operación de adición, como un sumador en paralelo. Las otras microoperaciones aritméticas básicas se presentan en la tabla 7-2. La sustracción se ejecuta con mayor frecuencia mediante la complementación y la adición. En lugar de utilizar el operador menos, podemos especificar la resta por medio de la instrucción siguiente:

$$R0 \leftarrow R1 + \overline{R2} + 1$$

$\overline{R2}$ es el símbolo del complemento a 1's de $R2$. La adición de 1 al complemento a 1's produce el complemento a 2's. Sumar el contenido de $R1$ al complemento a 2's de $R2$ es equivalente a $R1 - R2$.

Las microoperaciones de incremento y decremento se simbolizan por una operación más uno y menos uno, respectivamente. Estas microoperaciones se ejecutan con un circuito combinatorio o con un contador binario ascendente y descendente.

TABLA 7—2
Microoperaciones aritméticas

Designación simbólica	Descripción
$R0 \leftarrow R1 + R2$	Contenido de $R1$ más $R2$ transferido a $R0$
$R0 \leftarrow R1 - R2$	Contenido de $R1$ menos $R2$ transferido a $R0$
$R2 \leftarrow \overline{R2}$	Complemento del contenido de $R2$ (complemento a 1's)
$R2 \leftarrow \overline{R2} + 1$	Complemento a 2's del contenido de $R2$
$R0 \leftarrow R1 + \overline{R2} + 1$	$R1$ más el complemento a 2's de $R2$ (sustracción)
$R1 \leftarrow R1 + 1$	Incremento del contenido de $R1$ (cuenta ascendente)
$R1 \leftarrow R1 - 1$	Decremento del contenido de $R1$ (cuenta descendente)

Las operaciones aritméticas de multiplicación y división no se presentan en la tabla 7-2. La operación de multiplicación se puede representar por el símbolo * y la división por /. Estas dos operaciones son operaciones aritméticas válidas pero no están incluidas en el conjunto básico de microoperaciones. El único lugar donde pueden considerarse como microoperaciones es en un sistema digital donde se ejecutan por medio de un circuito combinatorio. En tal caso, las señales que realizan estas operaciones se propagan a través de compuertas, y el resultado se puede transferir a un registro destino por medio de un pulso de reloj, tan pronto como la señal de salida se propague a través del circuito combinatorio. En muchas computadoras, la operación de multiplicación se ejecuta con una secuencia de microoperaciones de suma y corrimiento. La división se ejecuta con una secuencia de microoperaciones de sustracción y corrimiento. La división se ejecuta con una secuencia de microoperaciones de resta y corrimiento. Para especificar el *hardware* en tal caso se requiere de una lista de instrucciones que utilicen las microoperaciones básicas de adición, sustracción y corrimiento.

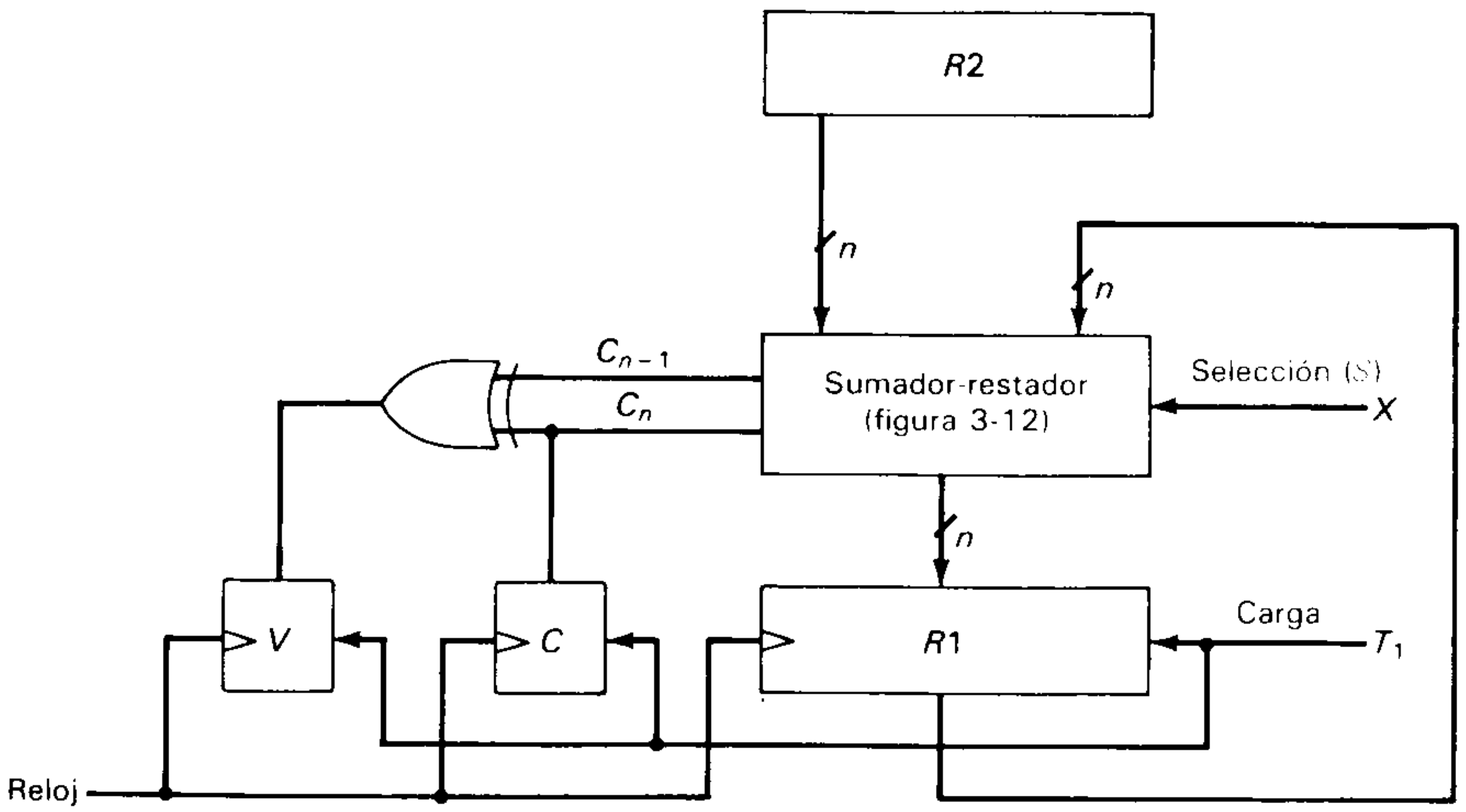


FIGURA 7—4
Ejecución de microoperaciones de adición y sustracción

Existe una relación directa entre las instrucciones escritas en una notación de transferencia de registros y los registros y funciones digitales que necesita su ejecución. Para ilustrarlo con un ejemplo, considérense las dos instrucciones que siguen:

$$\overline{X}T_1\colon R1 \leftarrow R1 + R2$$

$$XT_1\colon R1 \leftarrow R1 + \overline{R2} + 1$$

La variable de sincronización T_1 inicia una operación para sumar o restar. Si, al mismo tiempo, la variable de control X es igual a 0 entonces $\overline{X}T_1 = 1$ y el contenido de $R2$ se suma al contenido de $R1$. Si $X = 1$, entonces $XT_1 = 1$ y el contenido de $R2$ se resta al de $R1$. Nótese que las dos funciones de control son funciones booleanas y se reducen a 0 cuando $T_1 = 0$, condición que inhibe la ejecución de una u otra operación.

La ejecución de las dos instrucciones se muestra en el diagrama de bloque de la figura 7-4. Un sumador-restador de n bits (semejante al que se ilustra en la figura 3-12) recibe sus datos de entrada de los registros $R1$ y $R2$. La suma o diferencia se aplica a las entradas de $R1$. La entrada de selección S del sumador-restador selecciona la operación en el circuito. Cuando $S = 0$, se suman las dos entradas, y cuando $S = 1$, $R2$ se resta de $R1$. Al aplicar la variable de control X a la entrada de selección se produce la operación requerida. La salida del sumador-restador se carga en $R1$ si $\overline{X}T_1 = 1$ o si $XT_1 = 1$. Esto se puede simplificar a sólo T_1 porque

$$\overline{X}T_1 + XT_1 = (\overline{X} + X)T_1 = T_1$$

Por lo tanto, la variable de control X selecciona la operación y la variable de sincronización T_1 carga el resultado en $R1$. El acarreo de salida C_n se transfiere al flip-flop C. La función del flip-flop V consiste en detectar desbordamientos, como se explicó antes.

Desbordamiento

Cuando se suman dos números de n dígitos cada uno y la suma ocupa $n + 1$ dígitos, decimos que ha ocurrido un desbordamiento. Esto sucede con los números binarios o decimales, tengan signo o no. Cuando la adición se realiza con lápiz y papel, un desbordamiento no es problema, ya que no hay límite para la amplitud de la página donde se escribirá la suma. Un desbordamiento es un problema en las computadoras digitales porque la longitud de los registros es finita. Un resultado que contiene $n + 1$ bits no se puede alojar en un registro con una longitud estándar de n bits. Por este motivo, muchas computadoras detectan un desbordamiento y, cuando éste ocurre, se inicia un flip-flop correspondiente que después puede ser verificado por el usuario.

La detección de un desbordamiento después de la adición de dos números binarios depende de si los números se consideran con o sin signo. Cuando se suman dos números sin signo, se detecta un desbordamiento desde el acarreo final en la posición más significativa. En el caso de los números con signo, el último bit a la izquierda representa siempre el signo y los números negativos están en forma de complemento a 2's. Cuando se suman dos números con signo, el bit del signo se trata como parte del número y el acarreo final no indica desbordamiento (véase la sección 1-5).

Un desbordamiento no puede ocurrir después de una adición si un número es positivo y el otro negativo, ya que la adición de un número positivo a uno negativo produce un resultado que es menor que el mayor de los dos números originales. Un des-

bordamiento puede ocurrir si los dos números que se suman son ambos positivos o negativos. Para ver cómo puede suceder esto, considérese el ejemplo que sigue. Dos números binarios con signo, +70 y +80, se almacenan en dos registros de 8 bits. El intervalo de números que puede alojar cada registro es del +127 binario al —128 binario. Como la suma de los dos números es +150, excede la capacidad del registro de 8 bits. Esto sucede si los números son ambos positivos o ambos negativos. Las dos adiciones en binario se muestran a continuación junto con los dos últimos acarreos.

acarreos: 0 1		acarreos: 1 0	
+70	0 1000110	−70	1 0111010
+80	0 1010000	−80	1 0110000
+150	1 0010110	−150	0 1101010

Nótese que el resultado de 8 bits que debía haber sido positivo tiene un bit de signo negativo y el resultado de 8 bits, que debía haber sido negativo, tiene un bit de signo positivo. Sin embargo, si el acarreo que sale de la posición del bit del signo se toma como el bit del signo del resultado, entonces la respuesta de 9 bits así obtenida será correcta. Como la respuesta no se puede alojar en 8 bits, decimos que ocurrió un desbordamiento.

Una condición de desbordamiento se puede detectar observando el acarreo *en* la posición del bit del signo y el acarreo *fuera* de la posición del bit del signo. Si estos dos acarreos no son iguales, se produce una condición de desbordamiento. Esto se indica en los ejemplos donde se muestran explícitamente los dos acarreos. Si los dos acarreos se aplican a una compuerta OR excluyente, se detectará un desbordamiento cuando la salida de la compuerta sea igual a 1.

La adición y sustracción de dos números binarios con *hardware* digital se muestra en la figura 7-4. Si los números se consideran sin signo, entonces el bit C detecta un acarreo después de la adición o un préstamo después de la sustracción. Si se considera que los números tienen signo, entonces el bit V detecta un desbordamiento. Si $V = 0$ después de una adición o sustracción, esto indica que no ocurrió desbordamiento y que la respuesta $R1$ es correcta. Si $V = 1$, entonces el resultado de la operación contiene $n + 1$ bits. Sólo n bits del número están en $R1$. El $(n + 1)$-ésimo bit es el bit del signo y se ha desplazado o corrido de la posición al bit de acarreo C.

Microoperaciones lógicas

Las microoperaciones lógicas son útiles para manipular los bits almacenados en un registro. Estas operaciones consideran cada bit del registro por separado y lo tratan como una variable binaria. Los símbolos de las cuatro microoperaciones lógicas básí-

TABLA 7—3
Microoperaciones lógicas

Designación simbólica	Descripción
$R \leftarrow \overline{R}$	Complementa todos los bits del registro R
$R0 \leftarrow R1 \wedge R2$	Microoperación AND lógica (pone a bits en ceros)
$R0 \leftarrow R1 \vee R2$	Microoperación OR lógica (inicia a bits)
$R0 \leftarrow R1 \oplus R2$	Microoperación XOR lógica (complementa a bits)

cas se muestran en la tabla 7-3. La microoperación de complemento es la misma que el complemento a 1's y emplea una barra arriba del nombre del registro. El símbolo $\wedge$ se utiliza para denotar la microoperación AND y el símbolo + para denotar la microoperación OR. Con la utilización de estos símbolos especiales es posible diferenciar entre la microoperación de suma simbolizada por un signo de + y la microoperación OR. Aunque el símbolo + tiene dos significados, será posible distinguir entre ellos observando dónde se presenta. Cuando este símbolo se presenta en una microoperación, denota una operación de adición o suma. Cuando ocurre en una función de control o booleana, denota una operación OR. La microoperación OR utilizará siempre el símbolo $\vee$. Por ejemplo, en la instrucción

$$T_1 + T_2: R1 \leftarrow R2 + R3,\ R4 \leftarrow R5 \vee R6$$

el signo + entre T_1 y T_2 es una operación OR entre dos variables en una función de control (booleana). El signo + entre $R2$ y $R3$ especifica una microoperación de adición. La microoperación OR se designa por el símbolo + entre los registros $R5$ y $R6$.

Las microoperaciones lógicas se pueden ejecutar fácilmente con un grupo de compuertas. El complemento de un registro de n bits se obtiene con n compuertas inversoras. La microoperación AND se obtiene a partir de un grupo de compuertas AND y cada una de ellas recibe un par de bits de dos registros fuente. Las salidas de las compuertas se aplican a las entradas del registro destino. Las microoperaciones OR y Exclusive-OR (u OR excluyente) requieren una disposición de compuertas semejante.

Las microoperaciones lógicas pueden cambiar valores de bits, eliminar un grupo de bits o insertar nuevos valores de bits en un registro. Los ejemplos que siguen muestran cómo son manipulados los bits del registro $R1$ por microoperaciones lógicas con un operando lógico en $R2$.

La microoperación AND se utiliza para anular a 0 un bit o un grupo de bits seleccionado en un registro. Las relaciones booleanas $X \cdot 0 = 0$ y $X \cdot 1 = X$ prescriben que una variable binaria X, cuando se compara lógicamente (con AND) con 0 produce un 0; pero cuando se compara (con AND) con 1 no cambia el valor de X. Un bit o un grupo de bits dado en un registro puede ponerse en 0 si se compara (con AND) con 0. Considérese el ejemplo que sigue.

```
10101101 10101011      R1
00000000 11111111      R2
00000000 10101011      R1 ← R1 ∧ R2
```

El operando lógico de 16 bits en $R2$ tiene ceros en el byte de alto orden y unos en el byte de bajo orden. Comparando esto lógicamente (con AND) con el contenido de $R1$ es posible poner en ceros el byte de alto orden de $R1$ y dejar intactos los bits del byte de bajo orden. Por lo tanto, la operación AND se puede emplear para poner en ceros selectivamente los bits de un registro. Esta operación se conoce a veces como *enmascaramiento* de los bits porque cubre o suprime todos los unos de una parte seleccionada de un registro.

La microoperación OR se utiliza para iniciar un bit o un brupo de bits en un registro. Las relaciones booleanas $X + 1 = 1$ y $X + 0 = X$ prescriben que la variable binaria X, cuando se compara lógicamente (con OR) con 1, produce un 1, pero

cuando se compara (con OR) con 0 no cambia el valor de X. Un bit o grupo de bits dado de un registro se puede iniciar a 1 si se compara (con OR) con 1. Considérese el ejemplo que sigue.

