
Prefacio

El *diseño digital* se ocupa del diseño de circuitos electrónicos digitales. El tema se conoce también por otros nombres, como *diseño lógico*, *circuitos conmutadores*, *lógica digital* y *sistemas digitales*. Los circuitos digitales se emplean en el diseño de sistemas, por ejemplo computadoras digitales, calculadoras electrónicas, dispositivos digitales de control, equipo de comunicación digital y muchas otras aplicaciones que requieren hardware digital electrónico. En este libro se presentan las herramientas básicas que se usan en el diseño de los circuitos digitales y se proporcionan varios métodos y procedimientos adecuados para una variedad de aplicaciones del diseño digital.

Los componentes que se utilizan para construir sistemas digitales están encapsulados en paquetes de circuitos integrados. Los circuitos de integración a pequeña escala (SSI) contienen varias compuertas o *flip-flops* en un solo paquete. Los dispositivos de integración a mediana escala (MSI) ofrecen funciones digitales específicas, y los dispositivos de integración a gran escala (LSI) proporcionan módulos completos de computadora. Es importante que el diseñador digital se familiarice con los diversos componentes digitales que se encuentran en las formas de circuitos integrados. Por esta razón, los componentes MSI y LSI de uso más frecuente se introducen en el libro junto con explicaciones de sus propiedades lógicas. El uso de circuitos integrados en el diseño de circuitos digitales se ilustra mediante ejemplos en el libro, en los problemas que se presentan al final de los capítulos y en un conjunto de 15 experimentos que se recomienda realizar en el laboratorio.

El libro consta de 11 capítulos. Los Capítulos del 1 al 5 tratan de circuitos combinacionales. Los Capítulos 6 y 7 cubren los circuitos secuenciales síncronos.

Estos siete capítulos y el Capítulo 10, los cuales abarcan los circuitos integrados digitales, se toman del libro del autor *Digital Logic and Computer Design* —Lógica digital y diseño de computadoras— (Prentice-Hall, 1979). Los Capítulos 8, 9 y 11 contienen material referente a las máquinas de estado algorítmico, circuitos secuenciales asíncronos y experimentos de laboratorio con circuitos integrados. Los once capítulos proporcionan un conjunto coherente de temas adecuados para un primer curso de diseño digital.

En el **Capítulo 1** se analizan los diversos sistemas binarios adecuados para representar información en los sistemas digitales. El **Capítulo 2** es la introducción al álgebra booleana junto con las diversas compuertas lógicas que se emplean en la construcción de circuitos digitales. En el **Capítulo 3** se cubren los métodos de mapas y tablas para simplificar los circuitos digitales y se presenta un procedimiento sistemático para el implante de las lógicas NAND y NOR. Los primeros tres capítulos dan las bases necesarias para entender el resto del libro.

En el **Capítulo 4** se compendia un procedimiento formal para el análisis y diseño de los circuitos combinacionales. El **Capítulo 5** trata con los componentes de los circuitos combinacionales MSI y LSI. Se explican las funciones de uso frecuente como sumadores, comparadores, decodificadores y multiplexores y su uso en el diseño de circuitos digitales se ilustra con ejemplos. Se introducen la memoria solo de lectura (ROM) y el arreglo lógico programable (PLA) y se demuestra su utilidad en el diseño de circuitos combinacionales complejos. En el **Capítulo 6** se compendian diversos procedimientos formales para el análisis y diseño de circuitos secuenciales síncronos con reloj. En el **Capítulo 7** se presentan varios componentes secuenciales MSI como registradores, contadores, registradores de corrimiento y la memoria de acceso aleatorio (RAM).

En el **Capítulo 8** se incluye el método de diseño digital de la máquina de estado algorítmico (ASM). El diagrama ASM es un diagrama especial de flujo adecuado para describir tanto operaciones secuenciales como las operaciones en paralelo en el hardware digital. Varios ejemplos de diseño demuestran la aplicación del diagrama ASM en el diseño del control lógico de los sistemas digitales. En el **Capítulo 9** se presentan procedimientos formales para el análisis y diseño de los circuitos secuenciales asíncronos. Se compendian métodos para mostrar cómo un circuito secuencial asíncrono puede implementarse como un circuito combinacional con retroalimentación o como un circuito con seguro SR. El **Capítulo 10** trata acerca de la electrónica de los circuitos digitales y se presentan las familias lógicas más comunes de circuito integrado digital. Se supone cierto conocimiento de electrónica básica, pero no hay un prerequisito específico para el resto del libro.

En el **Capítulo 11** se compendian 15 experimentos que pueden realizarse en el laboratorio con hardware que comercialmente está disponible y es de bajo costo. Esos experimentos usan circuitos estándar integrados del tipo TTL. La operación de los circuitos integrados se explica con referencia a diagramas en los capítulos anteriores donde en forma original se introdujeron componentes similares. Cada experimento se presenta de manera informal y no paso a paso, de modo que se espera que el estudiante produzca los detalles del diagrama del circuito y formule un procedimiento para verificar la operación del circuito en el laboratorio.

Cada capítulo incluye un conjunto de problemas y una lista de referencias. En el Apéndice aparecen las respuestas a los problemas seleccionados para ayudar al estudiante y auxiliar al lector independiente. Para el instructor se publicó un *Manual de Soluciones*, disponible en inglés.

M. MORRIS MANO

Sistemas binarios

1

1-1 COMPUTADORAS DIGITALES Y SISTEMAS DIGITALES

Las computadoras digitales han hecho posibles muchos avances científicos, industriales y comerciales que de otra manera nunca se hubieran alcanzado. El programa espacial habría sido imposible sin monitoreo continuo por computadora en tiempo real y, muchas empresas de negocios funcionan en forma eficiente sólo con la ayuda del procesamiento automático de información. Las computadoras se usan en cálculos científicos, en el procesamiento de información comercial y de negocios, control de tránsito aéreo, vía espacial, campo educativo y muchas otras áreas. La propiedad más sorprendente de una computadora digital es su generalidad. Puede seguir una secuencia de instrucciones, denominada *programa*, que opera según la información dada. El usuario puede especificar y cambiar los programas y/o la información de acuerdo con la necesidad específica. A causa de esta flexibilidad, las computadoras digitales de propósito general pueden realizar una amplia variedad de tareas de procesamiento de información.

La computadora digital de propósito general es el ejemplo mejor conocido de un sistema digital. Otros ejemplos incluyen los intercambios de canales de comunicación telefónicos, voltmetros digitales, contadores de frecuencia, máquinas calculadoras y máquinas de teletipo. Es característico de un sistema digital la manipulación de *elementos discretos* de información. Es posible que dichos elementos sean impulsos eléctricos, los dígitos decimales, las letras de un alfabeto, operaciones aritméticas, signos de puntuación, o cualquier otro conjunto de símbolos con significado. La yuxtaposición de elementos discretos de información representa una cantidad de información. Por ejemplo, las letras *s*, *o* y *l* forman la palabra *sol*. Los dígitos 237 forman un número. Por tanto, una secuencia de elementos discretos forma un lenguaje, esto es, una disciplina que lleva información. Las primeras computadoras digitales se usaron principalmente para cálculos numéricos. En este caso, los elementos discretos que se utilizan son dígitos. Para esta aplicación surgió el término de *computadora digital*. Un nombre más apropiado pero más largo para una computadora digital sería un “sistema para procesar información discreta”.

Los elementos discretos de información se representan en un sistema digital mediante cantidades físicas denominadas *señales*. Las señales eléctricas como voltajes y corrientes son las más comunes. Las señales en todos los sistemas digitales en la actualidad tienen sólo dos valores discretos y se dice que son *binarios*. El diseñador de un sistema digital está limitado al uso de señales binarias debido a la confiabilidad más baja de los circuitos electrónicos de valores múltiples. En otras palabras, puede diseñarse un circuito con diez estados, que usa un valor discreto de voltaje para cada estado, pero tendría una confiabilidad muy baja de operación. En contraste, un circuito de transistor que está, ya sea encendido o apagado, tiene dos valores de señal posibles y puede construirse para que sea en extremo confiable. Debido a esta restricción física de los componentes, y debido a que la lógica humana tiende a ser binaria, los sistemas digitales que están limitados a tomar valores discretos están restringidos aún más a tener valores binarios.