10101101 10101011	$R1$
11111111 00000000	$R2$
11111111 10101011	$R1 \leftarrow R1 \vee R2$

El byte de alto orden de $R1$ se inicia a unos comparándolo (con OR) con todos los unos del operando $R2$. El byte de bajo orden se conserva intacto porque se compara (con OR) con ceros.

La microoperación XOR (OR excluyente) se utiliza para complementar un bit o un grupo de bits en un registro. Las relaciones booleanas $X \oplus 1 = \overline{X}$ y $X \oplus 0 = X$ prescriben que cuando la variable binaria X se compara (con XOR) lógicamente con 1 se complementa; pero cuando se compara (con XOR) con 0 se mantiene intacta. Comparando (con XOR) un bit o un grupo de bits en un registro con 1, es posible complementar los bits seleccionados. Considérese el ejemplo

10101101 10101011	$R1$
11111111 00000000	$R2$
01010010 10101011	$R1 \leftarrow R1 \oplus R2$

El byte de alto orden de $R1$ se complementa después de la operación XOR con el operando en $R2$.

TABLA 7—4
Microoperaciones de corrimiento

Designación simbólica	Descripción
$R \leftarrow \text{shl } R$	Corrimiento a la izquierda del registro R
$R \leftarrow \text{shr } R$	Corrimiento a la derecha del registro R
$R \leftarrow \text{rol } R$	Rotación a la izquierda del registro R
$R \leftarrow \text{ror } R$	Rotación a la derecha del registro R
$R \leftarrow \text{asl } R$	Corrimiento aritmético a la izquierda de R
$R \leftarrow \text{asr } R$	Corrimiento aritmético a la derecha de R

Microoperaciones de corrimiento

Las microoperaciones de corrimiento se utilizan para transferir datos en serie. También se utilizan en operaciones aritméticas, lógicas y de control. El contenido de un registro se puede correr a la izquierda o a la derecha. No hay símbolos estándar para las microoperaciones de corrimiento. Aquí adoptaremos los símbolos *shl* y *shr* para referirnos a operaciones de corrimiento a la izquierda y a la derecha, respectivamente. Por ejemplo

$$R1 \leftarrow \text{shl } R1,\ R2 \leftarrow \text{shr } R2$$

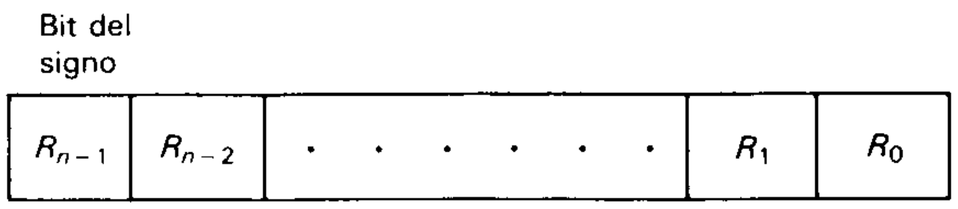


FIGURA 7—5
Definición del registro *R* para corrimientos aritméticos

son dos microoperaciones que especifican un corrimiento de 1 bit a la izquierda del registro *R*1 y un corrimiento de 1 bit a la derecha del registro *R*2. El símbolo del registro debe ser el mismo en ambos lados de la flecha, como en la operación de incremento.

Mientras se desplazan los bits de un registro, la posición del bit final recibe información de la entrada en serie. La posición final es el último bit a la derecha del registro durante una operación de corrimiento a la izquierda y el último bit a la izquierda del registro durante una operación de corrimiento a la derecha. El bit que se transfiere a la posición final a través de la entrada en serie se toma como 0 durante un corrimiento lógico, que se simboliza por shl o shr. En una operación de *rotación,* la salida en serie se conecta a la entrada en serie y los bits del registro giran sin que haya pérdida de información. El símbolo que se utiliza para la rotación se muestra en la tabla 7-4.

Un corrimiento aritmético es una microoperación que desplaza un número binario con signo a la izquierda o a la derecha. Un corrimiento aritmético a la izquierda multiplica un número binario con signo por 2. Un corrimiento aritmético a la derecha divide el número entre 2. Los corrimientos aritméticos deben dejar intacto el bit del signo porque el signo del número se mantiene idéntico cuando se multiplica por 2 o cuando se divide entre este mismo número.

El último bit a la izquierda en un registro contiene el bit del signo y los bits restantes alojan al número. La figura 7-5 ilustra un registro típico de n bits. El bit R_{n-1} en la última posición a la izquierda contiene el bit del signo. R_0 es el bit menos significativo y R_{n-2} es el bit más significativo del número. Se supone que los números negativos tienen un 1 en el bit del signo y están en forma de complemento a 2's.

El corrimiento aritmético a la derecha deja intacto el bit del signo y desplaza el número (incluyendo el bit del signo) a la derecha. Por lo tanto R_{n-1} se mantiene igual, R_{n-2} recibe el bit de R_{n-1} y así sucesivamente para los demás bits del registro. El bit en R_0 se pierde.

El corrimiento aritmético a la izquierda inserta un 0 en R_0 y corre todos los otros bits a la izquierda. El bit inicial de R_{n-1} se pierde y es reemplazado por el bit de R_{n-2}. Ocurre una inversión de signo si el bit en R_{n-1} cambia de valor después del corrimiento. Esto sucede si la multiplicación por 2 causa un desbordamiento. Ocurre un desbordamiento después de un corrimiento aritmético a la izquierda si R_{n-1} no es igual a R_{n-2} *antes* del corrimiento. Un flip-flop de desbordamiento V_s se puede utilizar para detectar un desbordamiento de corrimiento aritmético.

$$V_s = R_{n-1} \oplus R_{n-2}$$

Si $V_s = 0$, no hay desbordamiento, pero si $V_s = 1$, hay un desbordamiento y una inversión de signo después del corrimiento. V_s debe transferirse al flip-flop de desbordamiento con el mismo pulso del reloj que desplaza el registro.

7-4 TRANSFERENCIA DEL BUS

Una computadora digital común tiene muchos registros y trayectorias que deben generarse para transferir información de un registro a otro. El número de alambres será excesivo si se utilizan líneas y multiplexores separados entre cada registro y todos los otros registros del sistema. Un esquema más eficiente para transferir informa-

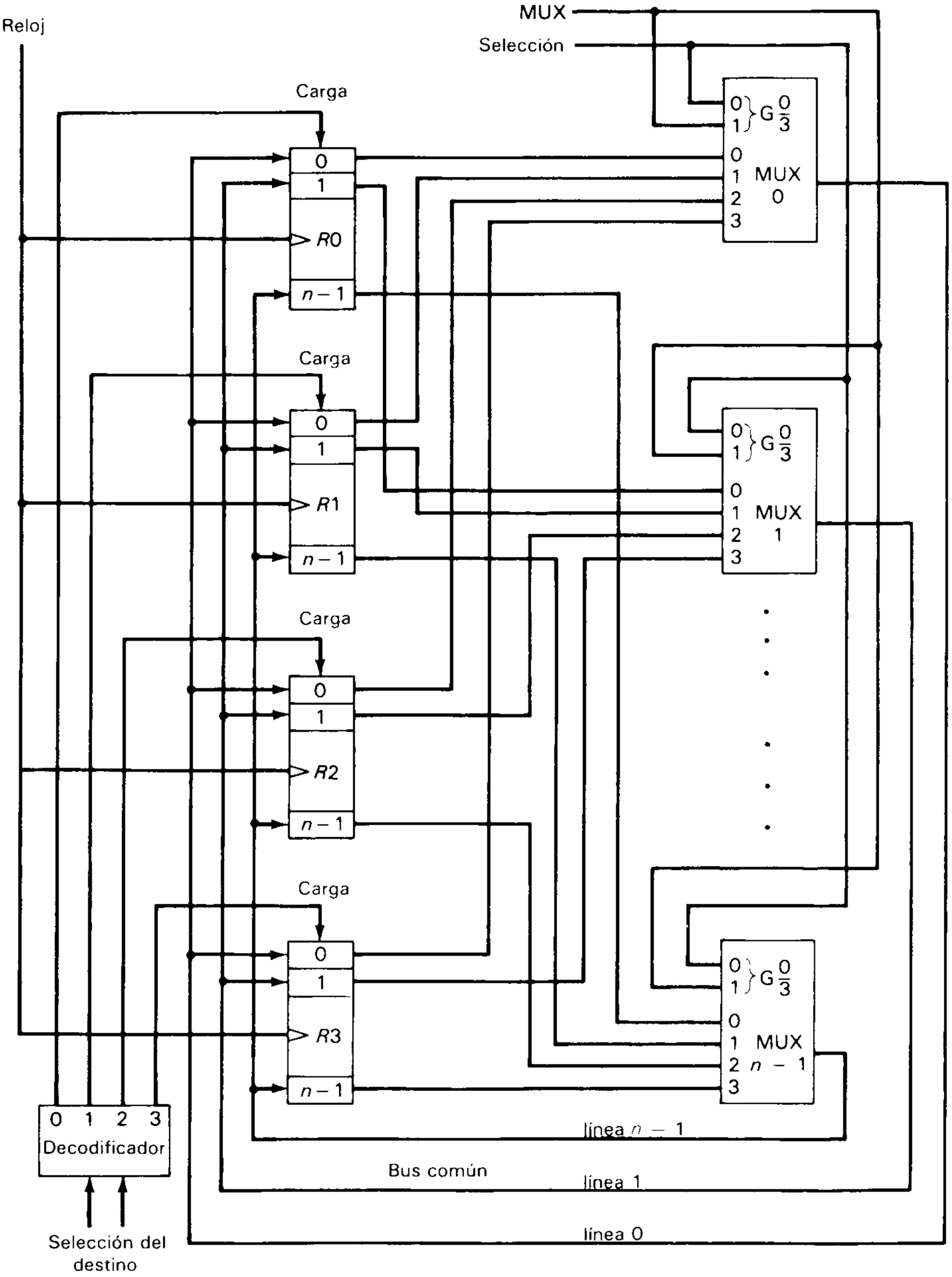


FIGURA 7—6
Sistema de bus de cuatro registros

ción entre registros en una configuración de registros múltiples es un sistema de *bus*. Una estructura de bus consta de un conjunto de líneas comunes, una para cada bit de un registro, a través de los cuales se transfiere información binaria, uno a la vez. Las señales de control seleccionan cuál registro será la fuente y cuál el destino durante cada transferencia de registros.

Una manera de construir un sistema de bus común es con multiplexores y un decodificador. Los multiplexores seleccionan un registro fuente cuya información binaria se coloca después en el bus. El decodificador selecciona un registro destino que acepta la información del bus. La construcción de un sistema de bus de cuatro registros se ilustra en la figura 7-6. Cada registro tiene n bits numerados del 0 al $n - 1$. Los bits de la misma posición significativa en cada registro se aplican a un multiplexor de 4 líneas en 1 línea para formar una línea del bus. En el diagrama sólo se ilustran tres multiplexores. El circuito completo debe tener n multiplexores numerados del 0 al $n - 1$. Las n líneas formadas por el sistema de bus común se envían a las n entradas de cada registro. La transferencia de información de bus a un registro destino se logra activando la entrada de control de carga del registro seleccionado. La entrada de carga en particular se selecciona a partir de las salidas del decodificador.

Las entradas de selección del MUX determinan qué registro colocará su contenido en el bus. Las entradas de selección del MUX pueden ser 00, 01, 10 u 11, que seleccionan los registros $R0$, $R1$, $R2$ o $R3$, respectivamente. Las entradas de selección del destino al decodificador determinan el registro que recibe la información del bus. Las entradas de selección del destino pueden ser 00, 01, 10 u 11, y seleccionan el registro destino $R0$, $R1$, $R2$ o $R3$, respectivamente.

Para ilustrar lo expuesto con un ejemplo en particular, considérese la transferencia dada por la siguiente instrucción:

$$R2 \leftarrow R0$$

Las variables de control que habilitan esta transferencia deben seleccionar el registro $R0$ como la fuente del bus y el registro $R2$ como el destino. Las entradas de selección del multiplexor deben ser 00 binario. Esto hace que el bit 0 de $R0$ se aplique a la línea 0 del bus que pasa por el MUX 0. Al mismo tiempo el bit 1 de $R0$ se aplica a la línea 1 del bus que pasa por el MUX 1. Esto se repite con todas las otras líneas del bus hasta la línea $n - 1$, la que recibe el bit $n - 1$ de $R0$ que pasa por el MUX $n - 1$. Por lo tanto, el valor de n bits de $R0$ se coloca en las líneas del bus común cuando la selección del MUX es 00. Las entradas de selección del destino deben ser 10 binario. Esto activa la salida 2 del decodificador, el cual, a su vez, activa la entrada de carga de $R2$. Con la siguiente transición del reloj, el contenido de $R0$, que está en el bus, se carga en el registro $R2$ para completar la transferencia.

Buffers con bus de tres estados

Un sistema de bus se puede construir con compuertas de tres estados en lugar de multiplexores. Una compuerta de tres estados es un circuito digital que exhibe tres estados. Dos de los estados son señales equivalentes al 1 y 0 lógicos, como en una compuerta convencional. El tercer estado es un estado de *alta impedancia,* el cual se comporta como un circuito abierto, lo que significa que la salida se desconecta y no tiene significado lógico. Las compuertas de tres estados pueden efectuar cualquier operación de lógica convencional como AND o NAND. Sin embargo, una de las compuertas que se

utiliza más comúnmente en el diseño de un sistema de bus es la compuerta buffer o separadora.

El símbolo gráfico de una compuerta buffer de tres estados se ilustra en la figura 7-7 y se distingue de un buffer normal por su pequeño símbolo triangular enfrente de la terminal de salida. El circuito tiene una entrada normal y una entrada de control que determinan el estado de salida. Cuando la entrada de control es igual a 1, la salida se habilita y la compuerta se conduce como cualquier buffer convencional, donde la salida es igual a la entrada normal. Cuando la entrada de control es 0, la salida se inhabilita y la compuerta pasa a un estado de alta impedancia, independientemente del valor de la entrada normal. El estado de alta impedancia de una compuerta de tres estados ofrece una característica especial que no se encuentra en otras compuertas, debido a la cual, se puede conectar un gran número de salidas de compuertas de tres estados con alambres a fin de formar una línea de bus común sin comprometer efectos de carga.

La construcción de un sistema de bus con buffers de tres estados se demuestra en la figura 7-8. Las salidas de cuatro buffers se conectan para formar una sola línea del bus. (Este tipo de conexión no se puede realizar con compuertas que no tengan salidas de tres estados.) Las entradas de control a los buffers determinan cuál de las cuatro entradas normales se comunicará con la línea del bus. No más de un buffer puede estar en el estado activo en cualquier momento dado. Los buffers conectados deben controlarse de manera que sólo un buffer de tres estados tenga acceso a la línea del bus mientras todos los otros buffers se mantienen en un estado de alta impedancia.

Una manera de asegurar que no más de una entrada de control esté activa en cualquier momento dado consiste en utilizar un decodificador como se ilustra en el diagrama. Cuando la entrada habilitadora del decodificador es 0, sus cuatro salidas son 0, y la línea del bus se encuentra en un estado de alta impedancia porque se inhabilitan los cuatro buffers. Cuando la entrada habilitadora esté activa, uno de los buffers de tres estados estará activo dependiendo del valor binario en las entradas de selección del decodificador. La investigación cuidadosa revelará que la figura 7-8 representa otra manera de construir un multiplexor de 4 × 1. El diagrama ilustra las entradas marcadas con un nombre de registro y el valor del bit en paréntesis. Por lo tanto, $R0(k)$ designa al bit k en el registro $R0$. El circuito de la figura 7-8 puede reemplazar al multiplexor de la figura 7-6 que produce la línea del bus del bit k. Para construir un bus común de cuatro registros de n bits con buffers de tres estados se requieren n grupos de cuatro buffers cada uno para seleccionar entre los cuatro registros.

Hay ocasiones en que es necesario emplear un sistema de bus bidireccional que pueda transferir información en ambas direcciones. Un bus bidireccional permite que la información binaria circule en una u otra de dos direcciones. Un bus bidireccional puede construirse con buffers de tres estados para controlar la dirección del flujo de información en el bus. Una línea de un bus bidireccional se muestra en la

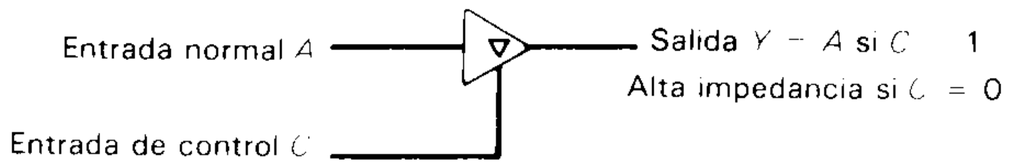


FIGURA 7—7
Simbolo gráfico de un buffer de tres estados

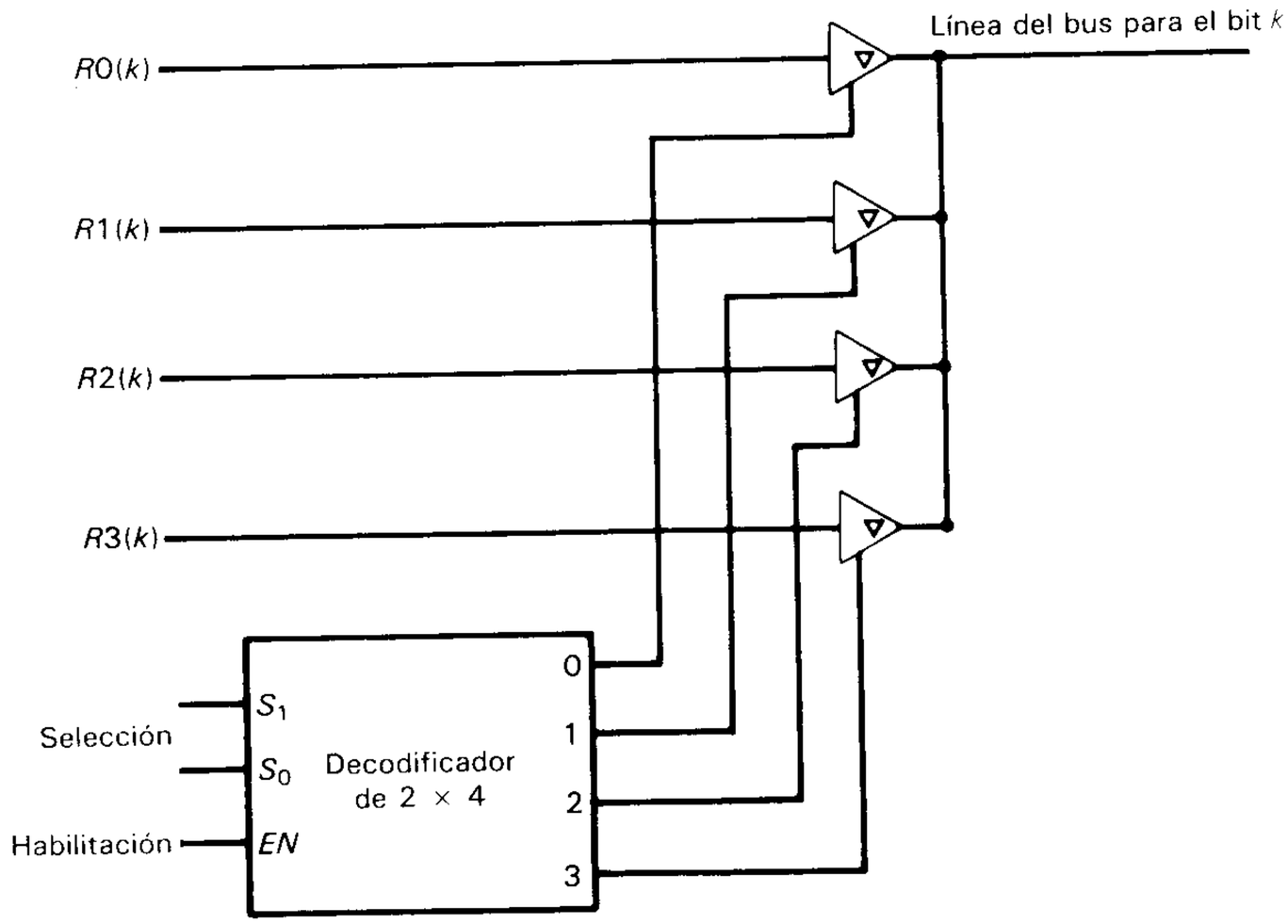


FIGURA 7—8
Línea de bus con buffers de tres estados

figura 7-9. El control del bus tiene dos líneas de selección S_{in} para la transferencia de entrada y S_{out} para la transferencia de salida. Estas líneas de selección controlan dos buffers de tres estados conectados espalda con espalda. Cuando $S_{in} = 1$ y $S_{out} = 0$, se habilita el buffer inferior y el superior se inhabilita pasando a un estado de alta impedancia. Esto forma una trayectoria para la entrada de datos que provienen del bus para atravesar el buffer inferior y llegar a la entrada de un registro del flip-flop.

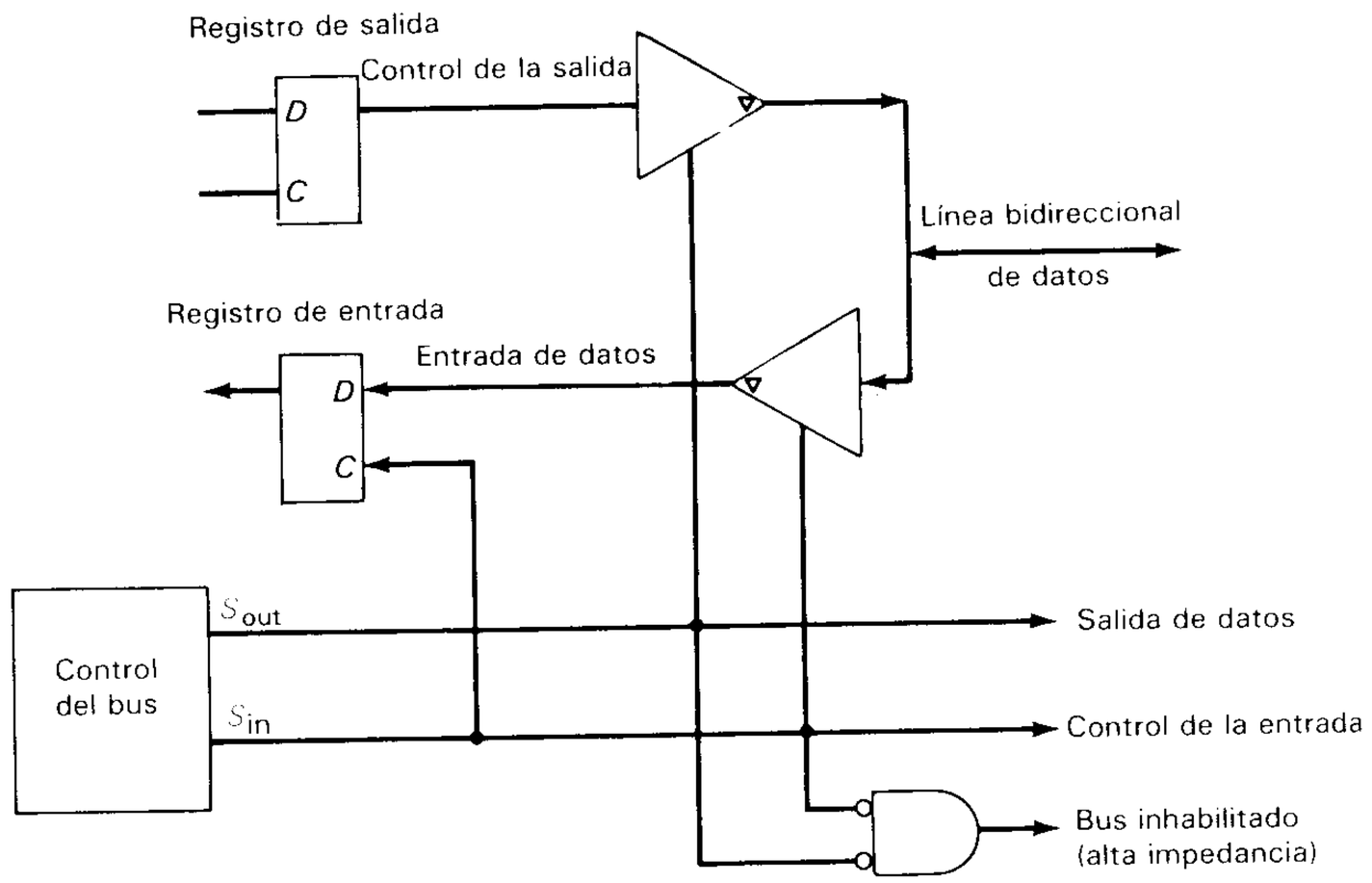


FIGURA 7—9
Línea de bus bidireccional con buffers de tres estados

Cuando $S_{out} = 1$ y $S_{in} = 0$, se habilita el buffer superior y el buffer inferior pasa a un estado de alta impedancia. Esto forma una trayectoria para la salida de datos que provienen de un registro del sistema para atravesar el buffer inferior y llegar a la salida de la línea del bus. La línea del bus se puede inhabilitar haciendo que ambas señales de control sean 0. Esto coloca ambos buffers en un estado de alta impedancia, lo que impide que cualquier transferencia de información de entrada o de salida atraviese la línea del bus. Esta condición existirá cuando una fuente externa utilice el bus común para comunicarse con algún otro destino externo. Las dos líneas de selección se pueden utilizar para informar a los módulos los externos conectados al bus del estado en el que se encuentra el bus bidireccional en cualquier momento dado.

Transferencia de la memoria

La operación de una unidad de memoria se describe en la sección 6-2. La transferencia de información de una palabra de memoria al exterior recibe el nombre de operación de lectura. La transferencia de nueva información a una palabra de memoria recibe el nombre de operación de escritura. Una palabra de memoria estará simbolizada por la letra M. La palabra de memoria en particular entre las muchas disponibles se selecciona por la dirección de la memoria durante la transferencia. Es necesario especificar la dirección de M cuando se escriben operaciones de transferencia de la memoria. Esto se hará encerrando la dirección en corchetes después de la letra M.

Considérese una unidad de memoria que recibe la dirección de un registro llamado registro de dirección y que está simbolizado por AR. Los datos se transfieren a otro registro que se conoce como registro de datos, simbolizado por DR. La operación de lectura se puede establecer de la manera siguiente:

$$\text{Lectura: } DR \leftarrow M[AR]$$

Esto produce una transferencia de información a DR de la palabra de memoria seleccionada que especifica la dirección en AR.

La operación de escritura es una transferencia de DR a la palabra de memoria seleccionada M, y se puede señalar en forma simbólica como sigue:

$$\text{Escritura: } M[AR] \leftarrow DR$$

Esto origina una transferencia de información de DR a la palabra de la memoria seleccionada por la dirección en AR.

En algunos sistemas, la unidad de memoria recibe direcciones y datos de muchos registrados a buses comunes. Considérese el caso que se representa en la figura 7-10. La dirección de la memoria viene de un bus de direcciones. Se conectan cuatro registros a este bus y cualquiera de ellos puede ofrecer una dirección. Los datos de la memoria se conectan a un bus de datos bidireccional. El contenido de la palabra de la memoria seleccionada durante una operación de lectura puede dirigirse a cualquiera de los cuatros registros que selecciona un decodificador. La palabra de datos de la memoria durante una operación de escritura proviene de uno de cuatro registros seleccionados por el bus de datos. La dirección del flujo de información en el bus de datos se determina a partir de las entradas de control del bus. En el caso de una operación de lectura, la trayectoria se extiende de la memoria a un registro

de datos. En el caso de una operación de escritura la trayectoria va de un registro de datos a la memoria. Cada línea bidireccional es controlada por una pareja de buffers de tres estados, como se indica en la figura 7-9.

Una instrucción de transferencia de la memoria en un sistema de registros múltiples debe especificar el registro de dirección y el registro de datos que se utilicen. Podemos ver por ejemplo, la transferencia de información del registro de datos $D2$ a una palabra de memoria seleccionada por la dirección del registro $A1$ se simboliza por la instrucción

$$M[A1] \leftarrow D2$$

Esta es una operación de escritura, donde $A1$ produce la dirección. La instrucción no especifica los buses de manera explícita. No obstante, implica las entradas de selección requeridas para que el bus de direcciones sea 01 (por $A1$) y la selección del registro de entradas de escritura sea 10 (por $D2$). El control del bus activa las señales de control de escritura y de salida.

La operación de lectura en una memoria con buses se puede especificar de manera análoga. La instrucción

$$D0 \leftarrow M[A3]$$

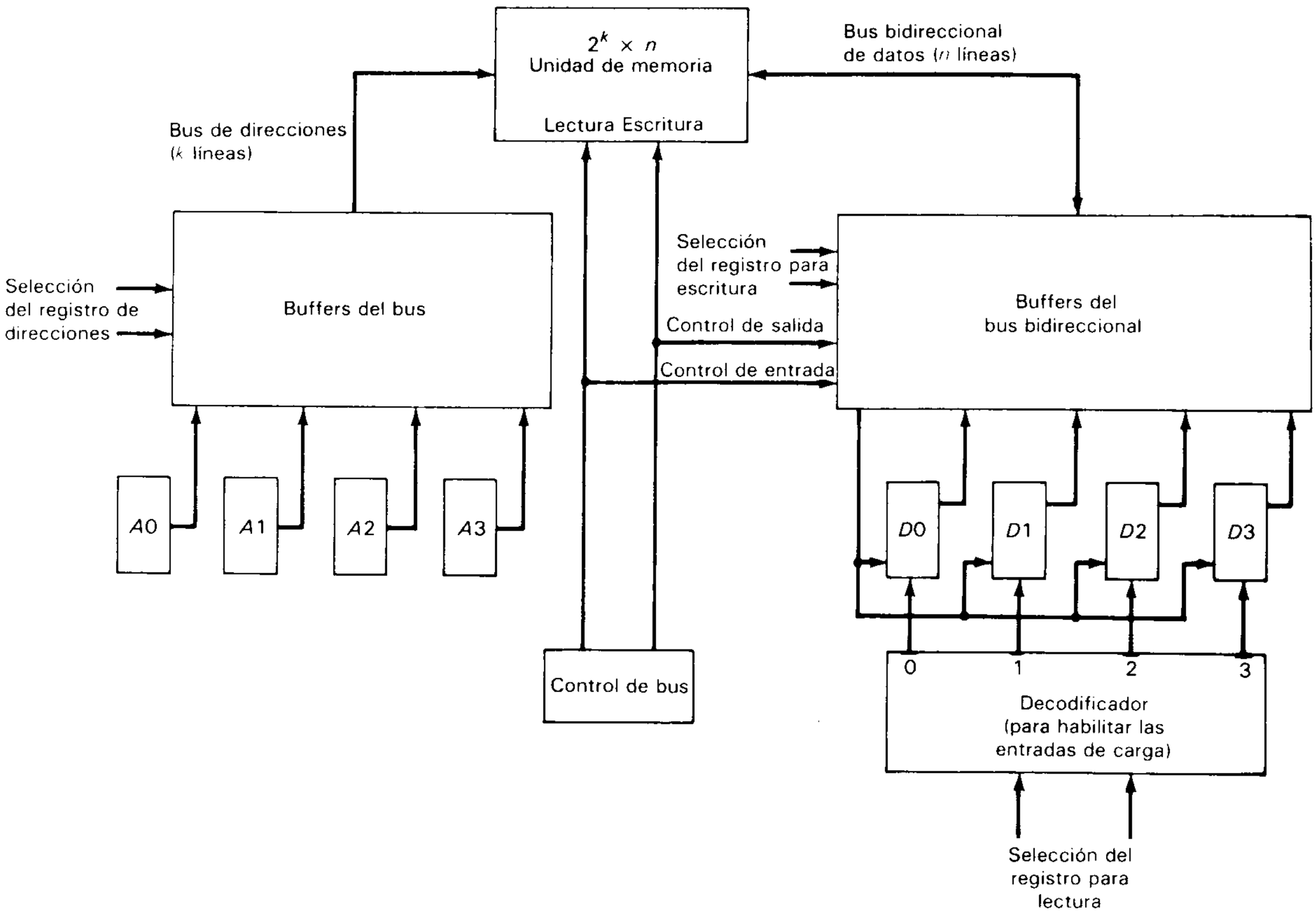


FIGURA 7—10
Unidad de memoria conectada a buses de direcciones y datos

simboliza una operación de lectura de una palabra de memoria cuya dirección está dada por el registro *A*3. La información que sale de la memoria se transfiere al registro de datos *D*0. Una vez más, esta instrucción implica las entradas de selección requeridas para que el bus de direcciones sea 11 (por *A*3) y para que las entradas al decodificador sean 00 (por *D*0), mientras que el control del bus activa las señales de control de lectura y de entrada.