Las cantidades discretas de información emergen ya sea de la propia naturaleza del proceso o es posible cuantificarlas a propósito a partir de un proceso continuo. Por ejemplo, en forma inherente una nómina es un proceso discreto que contiene los nombres de los empleados, número de seguro social, salarios semanales, impuestos sobre el ingreso, etc. El pago de un empleado se procesa usando valores discretos de información, por ejemplo letra del alfabeto (nombre), dígitos (salario) y símbolos especiales como \$. Por otra parte, un investigador científico puede observar un proceso continuo pero registra sólo cantidades específicas en una forma tabular. Por lo tanto, el científico está cuantificando su información continua. Cada número en su tabla es un elemento discreto de información.

Muchos sistemas físicos pueden describirse en forma matemática por ecuaciones diferenciales cuyas soluciones, como una función del tiempo, dan el comportamiento matemático completo del proceso. Una *computadora analógica* realiza una *simulación* directa de un sistema físico. Cada sección de la computadora es la análoga de una porción particular del proceso bajo estudio. Las variables en la computadora analógica se representan por señales continuas, por lo común voltajes eléctricos que varían con el tiempo. Las variables de señal se consideran análogas a las del proceso y se comportan de la misma forma. En consecuencia, las mediciones de voltaje analógico pueden sustituirse por las variables del proceso. El término *señal analógica* algunas veces se sustituye por *señal continua*, debido a que la “computadora analógica” ha llegado a significar una computadora que manipula variables continuas.

Para simular un proceso físico en una computadora digital, las magnitudes deben cuantificarse. Cuando las variables del proceso se presentan por señales continuas en tiempo real, estas últimas se cuantifican con un dispositivo de conversión analógica en digital. Un sistema físico cuya conducta se describe por ecuaciones matemáticas se simula en una computadora digital mediante métodos numéricos. Cuando el problema que va a procesarse es inherentemente discreto, como en aplicaciones comerciales, la computadora digital manipula las variables en su forma natural.

En la Fig. 1-1 se muestra un diagrama de bloques de la computadora digital. La unidad de memoria almacena los programas al igual que la entrada, la salida y la información intermedia. La unidad procesadora realiza las tareas aritméticas y de otros procesamientos de información como se especifica por un programa.

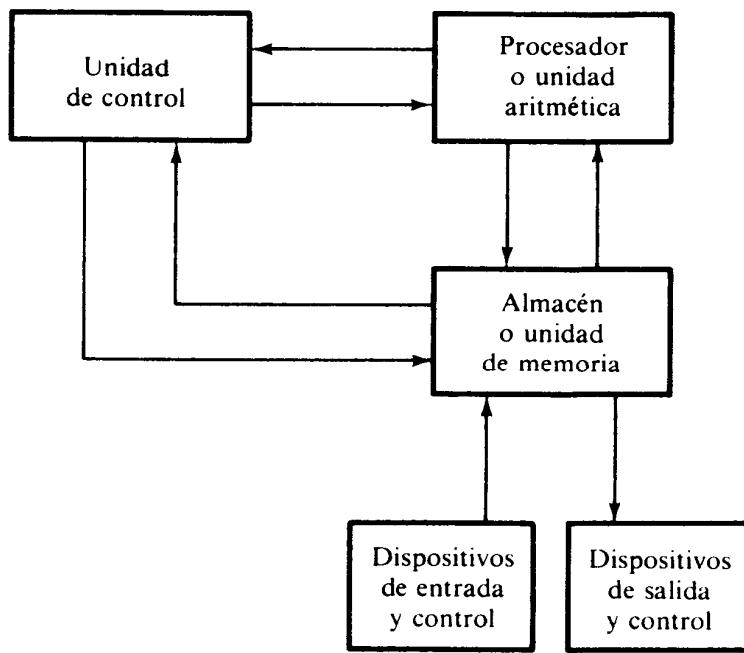


Figura 1-1 Diagrama de bloques de una computadora digital.

La unidad de control supervisa el flujo de información entre las diversas unidades. La unidad de control recupera las instrucciones, una por una, del programa que está almacenando en la memoria. Para cada instrucción, la unidad de control informa al procesador para que ejecute la operación especificada por la instrucción. Tanto el programa como la información se almacena en la memoria. La unidad de control supervisa las instrucciones del programa y el procesador manipula la información como se especifica en el programa.

El programa y la información preparada por el usuario se transfieren a la unidad de memoria mediante un dispositivo de entrada, como una lectora de tarjetas perforadas o una máquina de telescritura. Un dispositivo de salida, por ejemplo una impresora, recibe el resultado de los cálculos e imprime los resultados que se presentan al usuario. Los dispositivos de entrada y salida son sistemas digitales especiales impulsados por partes electromecánicas y controladas por circuitos digitales electrónicos.

Una calculadora electrónica es un sistema digital similar a una computadora digital, en la cual el dispositivo de entrada es un tablero de teclas y el dispositivo de salida es un exhibidor numérico. Las instrucciones entran en la calculadora mediante teclas de función, como más y menos. La información se introduce a través de las teclas numéricas. Los resultados se exhiben directamente en forma numérica. Algunas calculadoras se parecen mucho a una computadora digital si tienen capacidad de impresión y facilidades de programación. Sin embargo, una computadora digital es un dispositivo más poderoso que una calculadora. Una computadora digital debe conectarse con muchos otros dispositivos de entrada y salida, puede realizar no sólo cálculos aritméticos sino también operaciones lógicas y puede programarse para tomar decisiones basadas en condiciones internas y externas.

Una computadora digital es una interconexión de módulos digitales. Para entender la operación de cada módulo digital, es necesario tener un conocimiento

básico de los sistemas digitales y su comportamiento general. En los primeros cuatro capítulos de este libro se introducen las herramientas básicas del diseño digital, por ejemplo los números binarios y códigos, álgebra booleana y los bloques básicos con los cuales se construyen los circuitos electrónicos digitales. En los Capítulos 5 y 7 se presentan los componentes básicos que se encuentran en la unidad procesadora de una computadora digital. Las características operacionales de la unidad de memoria se explican al final del Capítulo 7. El diseño de la unidad de control se expone en el Capítulo 8 utilizando los principios básicos de los circuitos secuenciales del Capítulo 6.

Un procesador, cuando se combina con la unidad de control, forma un componente referido, una *unidad central de proceso* o CPU. Una CPU encerrada en un pequeño paquete de circuito integrado se denomina *microprocesador*. Es posible que la unidad de memoria, lo mismo que la parte que controla la interfase entre el microprocesador y los dispositivos de entrada y salida, esté encapsulada dentro del paquete del microprocesador o puede estar disponible en otros paquetes pequeños de circuitos integrados. Una CPU combinada con memoria y control de interfase para formar una computadora de tamaño pequeño se conoce como *microcomputadora*. La disponibilidad de los componentes de microcomputadoras ha revolucionado la tecnología de diseño de sistemas digitales, dando al diseñador la libertad para crear estructuras que antes eran antieconómicas. Los diversos componentes de un sistema microcomputador se construyen internamente con circuitos digitales.

Ya se ha mencionado que una computadora digital manipula elementos discretos de información y que esos elementos se representan en forma binaria. Los operandos que se utilizan para el cálculo pueden expresarse en el sistema de números binarios. Otros elementos discretos, incluyendo los dígitos decimales, se representan en códigos binarios. El procesamiento de datos se lleva a cabo mediante elementos lógicos binarios que usan señales binarias. Las cantidades se almacenan en elementos de almacenamiento binario. El propósito de este capítulo es introducir los diversos conceptos binarios como un marco de referencia para un estudio detallado en los capítulos subsecuentes.