7-5 UNIDAD PROCESADORA

La unidad procesadora es un componente central de un sistema de computación digital, y consta de un número de registros y de circuitos digitales que ejecutan diversas microoperaciones. La parte del procesador de la computadora se conoce a veces como la *trayectoria de datos* porque forma las trayectorias de las operaciones entre los registros. Se dice que las diversas partes son controladas por compuertas que forman la trayectoria requerida para cada operación en particular. En una unidad procesadora típica, las trayectorias de datos se forman por medio de buses y otras líneas comunes. Las compuertas de control que formulan la trayectoria dada son básicamente multiplexores y decodificadores cuyas líneas de selección especifican la trayectoria requerida. El procesamiento de información es realizado por un circuito común que recibe el nombre de unidad de aritmética y lógica, que se abrevia como ALU.

Cuando se incluye un gran número de registros en una unidad procesadora, resulta más eficaz conectarlos a través de buses comunes. Los registros se comunican entre sí no sólo para la transferencia directa de datos, sino también mientras realizan diversas microoperaciones. En la figura 7-11 se ilustra una organización de buses con cuatro registros y una ALU. Cada registro está conectado a dos conjuntos de multiplexores para formar los buses de entrada *A* y *B*. Las entradas de selección en cada conjunto de multiplexores seleccionan un registro para el bus correspondiente. Los buses *A* y *B* se aplican a las entradas de una unidad de aritmética y lógica común. Las entradas de selección de la ALU determinan la operación en particular que se efectúa. Las operaciones de corrimiento se ejecutan en la unidad de corrimiento. Los datos de salida de la ALU se pueden correr a la derecha o a la izquierda, o pueden pasar por la unidad de corrimiento sin cambio, dependiendo de la entrada de selección del corrimiento. El resultado de la operación se coloca en el bus de salida que a su vez se conecta a las entradas de todos los registros. El registro destino que recibe el resultado del bus de salida es seleccionado por un decodificador. El decodificador activa una entrada de carga de registros para producir la trayectoria de transferencia entre los datos del bus de salida y el registro destino.

El bus de salida tiene otras terminales para transferir datos de la unidad procesadora a un dispositivo externo. Los casos del exterior pueden entrar a la unidad procesadora a través de las terminales de datos de entrada en uno de los multiplexores.

A veces conviene suplementar a la ALU con un número de bits de estado o condición, útiles para verificar ciertas relaciones entre los valores de *A* y *B* después de una operación de la ALU. En la figura 7-11 se ilustran cuatro bits de estado o condición. El acarreo *C* y el desbordamiento *V* se explican junto con la figura 7-4. El bit de estado cero *Z* se inicia a 1 si la salida de la ALU contiene sólo ceros y se pone en 0 en caso contrario. Por lo tanto, $Z = 1$ si el resultado de una operación es cero y $Z = 0$ si el resultado es distinto de cero. El bit de estado del signo *S* se inicia al valor del bit del

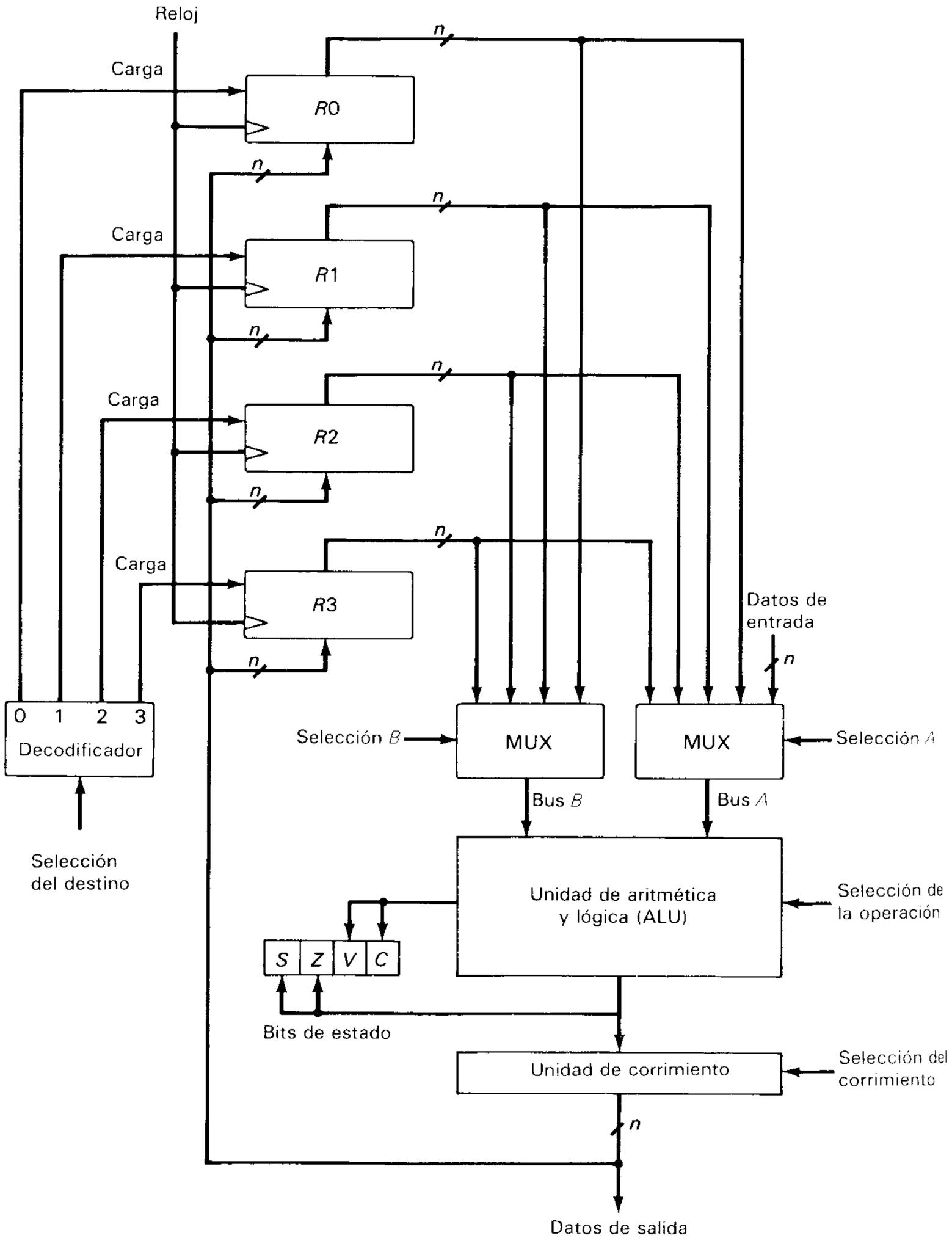


FIGURA 7—11
Diagrama de bloques de una unidad procesadora

signo del resultado. El bit del signo es siempre el último bit a la izquierda de la ALU.

Una unidad procesadora típica tendrá más de cuatro registros. Las computadoras con 16 o más registros son muy comunes. La construcción de un sistema de bus con un gran número de registros requiere multiplexores más grandes y la ALU, por lo contrario, es semejante a la organización que se representa en la figura 7-11.

La unidad de control que supervisa el sistema de bus del procesador debe dirigir el flujo de información a través de los buses, la ALU y la unidad de corrimiento se-

7-6 U

leccionando las diversas componentes de la unidad. Por ejemplo, para realizar la microoperación.

$$R1 \leftarrow R2 + R3$$

la unidad de control debe proporcionar variables de selección binaria a las siguientes entradas de selección:

1. Selector del MUX A: para colocar el contenido de $R2$ en el bus A.
2. Selector del MUX B: para colocar el contenido de $R3$ en el bus B.
3. Selector de operación de la ALU: para generar la operación aritmética $A + B$.
4. Selector de corrimiento: para generar una transferencia directa de la salida de la ALU al bus de salida (no hay corrimiento).
5. Selector del destino del decodificador: para cargar el contenido del bus de salida en $R1$.

Los cinco conjuntos de variables de selección deben generarse en forma simultánea y deben estar disponibles en las terminales correspondientes al inicio de un periodo de pulsos de reloj. Los datos binarios de los dos registros fuente deben propagarse a través de los multiplexores, la ALU, la unidad de corrimiento y a las entradas del registro destino, todo durante un periodo de pulsos del reloj. Después, cuando llega la siguiente transición de borde del reloj, la información binaria en el bus de salida se carga en el registro destino. Para obtener un tiempo de respuesta rápido, la ALU se construye con circuitos de alta velocidad y la unidad de corrimiento con compuertas combinatorias.

La operación de los multiplexores, los buses y el decodificador destino se presentó en la sección anterior. El diseño de una ALU y de la unidad de corrimiento se estudiará en las dos secciones que siguen. El control de la unidad procesadora a través de sus entradas de selección se demuestra en la sección 7-8.

7-6 UNIDAD DE ARITMETICA Y LOGICA (ALU)

Una unidad de aritmética lógica (ALU) es un circuito combinatorio que realiza un conjunto de microoperaciones de aritmética y lógica básicas. La ALU tiene un número de líneas de selección que sirven para elegir una operación determinada en la unidad. Las líneas de selección se decodifican dentro de la ALU de manera que k variables de selección puedan especificar hasta 2^k operaciones distintas.

La figura 7-12 muestra el diagrama de bloque de una ALU típica de 4 bits. Las cuatro entradas de datos de A se combinan con las cuatro entradas de datos de B para generar una operación en las salidas F. La entrada de selección del modo S_2 distingue entre operaciones aritméticas y de lógica (o lógicas). Las dos entradas de selección de funciones, S_1 y S_0, especifican la operación de aritmética o lógica en particular que se generará. Con tres líneas de selección es posible especificar cuatro operaciones aritméticas con S_2 en un estado y cuatro operaciones lógicas con S_2 en el otro estado. Los acarreos de entrada y salida tienen significado sólo durante una operación aritmética. El acarreo de entrada C_{in} se utiliza muy a menudo como cuarta variable de selección para operaciones aritméticas. En esta forma es posible duplicar el número de operaciones aritméticas de cuatro a ocho.

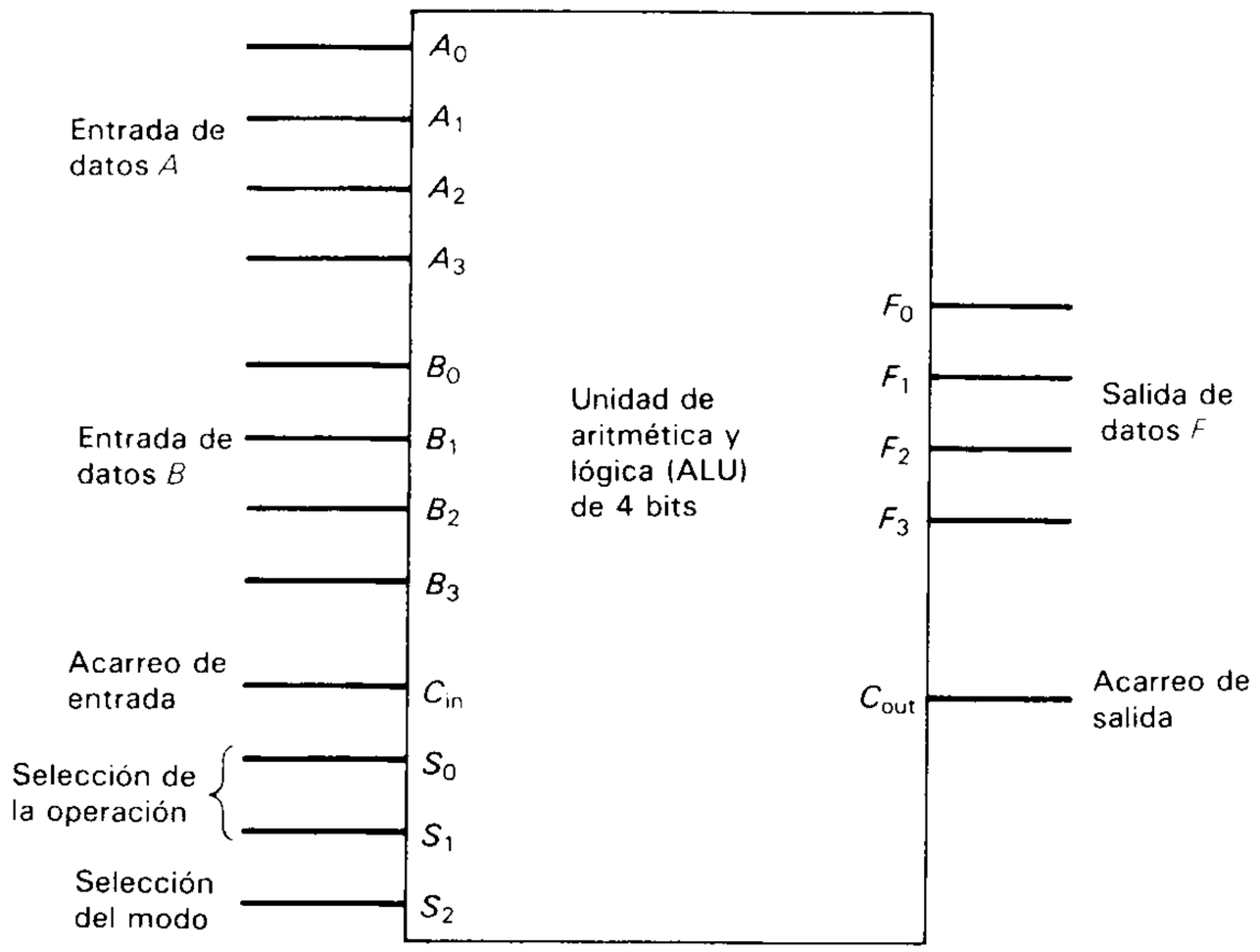


FIGURA 7—12
Diagrama de bloque de una ALU de 4 bits

El diseño de una ALU típica se realizará en tres etapas. Primero, se hará el diseño de la sección aritmética. Segundo, se presentará el diseño de la sección lógica. Después se combinarán las dos secciones para formar la unidad de aritmética y lógica.

Circuito aritmético

El componente básico de un circuito aritmético es el sumador en paralelo. Este se construye con un número de circuitos sumadores completos conectados en cascada (véase la figura 3-11). Mediante el control de las entradas de datos al sumador en paralelo es posible obtener tipos diferentes de operaciones aritméticas. El diagrama de bloques de la figura 7-13 demuestra una posible configuración donde un conjunto de entradas al sumador en paralelo es controlado por dos líneas de selección S_1 y S_0. Hay n bits en el circuito aritmético con dos entradas, A y B, y una salida F. Las n entradas de B pasan por un circuito combinatorio a las entradas Y del sumador en paralelo. El acarreo de entrada C_{in} se dirige a la entrada de acarreo del circuito sumador completo en la posición del bit menos significativo. El acarreo de salida C_{out} se realiza desde el sumador completo en la posición más significativa. La salida del sumador en paralelo se calcula a partir de la siguiente suma aritmética

$$F = A + Y + C_{in}$$

donde A es el número binario de n bits en las entradas X, y Y es el número binario de n bits en las entradas Y del sumador en paralelo. C_{in} es el acarreo de entrada que puede ser igual a 0 o 1. Nótese que el símbolo + en la ecuación anterior denota un signo aritmético más.

Con el control del valor de Y mediante las dos entradas de selección S_1 y S_0, es posible obtener diversas operaciones aritméticas. Esto se ilustra en la tabla 7-5. Si se ig-

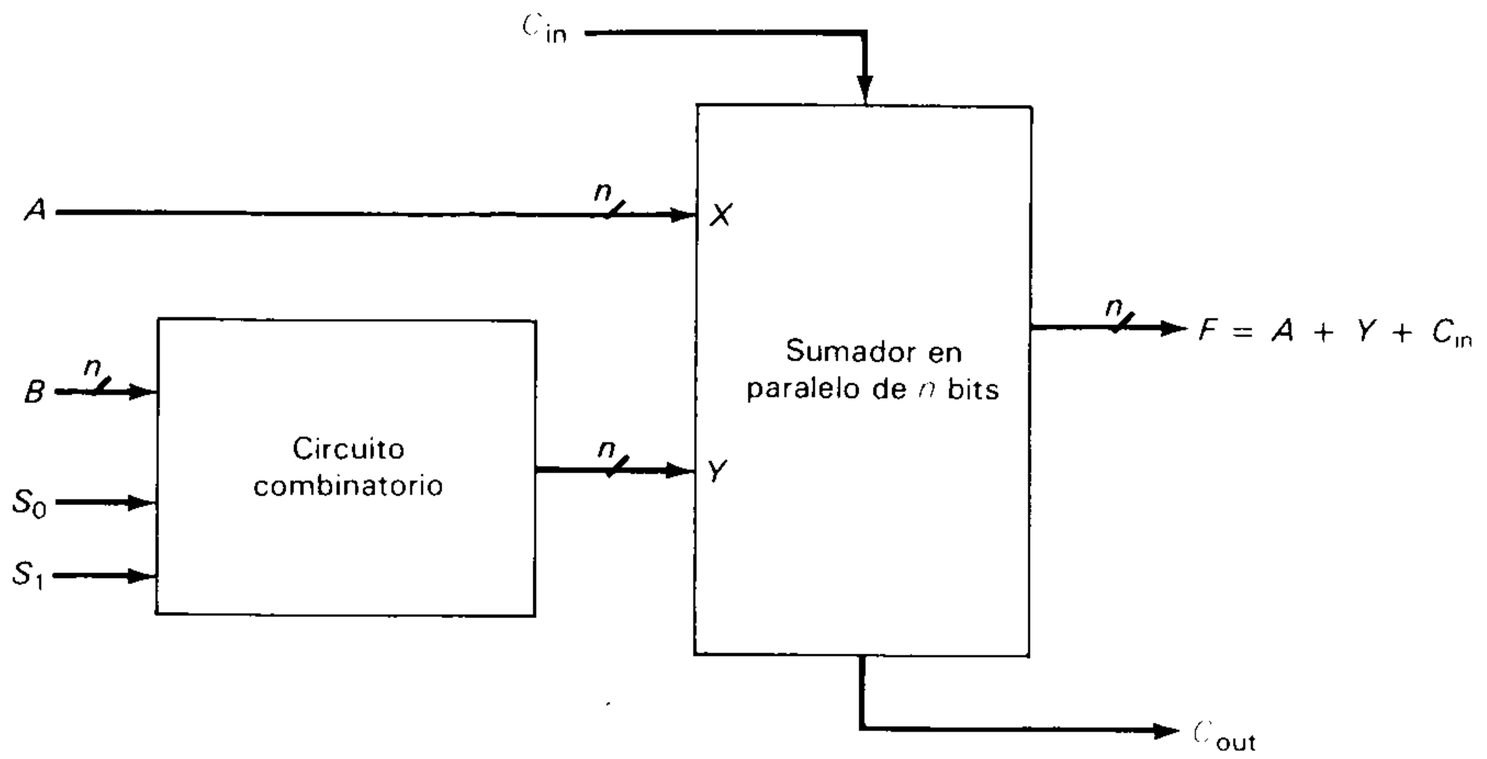


FIGURA 7—13
Diagrama de bloques de un circuito aritmético

noran las entradas de B y se insertan sólo ceros en las entradas Y, la suma de salida se convierte en $F = A + 0 + C_{in}$. Esto da $F = A$ cuando $C_{in} = 0$ y $F = A + 1$ cuando $C_{in} = 1$. En el primer caso se tiene una transferencia directa de la entrada A a la salida F. En el segundo caso, el valor de A se incrementa en 1. En el caso de una adición o suma aritmética directa, es necesario transferir las entradas B a las entradas Y del sumador en paralelo. Esto da $F = A + B$ cuando $C_{in} = 0$. La sustracción o resta aritmética se lleva a cabo cuando Y se convierte en el complemento de B para obtener $F = A + \overline{B} + 1$ cuando $C_{in} = 1$. Esto da A más el complemento a 2's de B que es equivalente a la sustracción. Al insertar sólo unos en las entradas de Y se produce la operación de disminución $F = A - 1$ cuando $C_{in} = 0$. Esto se debe a que un número sólo con unos es igual al complemento a 2's de 1. Por ejemplo, el complemento a 2's del 0001 binario es 1111. Sumar un número A al complemento a 2's produce $F = A +$ complemento a 2's de $1 = A - 1$.

El circuito combinatorio de la figura 7-13 se puede diseñar con n multiplexores y n sumadores completos.

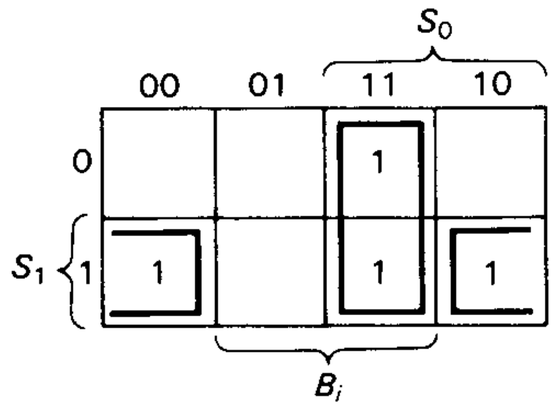
El número de compuertas del circuito combinatorio se puede reducir si, en lugar de utilizar multiplexores, realizamos el diseño lógico de una etapa del circuito combinatorio. Esto se puede hacer como se ilustra en la figura 7-14. La tabla de verdad de una etapa típica i del circuito combinatorio se representa en la figura 7-14(a). Las entradas son S_1, S_0, B_i y la salida es Y_i. Siguiendo los requisitos especificados en la tabla 7-5, hacemos $Y_i = 0$ cuando $S_1S_0 = 00$ y de manera análoga asignamos los

TABLA 7—5
Tabla de funciones de un circuito aritmético

Selección		Entrada	$F = A + Y + C_{in}$	
S_1	S_0	Y	$C_{in} = 0$	$C_{in} = 1$
0	0	sólo ceros	$F = A$ (transferencia)	$F = A + 1$ (incremento)
0	1	B	$F = A + B$ (suma)	$F = A + B + 1$
1	0	$\overline{B}$	$F = A + \overline{B}$	$F = A + \overline{B} + 1$ (sustracción)
1	1	sólo unos	$F = A - 1$ (decremento)	$F = A$ (transferencia)

Entradas			Salida	
S_1	S_0	B_i	Y_i	
0	0	0	0	$Y_i = 0$
0	0	1	0	
0	1	0	0	$Y_i = B_i$
0	1	1	1	
1	0	0	1	$Y_i = \overline{B}_i$
1	0	1	0	
1	1	0	1	$Y_i = 1$
1	1	1	1	

(a) Tabla de verdad



(b) Simplificación del mapa:

$Y_i = B_i S_0 + \overline{B}_i S_1$

FIGURA 7—14
Circuito combinatorio de una etapa del circuito aritmético

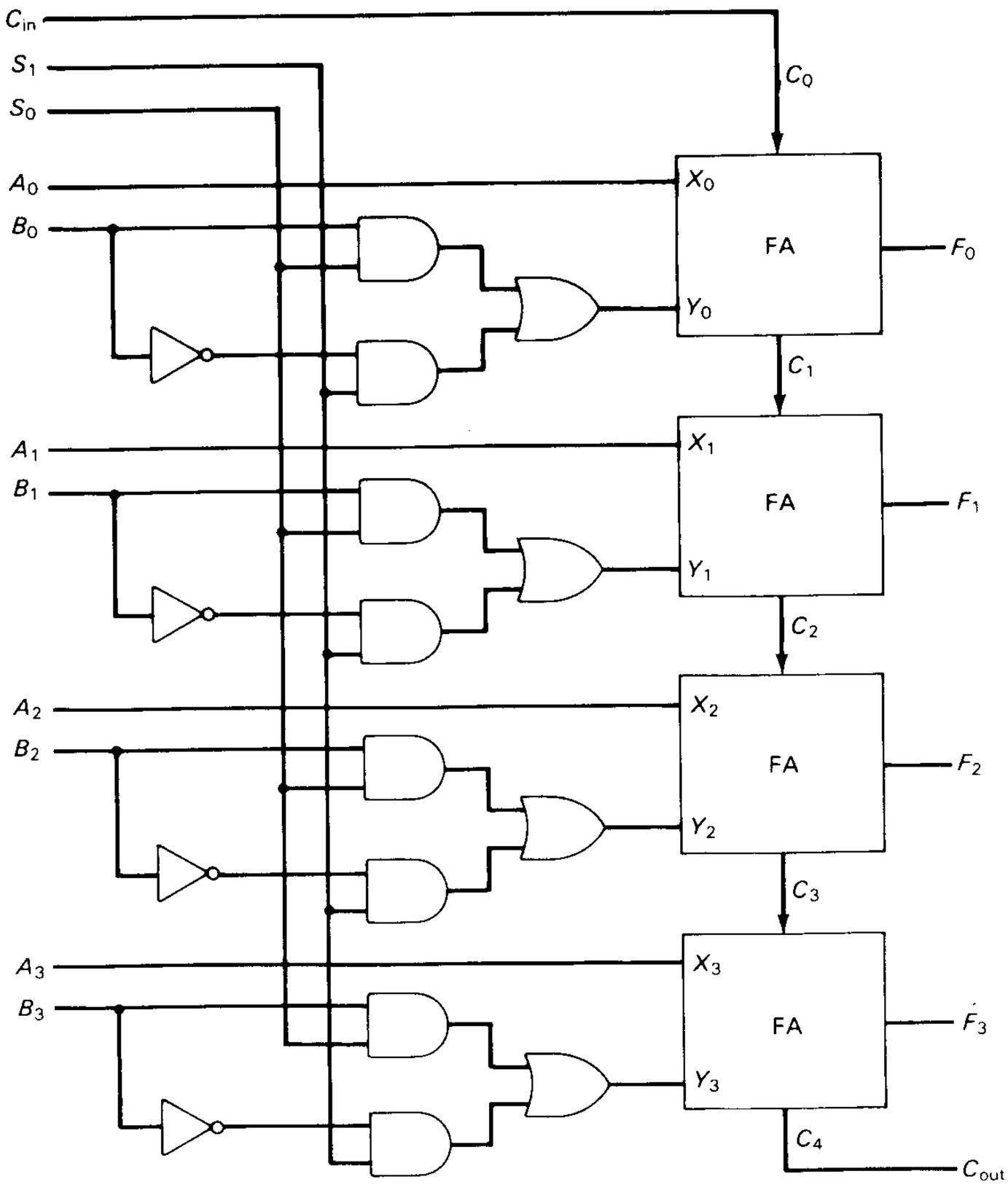


FIGURA 7—15
Diagrama de lógica de un circuito aritmético de 4 bits

otros tres valores de Y_i a cada una de las combinaciones de las variables de selección. La salida Y_i se simplifica en el mapa de la figura 7-14(b).

$$Y_i = B_i S_0 + \overline{B}_i S_1$$

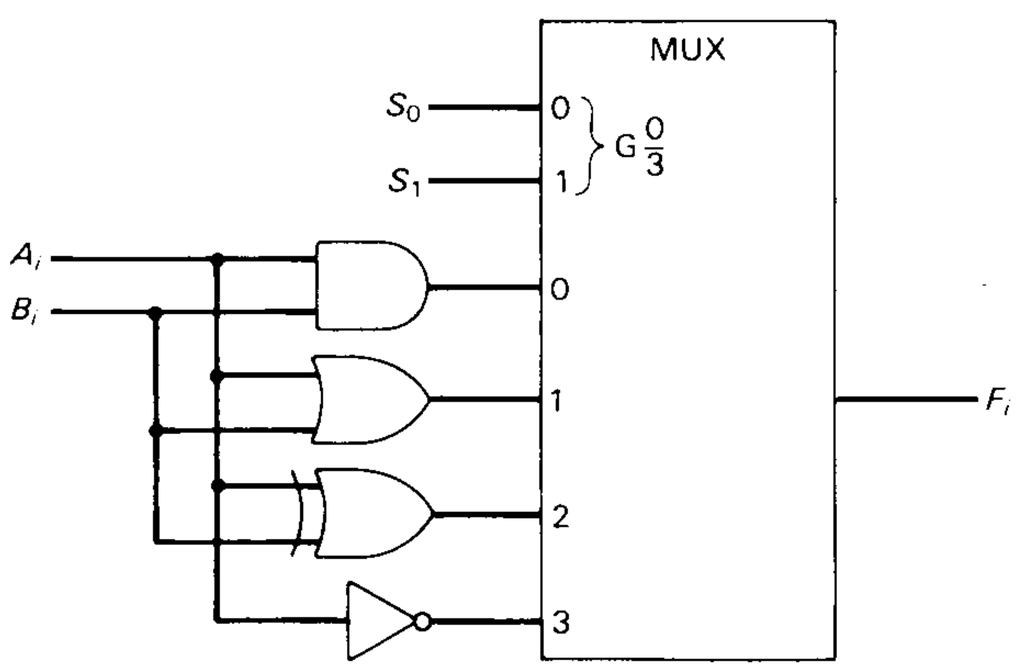
S_1 y S_0 son comunes a las n etapas. Cada etapa i está asociada con la entrada B y la salida Y_i para $i = 0, 1, 2, \ldots, n - 1$.

El diagrama de lógica de un circuito aritmético de 4 bits se presenta en la figura 7-15. Los cuatro circuitos sumadores completos (FA) constituyen el sumador en paralelo. El acarreo en la primera etapa es el acarreo de entrada C_{in}. Los demás acarreos se conectan internamente de una etapa a la siguiente. Las variables de selección son S_1, S_0 y C_{in}. Las variables S_1 y S_0 controlan todas las entradas Y de los sumadores completos de acuerdo con la función booleana que se obtiene en la figura 7-14(b). C_{in} suma 1 a la suma cuando es igual a 1. Las ocho operaciones aritméticas del circuito como función de S_1, S_0 y C_{in} se ilustran en la tabla 7-5.

Circuito de lógica o lógico

Las microoperaciones lógicas manipulan los bits de los operandos tratando a cada bit de un registro como a una variable binaria. Hay cuatro operaciones lógicas básicas de las cuales se pueden obtener las demás. Ellas son AND, OR, XOR (OR excluyente) y el complemento.

La figura 7-16(a) muestra una etapa del circuito lógico. Este consta de cuatro compuertas y un multiplexor. Cada una de las cuatro operaciones lógicas se genera a través de una compuerta que realiza la lógica requerida. Las salidas de las compuertas se aplican a un multiplexor con dos variables de selección, S_1 y S_0. Estas variables de selección eligen una de las entradas de datos del multiplexor y dirigen su valor a la salida. El diagrama presenta una etapa típica con el subíndice i. Para un circuito lógico con n bits, el diagrama debe repetirse n veces para $i = 0, 1, 2, \ldots, n - 1$. Las variables de selección se aplican a todas las etapas. La tabla de funciones de la figura 7-16(b) ilustra las operaciones lógicas que se obtienen para cada combinación de las variables de selección.