1-2 NUMEROS BINARIOS

Un número decimal como 7392 representa una cantidad igual a 7 millares, más 3 centenas, más 9 decenas, más 2 unidades. Los millares, centenas, etc., son potencias de 10 implicadas por la posición de los coeficientes. Para ser más exactos, 7392 debe escribirse como:

$$7 \times 10^3 + 3 \times 10^2 + 9 \times 10^1 + 2 \times 10^0$$

Sin embargo, la convención es escribir sólo los coeficientes y de sus posiciones se deducen las potencias necesarias de 10. En general, un número con un punto decimal se representa por una serie de coeficientes como sigue:

$$a_5a_4a_3a_2a_1a_0 \cdot a_{-1}a_{-2}a_{-3}$$

Los coeficientes a_j son uno de los diez dígitos (0, 1, 2,...,9), y el valor del subíndice j da el valor del lugar y, por tanto, la potencia de 10 por la cual debe multiplicarse el coeficiente.

$$10^5a_5 + 10^4a_4 + 10^3a_3 + 10^2a_2 + 10^1a_1 + 10^0a_0 + 10^{-1}a_{-1} \\ + 10^{-2}a_{-2} + 10^{-3}a_{-3}$$

El sistema de números decimales se dice que es de *base*, o *raíz*, 10 debido a que usa diez dígitos y los coeficientes se multiplican por potencias de 10. El sistema *binario* es un sistema diferente de números. Los coeficientes de los números del sistema binario tienen dos valores posibles: 0 y 1. Cada coeficiente a_j se multiplica por 2^j . Por ejemplo, el equivalente decimal del número binario 11010.11 es 26.75, como se muestra por la multiplicación de los coeficientes por potencias de 2:

$$1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} \\ + 1 \times 2^{-2} = 26.75$$

En general, un número expresado en un sistema base r tiene coeficientes multiplicados por las potencias de r :

$$a_n \cdot r^n + a_{n-1} \cdot r^{n-1} + \cdots + a_2 \cdot r^2 + a_1 \cdot r + a_0 \\ + a_{-1} \cdot r^{-1} + a_{-2} \cdot r^{-2} + \cdots + a_{-m} \cdot r^{-m}$$

Los coeficientes a_j varían en valor desde 0 a $r - 1$. Para distinguir entre números de bases diferentes, se encierran entre paréntesis los coeficientes y se escribe un subíndice igual a la base usada (excepto algunas veces para números decimales, donde el contenido indica obviamente que es decimal). Un ejemplo de un número con base 5 es

$$(4021.2)_5 = 4 \times 5^3 + 0 \times 5^2 + 2 \times 5^1 + 1 \times 5^0 + 2 \times 5^{-1} = (511.4)_{10}$$

Obsérvese que los valores de los coeficientes con base 5 pueden ser sólo 0, 1, 3 y 4.

Es costumbre tomar los r dígitos necesarios para los coeficientes del sistema decimal cuando la base del número es menor que 10. Se usan las letras del alfabeto para completar los diez dígitos decimales cuando la base del número es mayor que 10. Por ejemplo, en el sistema *hexadecimal* (base 16), los primeros diez dígitos se toman del sistema decimal. Las letras A, B, C, D, E y F se utilizan para los dígitos 10, 11, 12, 13, 14 y 15, respectivamente. Un ejemplo de un número hexadecimal es:

$$(B65F)_{16} = 11 \times 16^3 + 6 \times 16^2 + 5 \times 16 + 15 = (46687)_{10}$$

Los primeros 16 números en los sistemas decimal, binario, octal y hexadecimal se listan en la Tabla 1-1.

Las operaciones aritméticas con números en la base r siguen las mismas reglas que en el caso de los números decimales. Cuando se usa otra base diferente de la base familiar 10, debe tenerse cuidado de utilizar sólo los r dígitos permitidos. A continuación se muestran ejemplos de la suma, resta y multiplicación de dos números binarios:

La adición de dos números binarios se calcula según las mismas reglas que en los decimales, excepto que los dígitos de la adición en cualquier posición significativa

sumando:	101101	minuendo:	101101	multiplicando:	1011
adendo:	+ 100111	sustraendo:	- 100111	multiplicador:	× 101
suma:	1010100	diferencia:	000110		
					1011
					0000
					1011
				producto:	110111

pueden ser sólo 0 o 1. Cualquier “acarreo” que se obtenga en una posición significativa dada se usa por el par de dígitos en una posición significativa más alta. La sustracción es ligeramente más complicada. Las reglas siguen siendo las mismas que en el sistema decimal, excepto que “el préstamo (acarreo)” en una posición significativa dada añade 2 a un dígito minuendo. (Un “préstamo” en el sistema decimal añade 10 a un dígito minuendo.) La multiplicación es muy simple. Los dígitos multiplicadores siempre son 1 o 0. Así que los productos parciales son iguales ya sea al multiplicando o bien a 0.

1-3 CONVERSIONES DE LA BASE DE NUMEROS

Un número binario puede convertirse en decimal formando la suma de las potencias de dos de los coeficientes cuyo valor es 1. Por ejemplo:

$$(1010.011)_2 = 2^3 + 2^1 + 2^{-2} + 2^{-3} = (10.375)_{10}$$

TABLA 1-1 Números con bases diferentes

Decimal (base 10)	Binario (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

El número binario tiene cuatro números 1 y el equivalente decimal se encuentra por la adición de cuatro potencias de 2. En forma similar, un número que se expresa en la base r puede convertirse en su equivalente decimal multiplicando cada coeficiente por la potencia correspondiente de r y sumando. El siguiente es un ejemplo de la conversión de octal en decimal:

$$(630.4)_8 = 6 \times 8^2 + 3 \times 8 + 4 \times 8^{-1} = (408.5)_{10}$$

La conversión de decimal en binario o cualquier otro sistema con base r es más conveniente si el número se separa en una *parte entera* y en una *parte fraccional*, y la conversión de cada parte se hace por separado. La conversión de un entero de decimal en binario se explica de manera mas adecuada con un ejemplo.

EJEMPLO 1-1: Convierta el decimal 41 en binario. Primero, 41 se divide entre 2 para dar un cociente entero de 20 y un residuo de 1/2. El cociente se divide otra vez entre 2 para dar un nuevo cociente y un residuo. Este proceso se continúa hasta que el entero cociente llega a ser 0. Los *coeficientes* del número binario deseado se obtienen por los *residuos* como sigue:

<u>cociente entero</u>		<u>residuo</u>	<u>coeficiente</u>
$\frac{41}{2} = 20$	+	$\frac{1}{2}$	$a_0 = 1$
$\frac{20}{2} = 10$	+	0	$a_1 = 0$
$\frac{10}{2} = 5$	+	0	$a_2 = 0$
$\frac{5}{2} = 2$	+	$\frac{1}{2}$	$a_3 = 1$
$\frac{2}{2} = 1$	+	0	$a_4 = 0$
$\frac{1}{2} = 0$	+	$\frac{1}{2}$	$a_5 = 1$

respuesta: $(41)_{10} = (a_5a_4a_3a_2a_1a_0)_2 = (101001)_2$

El proceso aritmético puede manipularse en forma más conveniente como sigue:

entero	residuo
41	
20	1
10	0
5	0
2	1
1	0
0	1

101001 = respuesta

La conversión de enteros decimales en cualquier sistema con base r es similar al ejemplo anterior, excepto que la división se hace entre r en lugar de 2.