(a) Diagrama de lógica

S_1	S_0	Salida	Operación
0	0	$F = A \wedge B$	AND
0	1	$F = A \vee B$	OR
1	0	$F = A \oplus B$	XOR
1	1	$F = \overline{A}$	Complemento

(b) Tabla de funciones

FIGURA 7—16
Una etapa de un circuito lógico

Unidad de aritmética y lógica

El circuito lógico se puede combinar con el circuito aritmético a fin de producir una unidad de aritmética y lógica. Las variables de selección S_1 y S_0 pueden ser comunes a ambos circuitos siempre que se utilice una tercera variable de selección para diferenciar entre los dos. La configuración de una etapa de la ALU se ilustra en la figura 7-17. Las salidas de los circuitos aritmético y lógico en cada etapa se aplican a un

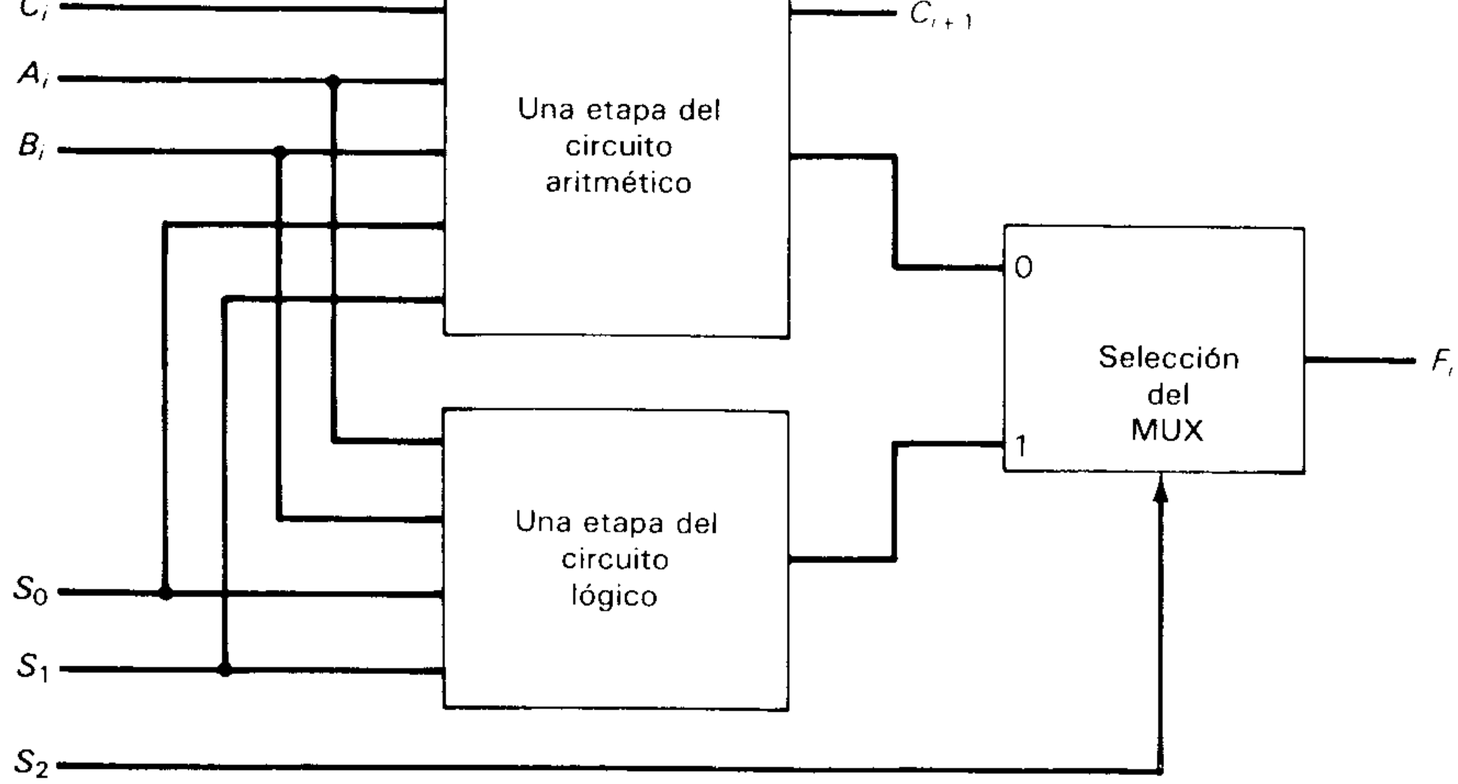


FIGURA 7—17
Una etapa de una ALU

multiplexor de 2 × 1 con variable de selección S_2. Cuando $S_2 = 0$, se selecciona la salida aritmética y cuando $S_2 = 1$, se selecciona la salida lógica. Nótese que el diagrama muestra sólo una etapa común de la ALU. El circuito de la figura 7-17 debe repetirse n veces para una ALU de n bits. El acarreo de salida C_{i+1} de una etapa arit-

TABLA 7—6
Tabla de funciones de la ALU

Selección de la operación					
S_2	S_1	S_0	C_{in}	Operación	Función
0	0	0	0	$F = A$	Transferencia de A
0	0	0	1	$F = A + 1$	Incremento de A
0	0	1	0	$F = A + B$	Adición
0	0	1	1	$F = A + B + 1$	Adición con acarreo
0	1	0	0	$F = A + \overline{B}$	A más complemento a 1's de B
0	1	0	1	$F = A + \overline{B} + 1$	Sustracción
0	1	1	0	$F = A - 1$	Decremento de A
0	1	1	1	$F = A$	Transferencia de A
1	0	0	0	$F = A \wedge B$	AND
1	0	1	0	$F = A \vee B$	OR
1	1	0	0	$F = A \oplus B$	XOR
1	1	1	0	$F = \overline{A}$	Complemento de A

mética dada debe conectarse al acarreo de entrada C_i de la siguiente etapa en secuencia. El acarreo de entrada a la primera etapa es el acarreo de entrada C_{in} que produce una variable de selección para las operaciones aritméticas.

La ALU que se especifica en la figura 7-17 efectúa ocho operaciones aritméticas y cuatro lógicas. Cada operación se selecciona a través de las variables S_2, S_1, S_0 y C_{in}. El acarreo de entrada C_{in} se utiliza para seleccionar sólo una operación aritmética.

La tabla 7-6 ilustra las 12 operaciones de la ALU. Las primeras ocho son operaciones aritméticas y se seleccionan con $S_2 = 0$. Las cuatro siguientes son operaciones lógicas y se seleccionan con $S_2 = 1$. El acarreo de entrada no tiene efecto durante las operaciones lógicas y se marca con un 0 por comodidad.

7-7 UNIDAD DE CORRIMIENTO

La unidad de corrimiento conectada al sistema del bus transfiere la salida de la ALU al bus de salida. La unidad de corrimiento transfiere la información corriéndola a la derecha o a la izquierda. Deben establecerse condiciones para realizar una transferencia directa sin corrimiento de la ALU al bus de salida. La unidad de corrimiento hace las operaciones de corrimiento que comúnmente no se tienen a disposición en la ALU.

Una elección obvia de una unidad de corrimiento sería un registro de corrimiento con carga en paralelo. La información de la ALU se puede transferir al registro en paralelo y después correrse a la derecha o a la izquierda. En este tipo de configuración, se necesita un pulso de reloj para cargar la salida de la ALU en el registro de corrimiento y se necesita otro pulso para el corrimiento. Estos dos pulsos son adi-

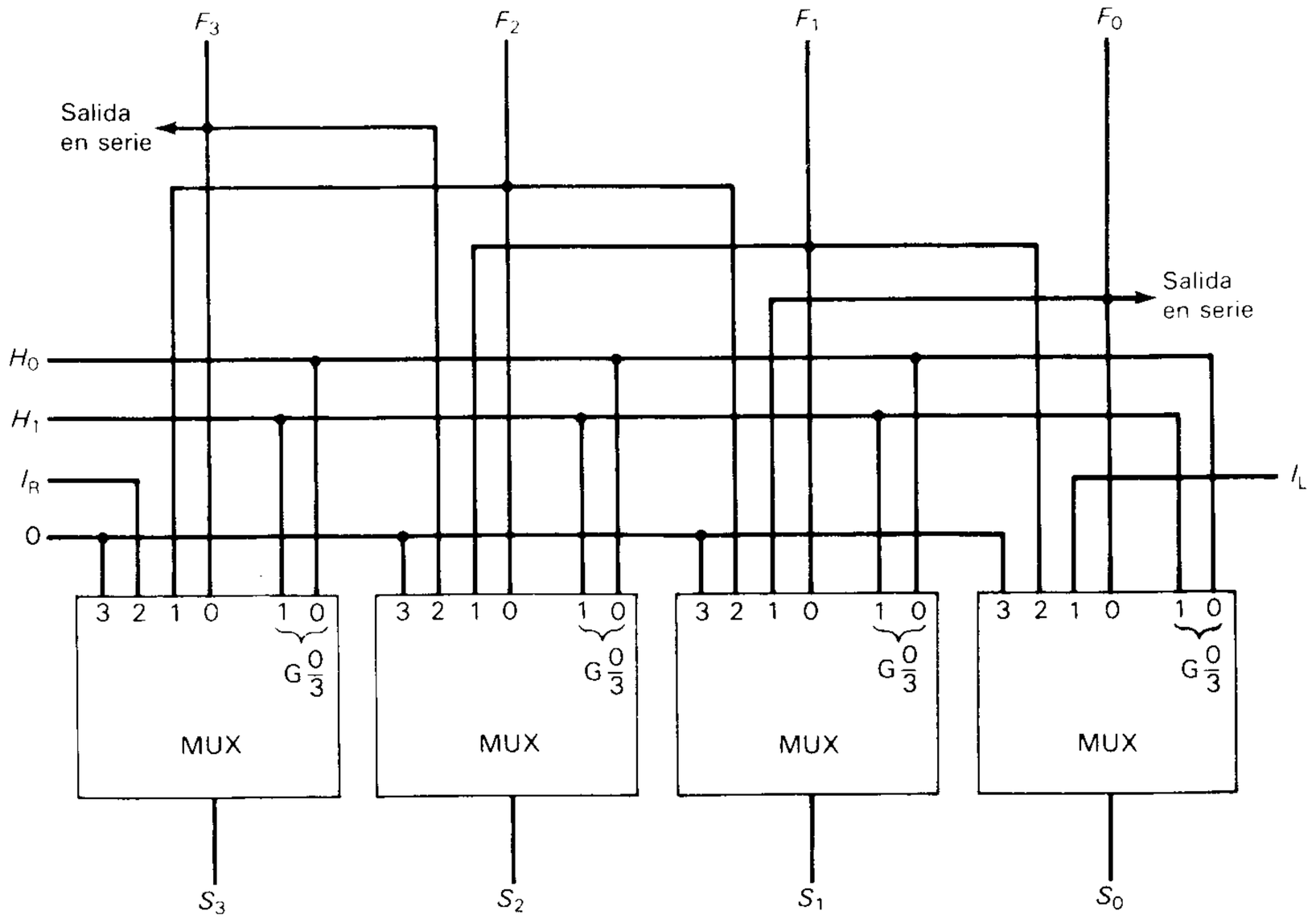


FIGURA 7—18
Unidad de corrimiento de 4 bits

TABLA 7—7
Tabla de funciones de la unidad de corrimiento

H_1	H_0	Operación	Función
0	0	$S \leftarrow F$	Transferencia de F a S (no hay corrimiento)
0	1	$S \leftarrow$ shl F	Corrimiento a la izquierda de F a S
1	0	$S \leftarrow$ shr F	Corrimiento a la derecha de F a S
1	1	$S \leftarrow 0$	Transferencia de ceros a S

cionales al pulso de reloj que se requiere para transferir la información del registro de corrimiento al registro destino seleccionado.

La transferencia de un registro fuente a un registro destino a través de la ALU y la unidad de corrimiento puede realizarse con sólo un pulso de reloj cuando la ALU y la unidad de corrimiento se construyen como circuitos combinatorios. En un circuito combinatorio, las señales se propagan a través de compuertas sin que se necesite un pulso de reloj. Por lo tanto, el único reloj que se necesita en el sistema del bus procesador es para cargar los datos del bus de salida en el registro destino seleccionado.

Una unidad de corrimiento de un circuito combinatorio se puede construir con multiplexores, según se indica en la figura 7-18. Las variables de selección H_1 y H_0 se aplican a los cuatro multiplexores para seleccionar el tipo de operación en la unidad de corrimiento. Con $H_1H_0 = 00$, no se ejecuta ningún corrimiento y las entradas de F van directamente a las salidas de la unidad de corrimiento en S. Los dos valores siguientes de las variables de selección producen una operación de corrimiento a la izquierda y a la derecha. Cuando $H_1H_0 = 11$, los multiplexores seleccionan las entradas anexas al 0 lógico y todas las salidas S son iguales a 0. La tabla 7-7 presenta un resumen de la operación de la unidad de corrimiento.

El diagrama de la figura 7-18 presenta sólo cuatro etapas de la unidad de corrimiento que, desde luego, debe tener n etapas en un sistema con n líneas en paralelo. Las entradas I_R e I_L son las entradas en serie del corrimiento a la derecha y a la izquierda, respectivamente. Se puede emplear otra variable de selección para especificar lo que hay en I_R e I_L durante el corrimiento. Por ejemplo, una tercera variable de selección, H_2, cuando se encuentra en un estado puede seleccionar un 0 para la entrada en serie durante el corrimiento. Cuando H_2 se encuentra en el otro estado, los datos se pueden trasladar (rotar) junto con el valor en el bit de estado del acarreo. En esta forma, un acarreo producido durante una operación de adición de la ALU puede correrse a la derecha y a la posición del bit más significativo de un registro.

Unidad de corrimiento de tambor

En algunas aplicaciones del procesador se deben correr datos más de una vez durante una sola operación. Una unidad de corrimiento de tambor es un circuito que corre los bits de datos de entrada un número de posiciones que prescribe el valor binario de un conjunto de líneas de selección. El corrimiento es una rotación cíclica que significa que la información binaria de entrada se corren en una dirección y que el bit que proviene del extremo más significativo de la unidad de corrimiento es llevado de regreso a la posición menos significativa.

En la figura 7-19 se ilustra una unidad de corrimiento de tambor de cuatro bits, la cual consta de cuatro multiplexores con dos líneas de selección comunes S_1 y S_0. Las

variables de selección determinan el número de posiciones que se correrán los datos de entrada a la izquierda por rotación. Cuando $S_1S_0 = 00$, no ocurre corrimiento y las entradas forman una trayectoria directa a las salidas. Cuando $S_1S_0 = 01$, la información binaria de entrada se hace girar una vez y D_0 se dirige a Y_1, la entrada D_1 se hace girar dos posiciones y cuando $S_1S_0 = 11$, la rotación se hace en tres posiciones de bits.

En la tabla 7-8 se presenta la tabla de funciones de la unidad de corrimiento de tambor de 4 bits. Para cada valor binario de las variables de selección, la tabla ilustra las entradas que se dirigen a la salida correspondiente. En consecuencia, para hacer girar tres posiciones, S_1S_0 debe ser igual a 11; y D_0 se dirige a Y_3, D_1 a Y_0, D_2 se dirige a Y_1 y D_3 va a Y_2.

Una unidad de corrimiento de tambor con 2^n líneas de entrada y salida requiere 2^n multiplexores, donde cada uno tiene 2^n entradas de datos y n entradas de selección. El número de posiciones de rotación lo prescriben las variables de selección y puede ser de 0 (sin corrimiento) a $2^n - 1$ posiciones.

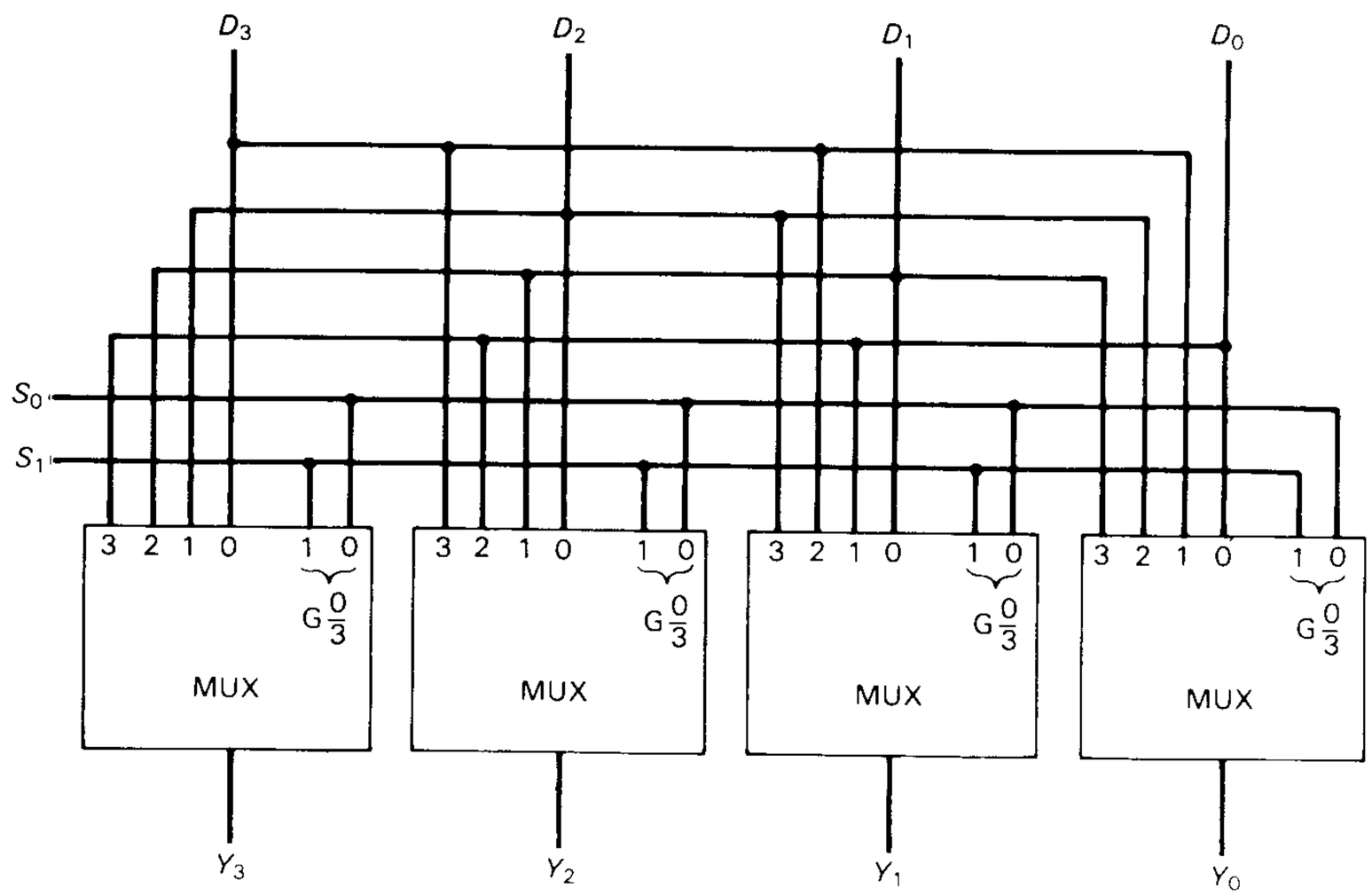


FIGURA 7—19
Unidad de corrimiento de tambor de 4 bits

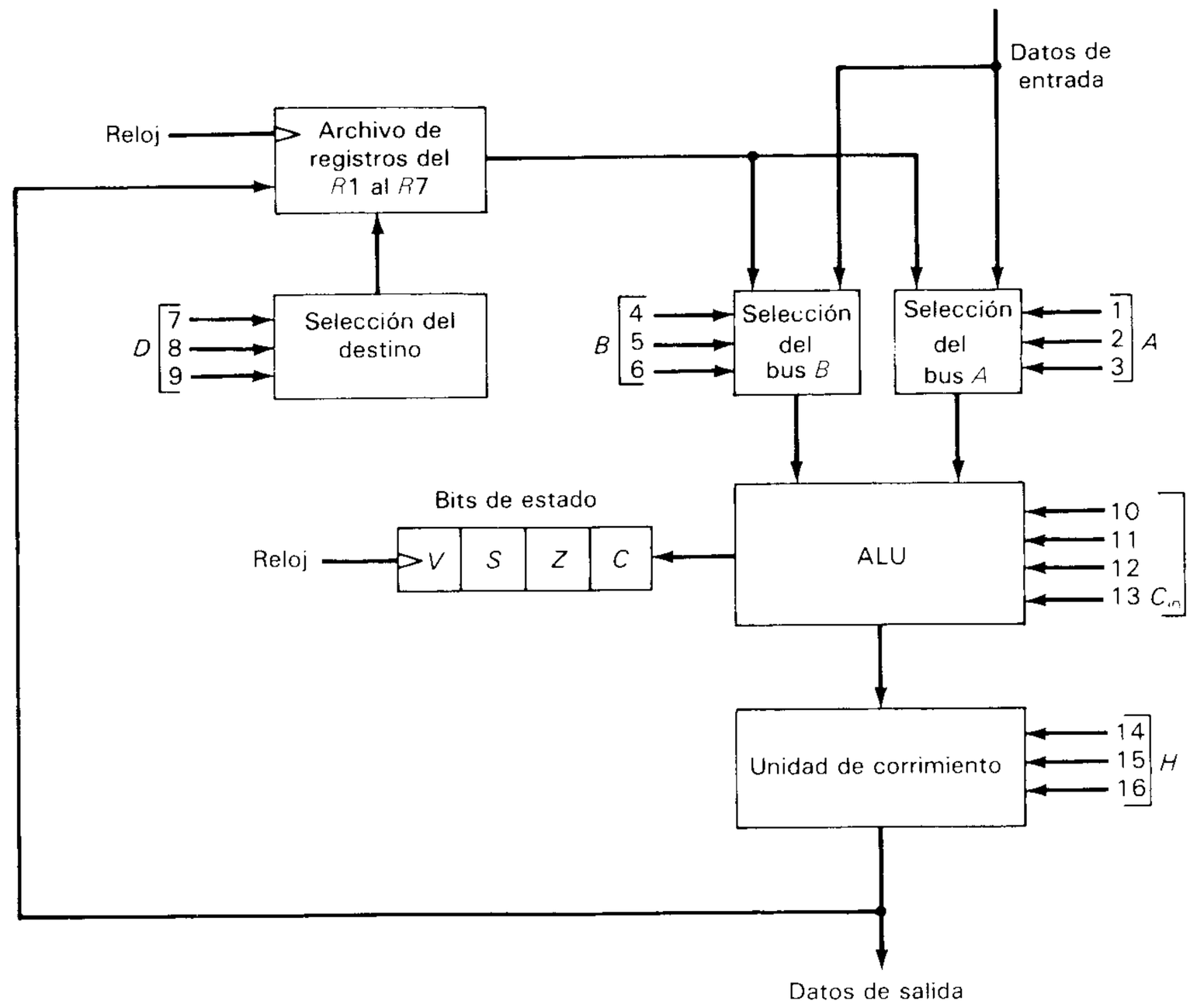
TABLA 7—8
Tabla de funciones de la unidad de corrimiento de tambor de 4 bits

Selección		Salida				
S_1	S_0	Y_3	Y_2	Y_1	Y_0	Operación
0	0	D_3	D_2	D_1	D_0	No hay corrimiento
0	1	D_2	D_1	D_0	D_3	Rotación una vez
1	0	D_1	D_0	D_3	D_2	Rotación dos veces
1	1	D_0	D_3	D_2	D_1	Rotación tres veces

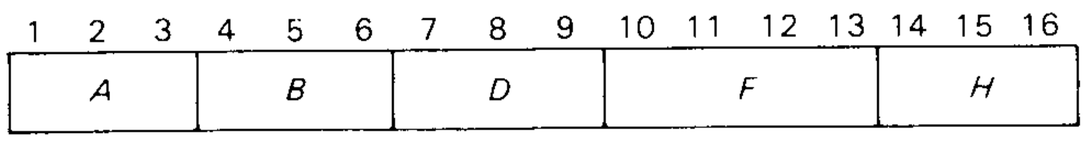
7-8 PALABRA DE CONTROL

Las variables de selección de una unidad procesadora controlan las microoperaciones que se ejecutan dentro de la unidad durante cualquier transición de pulsos de reloj dada. Las variables de selección controlan los buses, la ALU, la unidad de corrimiento y el registro destino. Ahora demostraremos por medio de un ejemplo la forma en que las variables de control seleccionan las microoperaciones. El ejemplo define un procesador específico junto con sus variables de selección. La elección de variables de control para algunas microoperaciones típicas se demostrará en seguida.

En la figura 7-20 se ilustra un diagrama de bloques de una unidad procesadora que tiene un archivo de registros de siete elementos, del $R1$ al $R7$. Las salidas de los siete registros pasan por dos conjuntos de multiplexores para seleccionar las entradas a la ALU. Los mismos multiplexores seleccionan los datos de entrada de una fuente externa (como una unidad de memoria). La salida de la ALU pasa por una unidad de co-



(a) Diagrama de bloques



(b) Palabra de control

FIGURA 7—20
Unidad procesadora con variables de control

TABLA 7—9
Codificación de la palabra de control de la unidad procesadora

Código binario	Función de campos de selección					
	A	B	D	F con $C_{in} = 0$	F con $C_{in} = 1$	H
000	Entrada	Entrada	Ninguno	$F = A$	$F = A + 1$	No hay corrimiento
001	$R1$	$R1$	$R1$	$F = A + \overline{B}$	$F = A + B + 1$	SHL
010	$R2$	$R2$	$R2$	$F = A + \overline{B}$	$F = A - B$	SHR
011	$R3$	$R3$	$R3$	$F = A - 1$	$F = A$	Bus = 0
100	$R4$	$R4$	$R4$	$F = A \wedge B$	—	—
101	$R5$	$R5$	$R5$	$F = A \vee B$	—	ROL
110	$R6$	$R6$	$R6$	$F = A \oplus B$	—	ROR
111	$R7$	$R7$	$R7$	$F = \overline{A}$	—	—

rrimiento y llega al bus de salida. La salida de la unidad de corrimiento se puede transferir a cualquiera de los registros y además puede dirigirse a un destino externo. La ALU proporciona los datos binarios de los cuatro bits de estado: C (acarreo), Z (cero), S (signo) y V (desbordamiento).

Hay 16 entradas de selección binarias en la unidad y su valor combinado especifica una *palabra de control*. La palabra de control de 16 bits se define en la figura 7-20(b), y consta de cinco partes llamadas campos, donde cada campo está designado por un símbolo alfabético (letra). Cuatro de los campos contienen tres bits cada uno, y un campo tiene cuatro bits. Los tres bits de A seleccionan un registro fuente para una entrada de la ALU. Los tres bits de B seleccionan un registro para la segunda entrada de la ALU. Los tres bits de D seleccionan un registro destino. Los cuatro bits de F seleccionan una de 12 operaciones en la ALU. Los tres bits de H seleccionan el tipo de corrimiento en la unidad de corrimiento. La palabra de control de 16 bits, cuando se aplica a las entradas de selección, especifica una microoperación en particular.

Las funciones de todas las variables de selección se especifican en la tabla 7-9. El código binario de 3 bits que se ilustra en la tabla especifica el código binario de cada uno de los cinco campos. El registro que seleccionan los campos A, B y D es el que tiene el número decimal equivalente al número binario del código. Cuando el campo A o B es 000, el multiplexor correspondiente selecciona los datos de entrada externos. Cuando D = 000, no se selecciona ningún registro destino sino que el contenido del bus de salida está disponible para las salidas externas. La codificación del campo F depende del valor del acarreo de entrada C_{in}. Los cuatro bits del campo F para cada operación se determinan a partir de los tres bits de la primera columna correspondiente de la tabla y el bit de C_{in}. Las operaciones que se citan de la ALU son las mismas que las que se especifican en la tabla 7-6.

Las cuatro primeras entradas del código del campo H especifican las operaciones de corrimiento de la tabla 7-7. El campo H tiene una tercera variable de selección para distinguir entre un corrimiento lógico y un corrimiento por rotación. Las operaciones SHL (corrimiento a la izquierda) y SHR (corrimiento a la derecha) que se citan en la tabla son corrimientos lógicos, lo que significa que 0 se aplica a sus entradas en serie

TABLA 7—10
Ejemplos de microoperaciones del procesador

Microoperación	Designación simbólica					Palabra de control				
	A	B	D	F	H	A	B	D	F	H
$R1 \leftarrow R2 - R3$	$R2$	$R3$	$R1$	$F = A - B$	No hay corrimiento	010	011	001	0101	000
$R4 \leftarrow \text{shr}\ (R5 + R6)$	$R5$	$R6$	$R4$	$F = A + B$	SHR	101	110	100	0010	010
$R7 \leftarrow R7 + 1$	$R7$	—	$R7$	$F = A + 1$	No hay corrimiento	111	000	111	0001	000
$R1 \leftarrow R2$	$R2$	—	$R1$	$F = A$	No hay corrimiento	010	000	001	0000	000
Salida $\leftarrow R3$	$R3$	—	Ninguno	$F = A$	No hay corrimiento	011	000	000	0000	000
$R4 \leftarrow \text{rol}\ R4$	$R4$	—	$R4$	$F = A$	ROL	100	000	100	0000	100
$R5 \leftarrow 0$	—	—	$R5$	—	Bus = 0	000	000	101	0000	011

correspondientes. Cuando la tercera variable de selección del campo H es igual a 1, esto indica un corrimiento por rotación. Esto se designa en la tabla por medio de los símbolos ROL (rotación a la izquierda) y ROR (rotación a la derecha). La operación ROL corre los datos de entrada a la izquierda e inserta el valor de la salida en serie en la entrada en serie. Lo mismo se aplica a la operación ROR, salvo que el corrimiento es a la derecha.

Se necesita una palabra de control de 16 bits para especificar una microoperación para el procesador. La palabra de control de una microoperación dada se puede obtener a partir de las variables de selección. Por ejemplo, la microoperación de sustracción, dada por la instrucción

$$R1 \leftarrow R2 - R3$$

especifica $R2$ para la entrada A de la ALU, $R3$ para la entrada B de dicha ALU, una operación de la ALU $F = A - B$, no se activa la unidad de corrimiento, y $R1$ para el registro destino. Por lo tanto, la palabra de control es especificada por los cinco campos y el valor binario correspondiente de cada campo se obtiene a partir de la codificación que se representa en la tabla 7-9. La palabra de control binaria de la microoperación de sustracción es 0100110010101000 y se obtiene como sigue:

Campo:	A	B	D	F	H
Símbolo:	$R2$	$R3$	$R1$	$F = A - B$	No hay corrimiento
Palabra de control:	010	011	001	0101	000

La palabra de control de esta microoperación y algunas otras, se ilustran en la tabla 7-10.

El ejemplo que sigue en la tabla es una microoperación de suma y corrimiento dada por la instrucción

$$R4 \leftarrow \text{shr}\ (R5 + R6)$$

Esta especifica una adición aritmética para la ALU y una operación de corrimiento a la derecha para la unidad de corrimiento. La suma que se genera en la ALU se corre a la derecha, dividiéndola entre 2. El resultado de la operación consiste en producir el valor promedio de los dos números binarios sin signo almacenados en los registros $R5$ y $R6$ (siempre que no haya acarreo). Si se conocen bien los símbolos en cada campo, la palabra de control en binario se obtiene obviamente como se indica en la tabla 7-10.