EJEMPLO 1-2: Convierta el decimal 153 en octal. La base requerida r es 8. Primero, 153 se divide entre 8 para dar un cociente entero de 19 y un residuo de 1. Entonces 19 se divide entre 8 para dar un cociente entero de 2 y un residuo de 3. Por último, 2 se divide entre 8 para dar un cociente de 0 y un residuo de 2. Este proceso puede manipularse en forma conveniente como sigue:

$$\begin{array}{r|l} 153 & \\ 19 & \quad 1 \\ 2 & \quad 3 \\ 0 & \quad 2 \end{array} = (231)_8$$

La conversión de una fracción decimal en binario se lleva a cabo por un método similar al que se utiliza para los enteros. Sin embargo, se usa la multiplicación en lugar de la división, y los enteros se acumulan en lugar de residuos. De nuevo, el método se explica de manera más adecuada con un ejemplo.

EJEMPLO 1-3: Convierta $(0.6875)_{10}$ en binario. Primero 0.6875 se multiplica por 2 para dar un entero y una fracción. La nueva fracción se multiplica por 2 para dar un entero y una nueva fracción. Este proceso se continúa hasta que la fracción llega a ser 0 o hasta que el número de dígitos tiene suficiente exactitud. Los coeficientes del número binario se obtienen mediante los enteros como sigue:

	entero	fracción	coeficiente
$0.6875 \times 2 =$	1	+	$a_{-1} = 1$
$0.3750 \times 2 =$	0	+	$a_{-2} = 0$
$0.7500 \times 2 =$	1	+	$a_{-3} = 1$
$0.5000 \times 2 =$	1	+	$a_{-4} = 1$

$$\text{respuesta: } (0.6875)_{10} = (0.a_{-1}a_{-2}a_{-3}a_{-4})_2 = (0.1011)_2$$

Para convertir una fracción decimal en un número expresado en base r , se usa un procedimiento similar. La multiplicación se hace por r en lugar de 2, y los coeficientes se encuentran por los enteros que pueden variar el valor desde 0 a $r - 1$ en lugar de 0 y 1.

EJEMPLO 1-4: Convierta $(0.513)_{10}$ en octal.

$$0.513 \times 8 = 4.104$$

$$0.104 \times 8 = 0.832$$

$$0.832 \times 8 = 6.656$$

$$0.656 \times 8 = 5.248$$

$$0.248 \times 8 = 1.984$$

$$0.984 \times 8 = 7.872$$

La respuesta, a siete cifras significativas, se obtiene de la parte entera de los productos:

$$(0.513)_{10} = (0.406517 \dots)_8$$

La conversión de números decimales tanto con las partes enteras como fraccionarias se hace por la conversión de la parte entera y fraccionaria por separado y, entonces, combinando las dos respuestas. Por el uso de los resultados de los Ejemplos 1-1 y 1-3; se obtiene:

$$(41.6875)_{10} = (101001.1011)_2$$

A partir de los Ejemplos 1-2 y 1-4, se obtiene:

$$(153.513)_{10} = (231.406517)_8$$

1-4 NUMEROS OCTALES Y HEXADECIMALES

La conversión de binario, octal y hexadecimal —y a la inversa— juega una parte importante en las computadoras digitales. Ya que $2^3 = 8$ y $2^4 = 16$, cada dígito octal corresponde a tres dígitos binarios y cada dígito hexadecimal corresponde a cuatro dígitos binarios. La conversión de binario en octal se lleva a cabo fácilmente por la partición del número binario en grupos de tres dígitos cada uno, principiando desde el punto binario y procediendo a la izquierda y a la derecha. Entonces, el dígito octal correspondiente se asigna a cada grupo. El siguiente ejemplo ilustra el procedimiento:

$$(\begin{array}{ccccc} \boxed{10} & \boxed{110} & \boxed{001} & \boxed{101} & \boxed{011} \\ 2 & 6 & 1 & 5 & 3 \end{array} \quad \begin{array}{ccccc} \boxed{111} & \boxed{100} & \boxed{000} & \boxed{110} \\ 7 & 4 & 0 & 6 \end{array})_2 = (26153.7406)_8$$

La conversión de binario en hexadecimal es similar, excepto que el número binario se divide en grupos de cuatro dígitos.

$$(\begin{array}{ccccc} \boxed{10} & \boxed{1100} & \boxed{0110} & \boxed{1011} \\ 2 & C & 6 & B \end{array} \quad \begin{array}{cc} \boxed{1111} & \boxed{0010} \\ F & 2 \end{array})_2 = (2C6B.F2)_{16}$$

El dígito correspondiente hexadecimal (u octal) para cada grupo de dígitos binarios se recuerda con facilidad después de estudiar los valores listados en la Tabla 1-1.

La conversión de octal o hexadecimal en binario se realiza por un procedimiento inverso al anterior. Cada dígito octal se convierte en su equivalente binario de tres dígitos. De manera semejante, cada dígito hexadecimal se convierte en su equivalente binario de cuatro dígitos. Esto se ilustra en los siguientes ejemplos:

$$(673.124)_8 = (\begin{array}{ccc} \underline{110} & \underline{111} & \underline{011} \\ 6 & 7 & 3 \end{array} \cdot \begin{array}{ccc} \underline{001} & \underline{010} & \underline{100} \\ 1 & 2 & 4 \end{array})_2$$

$$(306. D)_{16} = (\begin{array}{ccc} \underline{0011} & \underline{0000} & \underline{0110} \\ 3 & 0 & 6 \end{array} \cdot \begin{array}{c} \underline{1101} \\ D \end{array})_2$$

Los números binarios son difíciles de trabajar ya que requieren tres o cuatro veces más dígitos que su equivalente decimal. Por ejemplo, el número binario 111111111111 es equivalente al decimal 4 095. No obstante, las computadoras digitales utilizan números binarios y algunas veces es necesario que el operador humano o usuario se comunique en forma directa con la máquina mediante números binarios. Un esquema que retiene el sistema binario en la computadora, pero que reduce el número de dígitos que el humano debe considerar, emplea la relación entre el sistema de números binarios y el sistema octal o hexadecimal. Por este método, el humano piensa en términos de números octales o hexadecimales y lleva a cabo la conversión requerida por inspección, cuando es necesaria la comunicación directa con la máquina. Por tanto, el número binario 111111111111 contiene doce dígitos y se expresa en octal, 7777 (cuatro dígitos) o en hexadecimal como FFF (tres dígitos). Durante la comunicación entre personas (acerca de los números binarios en la computadora), la representación octal o hexadecimal es más deseable debido a que puede expresarse en forma compacta con un tercio o un cuarto del número de dígitos requeridos por el número binario equivalente. Cuando el humano se comunica con la máquina (a través de interruptores en la consola con luces indicadoras, o mediante programas escritos en *lenguaje de máquina*), la conversión de octal o hexadecimal en binario y viceversa se hace por inspección por el usuario humano.

1-5 COMPLEMENTOS

Los complementos se usan en las computadoras digitales para simplificar la operación de sustracción y para manipulaciones lógicas. Hay dos tipos de complementos para cada sistema base r : (1) el complemento de r y (2) el complemento de $(r - 1)$. Cuando el valor de la base se sustituye, los dos tipos reciben los nombres de 2 y de 1 para complementos de números binarios o, de 10 y de 9 para complementos de números decimales.

El complemento de r

Dado un número positivo N en base r con una parte entera de n dígitos, el complemento de r de N se define como $r^n - N$ para $N \neq 0$ y 0 para $N = 0$. El siguiente ejemplo numérico ayudará a aclarar la definición:

El complemento de 10 $(52\ 520)_{10}$ es $10^5 - 52\ 520 = 47\ 480$

El número de dígitos en el número es $n = 5$.

El complemento de 10 de $(0.3267)_{10}$ es $1 - 0.3267 = 0.6733$.

No hay parte entera, de modo que $10^n = 10^0 = 1$.

El complemento de 10 de $(25.639)_{10}$ es $10^2 - 25.639 = 74.361$.

El complemento de 2 de $(101100)_2$ es $(2^6)_{10} - (101100)_2 = (1000000 - 101100)_2 = 010100$.

El complemento de 2 de $(0.0110)_2$ es $(1 - 0.0110)_2 = 0.1010$.

Por la definición y los ejemplos, es claro que el complemento de 10 de un número decimal puede formarse dejando todos los ceros significativos sin cambio, se resta el primer dígito de cero menos significativo de 10 y, entonces, se restan todos los otros dígitos más significativos de 9. El complemento de 2 puede formarse dejando todos los ceros menos significativos y el primer dígito no cero sin cambio y, entonces, se reemplazan los 1 por 0 y los 0 por 1 en todos los otros dígitos más significativos. Un tercer método más simple para obtener el complemento de r se da después de la definición del complemento $(r - 1)$.

El complemento de r de un número existe para cualquier base r (r mayor que pero no igual a 1) y puede obtenerse por la definición que se presentó antes. Los ejemplos listados aquí usan números con $r = 10$ (decimal) y $r = 2$ (binario), debido a que estas son las dos bases de mayor interés para nosotros. El nombre del complemento se relaciona con la base del número que se usa. Por ejemplo, el complemento de $(r - 1)$ de un número en la base 11 se denomina complemento de 10, ya que $r - 1 = 10$ para $r = 11$.

El complemento de $(r - 1)$

Dado un número positivo N en base r con una parte entera de n dígitos y una parte fraccionaria de m dígitos, el complemento de $(r - 1)$ de N se define como $r^n - r^{-m} - N$. A continuación se presentan algunos ejemplos numéricos:

El complemento de 9 de $(52\ 520)_{10}$ es $(10^5 - 1 - 52\ 520) = 99\ 999 - 52\ 520 = 47\ 479$.

No hay parte fraccionaria, de modo que $10^{-m} = 10^0 = 1$.

El complemento de 9 de $(0.3267)_{10}$ es $(1 - 10^{-4} - 0.3267) = 0.9999 - 0.3267 = 0.6732$.

No hay parte entera, de modo que $10^n = 10^0 = 1$.

El complemento de 9 de $(25.639)_{10}$ es $(10^2 - 10^{-3} - 25.639) = 99.999 - 25.639 = 74.360$.

El complemento de 1 de $(101100)^2$ es $(2^6 - 1) - (101100) = (111111 - 101100)_2 = 010011$.

El complemento de 1 de $(0.0110)_2$ es $(1 - 2^{-4})_{10} - (0.0110)_2 = (0.1111 - 0.0110)_2 = 0.1001.$

Por los ejemplos, puede observarse que el complemento de 9 de un número decimal se forma simplemente al restar cada dígito de 9. El complemento de 1 de un número binario es aun más simple de formar: los 1 se cambian en 0 y los 0 en 1. Ya que el complemento de $(r - 1)$ se obtiene de manera muy fácil, algunas veces es conveniente usarlo cuando se desea el complemento de r . Por las definiciones y mediante una comparación de los resultados que se obtuvieron en los ejemplos, se concluye que el complemento de r puede obtenerse del complemento $(r - 1)$ después de la adición de r^{-m} al dígito menos significativo. Por ejemplo, el complemento de 2 de 10110100 se obtiene del complemento de 1 de 01001011 por la adición de 1 para obtener 01001100.

Vale la pena mencionar que el complemento del complemento restablece el número a su valor original. El complemento r de N es $r^n - N$ y el complemento de $(r^n - N)$ es $r^n - (r^n - N) = N$; y en forma similar para el complemento de 1.

Sustracción con complementos de r

El método directo de sustracción que se enseña en las escuelas elementales usa el concepto de préstamo. En este método, se presta un 1 de una posición significativa más alta cuando el dígito minuendo es menor que el dígito sustraendo correspondiente. Esto parece ser más fácil cuando las personas realizan la resta con lápiz y papel. Cuando la resta se implanta mediante componentes digitales, se encuentra que este método es menos eficiente que el método que utiliza complementos y suma como se establece más adelante.

La sustracción de dos números positivos ($M - N$), ambos en base r , puede hacerse como sigue:

1. Agréguese el minuendo M al complemento de r del sustraendo N .
 2. Inspecciónese el resultado que se obtuvo en el paso 1 para un “acarreo final”:
 - (a) Si ocurre un “acarreo final”, descártense.
 - (b) Si no ocurre un “acarreo final”, tómese el complemento r del número que se obtuvo en el paso 1 y colóquese un signo negativo enfrente.

Los siguientes ejemplos ilustran el procedimiento:

EJEMPLO 1-5: Usando el complemento de 10, reste 72 532 – 3 250.

$$\begin{array}{r}
 M = 72532 \\
 N = 03250 \\
 \hline
 10 \text{ complemento de } N = 96\,750 \\
 \hline
 \end{array}
 \quad
 \begin{array}{r}
 72532 \\
 + \\
 96750 \\
 \hline
 69282 \\
 \hline
 \end{array}$$

EJEMPLO 1-6: Reste: $(3\ 250 - 72\ 532)_{10}$.

$$\begin{array}{r}
 M = 03250 & 03250 \\
 N = 72532 & \\
 \\
 + & \\
 \hline
 10 \text{ complemento de } N = 27468 & 27468 \\
 & \text{no acarreo} \\
 & \hline
 & 30718
 \end{array}$$

respuesta: $-69\,282 = -$ (10 complemento de 30 618).

EJEMPLO 1-7: Utilice el complemento de 2 para realizar $M - N$ con los números binarios dados.

respuesta: 10 000

respuesta: $-10\ 000 = -$ (2 complemento de 1110000)

La prueba del procedimiento es: La adición de M al complemento r de N da $(M + r^n - N)$. Para números que tienen una parte entera de n dígitos, r^n es igual a 1 en la posición $(n + 1)$ ava (lo que se ha denominado “acarreo final”). Ya que se supone que tanto M como N son positivos, entonces:

- $$(a) \quad (M + r^n - N) > r^n \quad \text{si } M > N, \quad \text{o} \\ (b) \quad (M + r^n - N) < r^n \quad \text{si } M < N$$

En el caso (a) la respuesta es positiva e igual a $M - N$, la cual se obtiene en forma directa descartando y se lleva cuenta final r'' . En el caso (b), la respuesta es negativa e igual a $-(N - M)$. Este caso se detecta con la ausencia de un acarreo final. La respuesta se obtiene tomando un segundo complemento y añadiendo un signo negativo:

$$-\left[r^n - (M + r^n - N) \right] = -(N - M).$$

Sustracción con el complemento de $(r - 1)$

El procedimiento para la resta con el complemento de $(r - 1)$ es exactamente el mismo que se usó con el complemento de r , excepto por una variación, llamada “acarreo final desplazado”, como se muestra adelante. La resta de $M - N$, con ambos números positivos en la base r , puede calcularse en la siguiente forma:

1. Agréguese el minuendo M al complemento de $(r - 1)$ del sustraendo N .
 2. Inspecciónese el resultado que se obtuvo en el paso 1 para un acarreo final.
 - (a) Si ocurre un acarreo final, agréguese 1 al dígito menos significativo (acarreo final desplazado).
 - (b) Si no ocurre un acarreo final, tómese el complemento de $(r - 1)$ del número obtenido en el paso 1 y colóquese al frente un signo negativo.

La prueba de este procedimiento es muy semejante a la dada para el caso de complemento de r y se deja como ejercicio. Los ejemplos siguientes ilustran el procedimiento.

EJEMPLO 1-8: Repita los Ejemplos 1-5 y 1-6 usando complementos de 9.