BIBLIO

PROBL

Las microoperaciones de incremento y transferencia no utilizan la entrada B de la ALU. En estos casos, el campo B se marca con una diagonal. Por comodidad, asignamos 000 a cualquier campo no usado cuando se formule la palabra de control binaria, aunque se puede emplear cualquier otro número binario. Para colocar el contenido de un registro en las terminales de salida, situamos el contenido del registro en la entrada A de la ALU, pero ninguno de los registros se selecciona para aceptar los datos. La operación de la ALU $F = A$ coloca la información del registro en las terminales de salida. Para correr o rotar el contenido de un registro sin efectuar una operación en la ALU, debemos especificar una operación de la ALU $F = A$. Esto coloca el contenido de la entrada A en la salida F de la ALU sin ninguna variación. Para poner un registro en 0, el bus de salida se hace igual a todos ceros con H = 011. El campo destino D se hace igual al código del registro.

Se observa en estos ejemplos que se pueden generar muchas otras microoperaciones en la unidad procesadora. La forma más eficiente de generar palabras de control con un número grande de bits consiste en almacenarlas en una unidad de memoria. Una unidad de memoria que almacena palabras de control se denomina *memoria de control*. La salida de la memoria de control se aplica a las entradas de selección del procesador. Mediante la lectura de palabras de control consecutivas de la memoria, es posible iniciar la secuencia de microoperaciones deseada para el procesador. Este tipo de organización de control, llamada *microprogramación*, se analiza con mayor detalle en el capítulo siguiente.

BIBLIOGRAFIA

1. BOOTH, T. L. *Introduction to Computer Engineering Hardware and Software Design*. 3a./edic. Nueva York: Wiley, 1984.
2. DIETMEYER, D. L. *Logic Design of Digital Systems*. 2a./edic. Boston: Allyn & Bacon, 1978.
3. ERCEGOVAC, M. y LANG, T. *Digital Systems and Hardware/Firmware Algorithms*. Nueva York: Wiley, 1985.
4. GOSLING, J. B., *Design of Arithmetic Units for Digital Computers*. Nueva York: Springer-Verlag, 1980.
5. KLINE, R. M. *Structured Digital Design*. Englewood Cliffs: Prentice Hall, 1983.
6. MANO, M. M. *Computer System Architecture*. Englewood Cliffs: Prentice-Hall, 1982.

PROBLEMAS

7-1 Ilustre el diagrama de bloque del *hardware* que ejecuta la siguiente instrucción de transferencia de registros:

$$T_3\colon R2 \leftarrow R1,\ R1 \leftarrow R2$$

7-2 Las salidas de cuatro registros $R0$, $R1$, $R2$ y $R3$ están conectadas a través de multiplexores a las entradas de un quinto registro $R5$. Cada registro tiene 8 bits de longitud. Las transferencias requeridas están regidas por cuatro variables de sincronización, de la T_0 a la T_3, de la manera siguiente:

$$T_0\text{: } R5 \leftarrow R0$$

$$T_1\text{: } R5 \leftarrow R1$$

$$T_2\text{: } R5 \leftarrow R2$$

$$T_3\text{: } R5 \leftarrow R3$$

Las variables de sincronización son mutuamente excluyentes y sólo una variable puede ser igual a 1 en cualquier momento dado mientras que las otras tres son igual a 0. Trace un diagrama de bloques donde represente la aplicación en *hardware* de las transferencias de registros. Incluya las conexiones necesarias de las cuatro variables de sincronización a las líneas de selección de los multiplexores y a la entrada de carga del registro $R5$.

7-3 Utilizando dos registros de 4 bits, $R1$ y $R2$, un multiplexor cuádruple de 2 a 1 líneas con habilitación y cuatro inversores, trace el diagrama de circuitos que ejecutan las siguientes instrucciones:

$T_1\text{: } R2 \leftarrow R1$	**Transferencia de *R*1 a *R*2**
$T_2\text{: } R2 \leftarrow \overline{R2}$	**Complemento de *R*2**
$T_3\text{: } R2 \leftarrow 0$	**Puesta a ceros de *R*2 sincrónico con el reloj**

Las tres variables de sincronización de la T_1 a la T_3, son mutuamente excluyentes y sólo una variable puede ser igual a 1 en cualquier momento dado. Utilice diagramas gráficos estándar como en la figura 5-3(b) para los registros y la figura 3-31(b) para el multiplexor.

7-4 Trace el diagrama de bloque del *hardware* que ejecuta las instrucciones siguientes:

$$XT_1 + YT_2\text{: } AR \leftarrow AR + BR$$

Donde AR y BR son dos registros de n bits, X y Y son variables de control y T_1 y T_2 son variables de sincronización. Incluya las compuertas lógicas para la función de control. (Recuerde que el símbolo + designa una operación OR de una función de control o booleana, pero representa un signo más aritmético en una microoperación.)

7-5 Considere las siguientes instrucciones de transferencia de registros para los registros de 4 bits $R1$ y $R2$.

$$XT\text{: } R1 \leftarrow R1 + R2$$

$$\overline{X}T\text{: } R1 \leftarrow R2$$

Cuando la variable de sincronización $T = 1$, el contenido de $R2$ se suma al contenido de $R1$ si $X = 1$ o bien el contenido de $R2$ se transfiere a $R1$ si $X = 0$. Trace un diagrama de lógica donde ilustre la ejecución de las dos instrucciones. Utilice símbolos gráficos estándar para los registros de 4 bits [figura 5-3(b)], el sumador de 4 bits (figura 3-28) y el multiplexor que selecciona las entradas a $R1$ [figura 3-31(b)]. En el diagrama, muestre cómo las variables de control X y T seleccionan las entradas a $R1$.

7-6 Utilizando un contador de 4 bits con carga en paralelo como en la figura 5-19 y un sumador de 4 bits como en la figura 3-28, trace el diagrama de lógica con símbolos gráficos estándar que ejecute las instrucciones siguientes:

$T_1\text{: } R1 \leftarrow R1 + R2$	**Suma de *R*2 a *R*1**
$\overline{T}_1T_2\text{: } R1 \leftarrow R1 + 1$	**Incremento de *R*1**

7-7 Un sistema digital tiene tres registros: *AR, BR* y *PR*. Tres flip-flops generan las variables de control para el sistema: *S* es un flip-flop que es iniciado por una señal de inicio externa para dar inicio a la operación del sistema; *F* y *R* son dos multivibradores que se utilizan para dar secuencia o sucesión a las microoperaciones cuando el sistema está en operación. Un cuarto flip-flop D es iniciado por el sistema cuando se completa la operación. La operación del sistema digital la describen las siguientes instrucciones de transferencia de registros:

$$S\text{: } PR \leftarrow 0,\ S \leftarrow 0,\ D \leftarrow 0,\ F \leftarrow 1$$

$$F\text{: } F \leftarrow 0,\ \text{si } (AR = 0) \quad \text{entonces } (D \leftarrow 1) \text{ de lo contrario } (R \leftarrow 1)$$

$$R\text{: } PR \leftarrow PR + BR,\ AR \leftarrow AR - 1,\ R \leftarrow 0,\ F \leftarrow 1$$

(a) Muestre que el sistema digital multiplica el contenido de los registros *AR* y *BR* y coloca el producto en el registro *PR*.
(b) Trace un diagrama de bloque de la ejecución en *hardware*. Incluya una señal de entrada de inicio para iniciar el flip-flop *S* y una señal de salida del flip-flop D.

7-8 Suponga que los registros *R*1 y *R*2 de la figura 7-4 contienen dos números sin signo de *n* bits. Cuando la entrada de selección *X* es igual a 1, el circuito sumador-restador realiza la operación aritmética *R*1 + complemento de 2 de *R*2. Esta suma y el acarreo de salida C_n se transfieren a *R*1 y *C* cuando $T_1 = 1$ y el reloj pasa por una transición positiva.
(a) Demuestre que si $C = 1$, entonces el valor que se transfiere a *R*1 es igual a *R*1 — *R*2. Pero si $C = 0$, el valor que se transfiere a *R*1 es el complemento a 2's de (*R*2 — *R*1). (Refiérase a la sección 1-4 y al ejemplo 1-9 para conocer el procedimiento de sustracción de números sin signo con complemento a 2's.)
(b) Indique cómo se puede utilizar el valor en el bit *C* a fin de detectar un préstamo después de realizarse la sustracción de dos números sin signo.

7-9 Convierta los siguientes números decimales en binarios con signo con 8 bits cada uno (incluyendo al signo). Realice la operación con los números binarios con signo y verifique los dos últimos acarreos. Indique si los acarreos señalan un desbordamiento. Recuerde que los números de 8 bits tienen un intervalo de capacidad de +127 a —128.
(a) (+65) + (+36) (c) (−36) + (−90)
(b) (+65) − (−90) (d) (−65) − (+90)

7-10 Realice las operaciones lógicas AND, OR y XOR con los números de 8 bits 10011100 y 10101010.

7-11 Dado el valor de 16 bits 01011010 11000011. ¿Qué operación se debe efectuar
(a) para poner en 0 los últimos ocho bits?
(b) para iniciar a 1 los primeros ocho bits?
(c) para complementar los ocho bits de enmedio?

7-12 La siguiente microoperación lógica se realiza con los registros *AR* y *BR*.

$$AR \leftarrow AR \wedge \overline{BR} \qquad \textbf{Complemento de } \textit{BR} \textbf{ y AND a } \textit{AR}$$

Determine cómo manipulan los bits de *BR* a los de *AR*.

7-13 Comenzando a partir de los ocho bits 10110010, demuestre los valores que se obtienen después de cada una de las microoperaciones de corrimiento que se presentan en la tabla 7-4.

7-14 Demuestre que la instrucción

$$R1 \leftarrow R1 + R1$$

es la misma que el corrimiento lógico a la izquierda del registro *R*1.

7-15 (a) Represente el número —26 como un número con signo de 8 bits. Demuestre cómo se puede multiplicar y dividir el número por y entre 2 por medio de la microoperación de corrimiento aritmético. Indique si hay un desbordamiento.
(b) Repita la operación con el número binario equivalente —94 y +94.

7-16 Sean S_1 y S_0 las variables de selección del multiplexor del sistema de bus de la figura 7—6 y sean D_1 y D_0 las variables de selección del decodificador destino.
(a) Determine la transferencia que ocurre cuando las variables de control $S_1S_0D_1D_0$ son iguales a (1) 0001, (2) 0110, (3) 1111.
(b) Dé la selección de 4 bits de las siguientes transferencias: (1) $R0 \leftarrow R1$; (2) $R2 \leftarrow R1$; (3) $R3 \leftarrow R2$.

7-17 Trace el diagrama de un sistema de bus similar al que se muestra en la figura 7-6 pero utilice buffers de tres estados y un decodificador en vez de los multiplexores.

7-18 Las siguientes transferencias de memoria se especifican para el sistema de la figura 7-10.
(a) $D2 \leftarrow M[A3]$ (b) $M[A1] \leftarrow D0$
Especifique la operación de la memoria y determine las variables de selección binarias para los dos buffers del bus y el decodificador.

7-19 Una unidad procesadora como la de la figura 7-11 tiene 30 registros. ¿Cuántas líneas de selección se necesitan para cada conjunto de multiplexores y para el decodificador?

7-20 Dada una ALU de 8 bits con salidas de la F_0 a la F_7 y acarreos C_7 y C_8, muestre el circuito lógico para iniciar los cuatro bits de estado, C (acarreo), V (desbordamiento), Z (cero) y S (signo).

7-21 Supóngase que la ALU de 4 bits de la figura 7-12 está contenida en un paquete de CI. Demuestre las conexiones entre dos CI de este tipo para formar una ALU de 8 bits.

7-22 Diseñe un circuito aritmético con una variable de selección S y dos entradas de datos de n bits, A y B. El circuito genera las cuatro operaciones aritméticas siguientes junto con el acarreo de entrada C_{in}. Trace el diagrama de lógica para las dos primeras etapas.

S	$C_{in} = 0$	$C_{in} = 1$
0	$F = A + B$ (suma)	$F = A + 1$ (incremento)
1	$F = A - 1$ (decremento)	$F = A + \overline{B} + 1$ (sustracción)

7-23 Muestre el diagrama de lógica de un circuito aritmético de 4 bits que realiza las operaciones que se especifican en la tabla 7-5. Utilice cuatro multiplexores y cuatro sumadores completos.

7-24 Diseñe un circuito aritmético de cuatro bits con dos variables de selección S_1 y S_0 que genere las siguientes operaciones aritméticas. Trace el diagrama de lógica de una etapa típica.

S_1	S_0	$C_{in} = 0$	$C_{in} = 1$
0	0	$F = A + B$ (suma)	$F = A + B + 1$
0	1	$F = A$ (transferencia)	$F = A + 1$ (incremento)
1	0	$F = \overline{B}$ (complemento)	$F = \overline{B} + 1$ (negación)
1	1	$F = A + \overline{B}$	$F = A + \overline{B} + 1$ (sustracción)

7-25 Diseñe un circuito aritmético con una variable de selección S y dos entradas de datos A y B. Cuando $S = 0$, el circuito realiza la operación de adición $F = A + B$. Cuando $S = 1$, el circuito efectúa la operación de incremento $F = A + 1$.

7-26 La entrada X_i y Y_i de cada circuito sumador completo de un circuito aritmético tiene la lógica digital especificada por las funciones booleanas siguientes

$$X_i = A_i \qquad Y_i = \overline{B}_i S + B_i \overline{C}_{in}$$

donde S es una variable de selección, C_{in} es el acarreo de entrada, y A_i y B_i son los datos de entrada en la etapa i.

(a) Trace el diagrama de lógica del circuito aritmético de 4 bits.
(b) Determine las cuatro operaciones aritméticas que se realizan cuando S y C_{in} son iguales a 00, 01, 10 y 11.

7-27 Diseñe un circuito digital que realice las cuatro operaciones lógicas de OR excluyente, NOR excluyente, NOR y NAND. Utilice dos variables de selección. Muestre el diagrama de lógica de una etapa típica.

7-28 Sume otro multiplexor a la unidad de corrimiento de la figura 7-18 con dos líneas de selección separadas G_1 y G_0. Este multiplexor se utiliza para especificar la entrada en serie I_R durante la operación de corrimiento a la derecha en la forma siguiente:

G_1	G_0	Operación
0	0	Corrimiento lógico a la derecha ($I_R = 0$)
0	1	Rotación a la derecha ($I_R = F_0$)
1	0	Rotación a la derecha con acarreo ($I_R = C$, $C = F_0$)
1	1	Corrimiento aritmético a la derecha ($I_R = F_3$)

Ilustre la conexión del multiplexor a la unidad de corrimiento.

7-29 El campo H de selección del corrimiento según se define en la tabla 7-9, tiene tres variables H_2, H_1 y H_0. Las dos últimas variables de selección se utilizan para la unidad de corrimiento como se especifica en la tabla 7-7. Diseñe el circuito asociado con la variable de selección H_2 que produce una rotación.

7-30 Obtenga la tabla de funciones de una unidad de corrimiento de tambor de 4 bits que hace girar los bits a la derecha. (En la tabla 7-8 se supone una rotación a la izquierda.) Trace el circuito con cuatro multiplexores.

7-31 Obtenga la tabla de funciones de una unidad de corrimiento de tambor de 8 bits que hace girar los bits a la izquierda.

7-32 Especifique la palabra de control que se debe aplicar al procesador de la figura 7-20 para ejecutar las siguientes microoperaciones:

(a) $R2 \leftarrow R1 + 1$
(b) $R3 \leftarrow R4 + R5$
(c) $R6 \leftarrow \overline{R6}$
(d) $R7 \leftarrow R7 - 1$
(e) $R1 \leftarrow \text{shl } R1$
(f) $R2 \leftarrow \text{ror } R2$
(g) $R5 \leftarrow R3 + R1$
(h) $R6 \leftarrow R7$

7-33 Dada la siguiente palabra de control de 16 bits de la unidad procesadora de la figura 7-20, determine la microoperación que se ejecuta para cada palabra de control.

(a) 001 010 011 0101 000
(b) 110 000 001 0000 001
(c) 010 010 010 1100 000
(d) 000 001 000 1000 000
(e) 111 000 011 0000 110