(a) $M = 72532$ $N = 03250$ $9 \text{ complemento de } N = 96749$

	$+ 96749$	
	 + 1 69281 — 69282	
	acarreo final y se pasa a la derecha	

respuesta: 69282

$$\begin{array}{r}
 \text{(b)} \qquad \qquad M = 03250 \qquad \qquad 03250 \\
 \qquad \qquad N = 72532 \\
 9 \text{ complemento de } N = 27467 \qquad \qquad + \qquad 27467 \\
 \hline
 \qquad \qquad \qquad \text{no acarreo} \qquad \qquad \sqrt{30717}
 \end{array}$$

respuesta : $-69282 = - (9 \text{ complemento de } 30717)$

EJEMPLO 1-9: Repita el Ejemplo 1-7 utilizando complementos de 1.

(a) $M = 1010100$ 1010100
 $N = 1000100$ $+ 0111011$
 1 complemento de $N = 0111011$ $\overline{0001111}$
 acarreo final
 y se pasa a la derecha $\begin{array}{r} 1 \\ \hline 0001111 \\ + \\ \hline 0010000 \end{array}$

respuesta : 10000

(b) $M = 1000100$ 1000100
 $N = 1010100$ $+ 0101011$
 1 complemento de $N = 0101011$ $\overline{1101111}$
 no acarreo

respuesta: $-10000 = -(1 \text{ complemento de } 1101111)$

Comparación entre los complementos de 1 y 2

Una comparación entre los complementos de 1 y 2 revela las ventajas y desventajas de cada uno. El complemento de 1 tiene la ventaja de ser más fácil de implantar por componentes digitales, ya que lo único que debe hacerse es cambiar los 0 en números 1 y los 1 en números 0. El implante del complemento de 2 puede obtenerse en dos formas: (1) por la adición de 1 al dígito menos significativo del complemento de 1 y (2), dejando todos los 0 precedentes en las posiciones menos significativas y el primer 1 sin cambio, y sólo entonces cambiar todos los 1 en 0 y todos los 0 en 1. Durante la resta de dos números por complementos, el complemento de 2 es ventajoso ya que sólo se requiere una operación aritmética de adición. El complemento de 1 necesita dos adiciones aritméticas cuando ocurre un acarreo final desplazado. El complemento de 1 tiene la desventaja adicional de poseer dos ceros aritméticos: uno con todos los 0 y uno con todos los 1. Para ilustrar este hecho considérese la resta de dos números binarios iguales $1100 - 1100 = 0$.

Utilizando el complemento de 1:

$$\begin{array}{r} 1100 \\ + 0011 \\ \hline 1111 \end{array}$$

Se complementa de nuevo para obtener -0000 .

Usando el complemento de 2:

$$\begin{array}{r}
 1100 \\
 + 0100 \\
 \hline
 + 0000
 \end{array}$$

Mientras que el complemento de 2 tiene sólo un cero aritmético, el complemento de 1 cero puede ser positivo o negativo, lo cual puede complicar las cosas.

Los complementos, muy útiles para las manipulaciones aritméticas en computadoras digitales, se exponen con más detalle en los Capítulos 8 y 9. Sin embargo, el complemento de 1 también es útil en manipulaciones lógicas (como se mostrará después), ya que el cambio de 1 a 0 y viceversa es equivalente a una operación lógica de inversión. El complemento de 2 se utiliza sólo en conjunción con aplicaciones aritméticas. En consecuencia, es conveniente aceptar la siguiente convención: Cuando la palabra *complemento*, sin mención del tipo, se usa en conjunción con una aplicación no aritmética, se supone que el tipo es complemento de 1.

1-6 CODIGOS BINARIOS

Los sistemas electrónicos digitales utilizan señales que tienen dos valores distintos y elementos de circuito que tienen dos estados estables. Hay una analogía directa entre las señales binarias, los elementos de circuito binario y dígito binario. Un número binario de n dígitos, por ejemplo, puede representarse por n elementos de números binarios, cada uno con una señal de salida equivalente a 0 o a 1. Los sistemas digitales representan y manipulan no sólo números binarios, sino también otros muchos elementos discretos de información. Cualquier elemento discreto de información distinto entre un grupo de cantidades puede representarse por un código binario. Por ejemplo, el *rojo* es un color definido del espectro. La letra *A* es una letra distinta del alfabeto.

Un *bit*, por definición, es un dígito binario. Cuando se usa junto con un código binario, es mejor considerarlo como si denotara una cantidad binaria igual a 0 o 1. Para representar un grupo de 2^n elementos distintos en un código binario, se requiere un mínimo de n bits. Esto se debe a que es posible ordenar n bits en 2^n formas distintas. Por ejemplo, un grupo de cuatro cantidades diferentes puede representarse mediante un código de 2 bits, con cada cantidad asignada a una de las siguientes combinaciones de bit: 00, 01, 10, 11. Un grupo de ocho elementos requiere un código de tres bits, con cada elemento asignado a uno y sólo uno de los siguientes: 000, 001, 010, 011, 100, 101, 110, 111. En los ejemplos se muestra que las distintas combinaciones de bits para un código de n bits pueden encontrarse al contar en binario desde 0 a $(2^n - 1)$. Algunas combinaciones de bit quedan sin asignarse cuando el número de elementos del grupo que va a codificarse no es un múltiplo de la potencia de 2. Los diez dígitos decimales 0, 1, 2, ..., 9 son un ejemplo de tal clase de grupo. Un código binario que se distingue entre diez elementos debe contener cuando menos cuatro bits; tres bits pueden distinguir un máximo de ocho elementos. Cuatro bits pueden formar 16 combinaciones distintas, pero como sólo se codifican diez dígitos, las seis combinaciones restantes quedan sin asignar y no se utilizan.

Aunque el número *mínimo* de bits necesario para codificar 2^n cantidades distintas es n , no hay número *máximo* de bits que puedan usarse para un código binario. Por ejemplo, los diez dígitos decimales pueden codificarse con diez bits y, cada dígito decimal se asigna a una combinación de bits de nueve números 0 y un 1. En este código binario particular, al dígito 6 se le asigna la combinación de bits 0001000000.

Códigos decimales

Los códigos binarios para dígitos decimales requieren un mínimo de cuatro bits. Se obtienen numerosos códigos diferentes al ordenar cuatro o más bits en diez distintas combinaciones. Unas cuantas posibilidades se muestran en la Tabla 1-2.

El BCD (de las iniciales en inglés para decimal codificado binario) es una asignación directa del equivalente binario. Es posible asignar pesos a los bits binarios de acuerdo con sus posiciones. Los pesos en el código BCD son 8, 4, 2, 1. Por ejemplo, la asignación de bits 0110 puede interpretarse por los pesos que representan el dígito decimal 6, porque $0 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1 = 6$. También es factible asignar pesos negativos a un código decimal, como se muestra por el código 8, 4, -2, -1. En este caso la combinación de bits 0110 se interpreta como el dígito decimal 2, cuando se obtiene por $0 \times 8 + 1 \times 4 + 1 \times (-2) + 0 \times (-1) = 2$. Los otros dos códigos pesados que se muestran en la tabla son 2421 y el 5043210. Un código decimal que se ha usado en algunas computadoras antiguas es el código exceso-3. Este es un código sin peso; sus asignaciones se obtienen por el valor correspondiente del BCD después de la adición de 3.

Los números se representan en las computadoras digitales ya sea en binario o en decimal a través de un código binario. Cuando se especifican los datos, al usuario le gusta dar información en forma decimal. Los números decimales de entrada se almacenan internamente en la computadora mediante un código decimal. Cada dígito decimal requiere cuando menos el almacenamiento de cuatro elementos binarios. Los números decimales se convierten en binarios cuando las operaciones aritméticas se hacen de manera interna con números representados en binario. También es posible

TABLA 1-2 Códigos binarios para los dígitos decimales

Dígito decimal	(BCD) 8421	Exceso-3	84-2-1	2421	(Biquinario) 5043210
0	0000	0011	0000	0000	0100001
1	0001	0100	0111	0001	0100010
2	0010	0101	0110	0010	0100100
3	0011	0110	0101	0011	0101000
4	0100	0111	0100	0100	0110000
5	0101	1000	1011	1011	1000001
6	0110	1001	1010	1100	1000010
7	0111	1010	1001	1101	1000100
8	1000	1011	1000	1110	1001000
9	1001	1100	1111	1111	1010000

llevar a cabo en forma directa las operaciones aritméticas en decimal, con todos los números que quedan en una forma codificada a través del proceso. Por ejemplo, el número decimal 395, cuando se convierte en binario es igual a 110001011 y consta de nueve dígitos binarios. El mismo número, cuando se representa internamente en el código BCD, ocupa cuatro bits para cada dígito decimal, para un total de 12 bits: 001110010101. Los primeros cuatro bits representan un 3, los siguientes cuatro un 9 y los últimos cuatro un 5.

Es muy importante entender la diferencia entre *conversión* de un número decimal en binario y la *codificación* binaria de un número decimal. En cada caso el resultado final es una serie de bits. Los bits que se obtienen por conversión son dígitos binarios. Los bits que se obtienen por la codificación son combinaciones de 1 y 0 arreglados de acuerdo con las reglas del código que se emplee. Por consiguiente, es de extrema importancia darse cuenta de que una serie de números 1 y 0 en un sistema digital algunas veces representa un número binario y en otras ocasiones representa alguna otra cantidad discreta de información como se especifique por un código binario dado. Por ejemplo, el código BCD se ha escogido para que sea tanto un código como una conversión binaria discreta, en tanto los números decimales sean enteros desde 0 a 9. Para números mayores de 9, la conversión y la codificación son por completo diferentes. Este concepto es tan importante que vale la pena repetirlo con otro ejemplo. La conversión binaria del decimal 13 es 1101; la verificación del decimal 13 con el código BCD es 00010011.

De los cinco códigos binarios que se listan en la Tabla 1-2, el BCD parece ser el de uso más natural y por supuesto es el que se encuentra por lo común. Los otros códigos de cuatro bits listados tienen una característica en común que no se encuentra en el BCD. El exceso-3 el 2, 4, 2, 1, y el 8, 4, -2, -1, son códigos autocomplementarios, es decir, el complemento a 9 del número decimal se obtiene fácilmente cambiando los números 1 a 0 y los 0 a 1. Por ejemplo, el decimal 395 se representa en el código 2, 4, 2, 1, por 00111111011. El complemento a 9 de 604, se representa por 110000000100, lo cual se obtiene por el reemplazo de los 1 por 0 y los 0 por 1. Esta propiedad es útil cuando se hacen operaciones aritméticas en forma interna con números decimales (en un código binario) y la resta se calcula mediante el complemento a 9.

El código biquinario que se muestra en la Tabla 1-2 es un ejemplo de un código de siete bits con propiedades de detección de error. Cada dígito decimal consta de cinco números 0 y dos 1 colocados en las columnas pesadas correspondientes. Es posible comprender la propiedad de detección de error de este código al percibirse de que el sistema digital representa el binario 1 por una señal distinta y el binario 0 por una segunda diferente. Durante la transmisión de señales de una localidad a otra es factible que ocurra un error. Uno o más bits pueden cambiar de valor. Un circuito en el lado receptor puede detectar la presencia de más (o menos) dos números 1 y, si la combinación recibida de bits no coincide con esta combinación permitida, se detecta error.

Códigos de detección de error

La información binaria, se trate de señales de pulso modulado o bien, entrada o salida digital a computadora, puede transmitirse a través de alguna forma de medio de

comunicación, como alambres u ondas de radio. Cualquier ruido externo que se introduce en un medio de comunicación física cambia los valores de bits de 0 a 1 y viceversa. Puede utilizarse un código de detección de errores para detectar los errores cuando se realiza la transmisión. No es posible corregir el error detectado pero se indica su presencia. El procedimiento usual es observar la frecuencia de errores. Si el error ocurre sólo de manera esporádica, al azar y sin efecto pronunciado en la información global transmitida, entonces no se hace nada o el mensaje erróneo en particular se transmite otra vez. Si ocurren errores con frecuencia y se distorsiona el significado de la información recibida, se verifica el sistema por mal funcionamiento.

Un bit de *paridad* es un bit adicional incluido con un mensaje para hacer que el número total de los 1 sea impar o par. Un mensaje de cuatro bits y un bit de paridad, P, se muestran en la Tabla 1-3. En (a), P se escoge de modo que la suma de todos los números 1 sea impar (un total de cinco bits). En (b), P se escoge de modo que la suma de todos los 1 sea par. Durante la transferencia de información de una localidad a otra, el bit de paridad se manipula como sigue: Al final del envío, el mensaje (en este caso los primeros cuatro bits) se aplica a un circuito “generador de paridad”, donde el bit P requerido se genera. El mensaje, incluyendo el bit de paridad, se transfiere a su destino. En el extremo receptor, todos los bits que llegan (en este caso cinco) se aplican a un circuito “de verificación de paridad” para verificar la apropiada paridad adoptada. Se detecta un error si la paridad verificada no corresponde a la adoptada. El método de paridad detecta la presencia de uno, tres o cualquier combinación impar de errores. Una combinación par de errores no se detecta. En la Sección 4-9 se encontrará un análisis adicional de la generación de paridad y verificación.

El código reflejado

Los sistemas digitales pueden diseñarse para procesar datos sólo en una forma discreta. Muchos sistemas físicos suministran salida de información continua. Esta información puede convertirse en forma digital o discreta antes de que se aplique a un sistema digital. La información continua o analógica se convierte en forma digital mediante un convertidor de analógico a digital. Algunas veces es conveniente usar el código reflejado, que se muestra en la Tabla 1-4, para representar la información digital convertida en una información analógica. La ventaja del código reflejado sobre los números binarios puros es que un número en el código reflejado cambia sólo por un bit conforme proceda de un número al siguiente. Una aplicación típica del código reflejado ocurre cuando la información analógica se representa por el cambio continuo de una posición de eje. El eje está dividido en segmentos y, a cada segmento se le asigna un número. Si se hace que los segmentos adyacentes correspondan a los números adyacentes de código reflejado, la ambigüedad se reduce cuando la detección se percibe en la línea que separa dos segmentos cualesquiera. El código reflejado que se muestra en la Tabla 1-4 es sólo uno de muchos códigos posibles de esta clase. Para obtener un código reflejado diferente, puede comenzarse con cualquier combinación de bits y obtener después la siguiente combinación de bits cambiando sólo un bit de 0 a 1, o de 1 a 0, en cualquier forma que se haga al azar, en tanto que dos números no tengan asignaciones idénticas de código. El código reflejado también se conoce como código *Gray*.

TABLA 1-3 Generación de bits de paridad

(a) Mensaje	P (impar)	(b) Mensaje	P (par)
0000	1	0000	0
0001	0	0001	1
0010	0	0010	1
0011	1	0011	0
0100	0	0100	1
0101	1	0101	0
0110	1	0110	0
0111	0	0111	1
1000	0	1000	1
1001	1	1001	0
1010	1	1010	0
1011	0	1011	1
1100	1	1100	0
1101	0	1101	1
1110	0	1110	1
1111	1	1111	0

TABLA 1-4 Código reflejado de 4 bits

Código reflejado	Equivalencia decimal
0000	0
0001	1
0011	2
0010	3
0110	4
0111	5
0101	6
0100	7
1100	8
1101	9
1111	10
1110	11
1010	12
1011	13
1001	14
1000	15

Códigos alfanuméricos

Muchas de las aplicaciones de las computadoras digitales requieren la manipulación de datos que constan no sólo de números, sino también de letras. Por ejemplo, una compañía de seguros con millones de tenedores de póliza debe usar una computadora digital para procesar sus archivos. Para representar el nombre del tenedor de póliza en forma binaria, es necesario tener un código binario para el alfabeto. Además, el mismo código binario debe representar números decimales y algunos otros caracteres especiales. Un código alfanumérico (algunas veces abreviado *alfamérico*) es un código binario de un grupo de elementos que constan de diez dígitos decimales, las 26 letras del alfabeto y cierto número de símbolos especiales como \$. El número total de elementos en un grupo alfanumérico es mayor de 36. Por lo tanto, debe codificarse con un mínimo de seis bits ($2^6 = 64$, pero $2^5 = 32$ no es suficiente).

Un arreglo posible de un código alfanumérico de seis bits se muestra en la Tabla 1-5 bajo el nombre de “código interno”. Con unas pocas variaciones, se utiliza en muchas computadoras para representar en forma interna los caracteres alfanuméricos. La necesidad de representar más de 64 caracteres (las letras minúsculas y algunos caracteres especiales de control para la transmisión de información digital) dieron lugar a los códigos alfanuméricos de siete y ocho bits. Uno de dichos códigos se conoce como el ASCII (American Standard Code for Information Interchange); otro se conoce como EBCDIC (Extended BCD Interchange Code). El código ASCII que se lista en la Tabla 1-5 consta de siete bits pero es, en la práctica, un código de ocho bits debido a que de manera invariable se agrega un octavo bit por paridad. Cuando se transfiere información discreta a través de tarjetas perforadas, los caracteres alfanuméricos usan un código binario de doce bits. Una tarjeta perforada consta de 80 columnas y 12 renglones. En cada columna se representa un carácter alfanumérico mediante agujeros perforados en los renglones apropiados. Un agujero se detecta como un 1 y la ausencia de perforación como 0. Los 12 renglones están marcados, principiando desde la parte superior, como la perforación 12, 11, 0, 1, 2, ..., 9. Los primeros tres se denominan perforación de *zona* y los últimos nueve se conocen como perforación *numérica*. El código de tarjeta de 12 bits que se muestra en la Tabla 1-5 lista los renglones cuando se perfora un agujero (dando los números 1). Se supone que los renglones restantes no listados son 0. El código de tarjeta de 12 bits es ineficiente con respecto al número de bits que se utilizan. La mayoría de las computadoras traducen el código de entrada a un código externo de 6 bits. Como ejemplo, la representación en el código interno de nombre de “John Doe” es:

100001	100110	011000	100101	110000	010100	100110	010101
J	O	H	N	espacio	D	O	E

1-7 ALMACENAMIENTO BINARIO Y REGISTROS

Los elementos discretos de información en una computadora digital deben tener una existencia física en ciertos medios de almacenamiento de información. Además,

TABLA 1-5 Códigos de caracteres alfanuméricos

Carácter	Código interno		Código ASCII		Código EBCDIC	Código de tarjeta
	6-Bit	7-Bit	7-Bit	8-Bit	12-Bit	
A	010 001	100 0001	1100 0001	1100 0001	12,1	
B	010 010	100 0010	1100 0010	1100 0010	12,2	
C	010 011	100 0011	1100 0011	1100 0011	12,3	
D	010 100	100 0100	1100 0100	1100 0100	12,4	
E	010 101	100 0101	1100 0101	1100 0101	12,5	
F	010 110	100 0110	1100 0110	1100 0110	12,6	
G	010 111	100 0111	1100 0111	1100 0111	12,7	
H	011 000	100 1000	1100 1000	1100 1000	12,8	
I	011 001	100 1001	1100 1001	1100 1001	12,9	
J	100 001	100 1010	1101 0001	1101 0001	11,1	
K	100 010	100 1011	1101 0010	1101 0010	11,2	
L	100 011	100 1100	1101 0011	1101 0011	11,3	
M	100 100	100 1101	1101 0100	1101 0100	11,4	
N	100 101	100 1110	1101 0101	1101 0101	11,5	
O	100 110	100 1111	1101 0110	1101 0110	11,6	
P	100 111	101 0000	1101 0111	1101 0111	11,7	
Q	101 000	101 0001	1101 1000	1101 1000	11,8	
R	101 001	101 0010	1101 1001	1101 1001	11,9	
S	110 010	101 0011	1110 0010	1110 0010	0,2	
T	110 011	101 0100	1110 0011	1110 0011	0,3	
U	110 100	101 0101	1110 0100	1110 0100	0,4	
V	110 101	101 0110	1110 0101	1110 0101	0,5	
W	110 110	101 0111	1110 0110	1110 0110	0,6	
X	110 111	101 1000	1110 0111	1110 0111	0,7	
Y	111 000	101 1001	1110 1000	1110 1000	0,8	
Z	111 001	101 1010	1110 1001	1110 1001	0,9	
0	000 000	011 0000	1111 0000	1111 0000	0	
1	000 001	011 0001	1111 0001	1111 0001	1	
2	000 010	011 0010	1111 0010	1111 0010	2	
3	000 011	011 0011	1111 0011	1111 0011	3	
4	000 100	011 0100	1111 0100	1111 0100	4	
5	000 101	011 0101	1111 0101	1111 0101	5	
6	000 110	011 0110	1111 0110	1111 0110	6	
7	000 111	011 0111	1111 0111	1111 0111	7	
8	001 000	011 1000	1111 1000	1111 1000	8	
9	001 001	011 1001	1111 1001	1111 1001	9	
espacio	110 000	010 0000	0100 0000	0100 0000	sin perforación	
.	011 011	010 1110	0100 1011	0100 1011	12,8,3	
(111 100	010 1000	0100 1101	0100 1101	12,8,5	
+	010 000	010 1011	0100 1110	0100 1110	12,8,6	
\$	101 011	010 0100	0101 1011	0101 1011	11,8,3	
*	101 100	010 1010	0101 1100	0101 1100	11,8,4	
)	011 100	010 1001	0101 1101	0101 1101	11,8,5	
-	100 000	010 1101	0110 0000	0110 0000	11	
/	110 001	010 1111	0110 0001	0110 0001	0,1	
,	111 011	010 1100	0110 1011	0110 1011	0,8,3	
=	001 011	011 1101	0111 1110	0111 1110	8,6	

cuando los elementos discretos de información se representan en forma binaria, el medio de almacenamiento de la información debe contener elementos binarios de almacenamiento, para almacenar bits individuales. Una *celda binaria* es un dispositivo que posee dos estados estables y es capaz de almacenar un bit de información. La entrada a la celda recibe señales de excitación que la fijan a uno de los dos estados. La salida de la celda es una cantidad física que distingue entre los dos estados. La información almacenada en una celda es 1 cuando está en un estado estable y 0 cuando está en el otro estado estable. Los ejemplos de celdas binarias son circuitos flip-flop electrónicos, núcleos de ferrita usados en memorias y posiciones perforadas con un agujero o no perforadas en una tarjeta.

Registros

Un *registro* es un grupo de celdas binarias. Ya que una celda almacena un bit de información, se deduce que un registro con n celdas puede almacenar cualquier cantidad discreta de información que contenga n bits. El *estado* de un registro es un número múltiplo n de 1 y 0, con cada bit designando el estado en una celda en el registro. El *contenido* de un registro es una función de la interpretación dada a la información que está almacenada en él. Considérese, por ejemplo, el siguiente registro de 16 celdas:

1	1	0	0	0	0	1	1	1	1	0	0	1	0	0	1
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

En forma física, puede pensarse que el registro está compuesto de 16 celdas binarias, con cada celda almacenando ya sea un 1 o un 0. Supóngase que la configuración de bits almacenada en el registro es como se muestra. El estado de registro es el número múltiplo 16, 1100001111001001. Claramente, un registro con n celdas puede estar en uno de 2^n estados posibles. Ahora bien, si se supone que el contenido del registro representa un entero binario, entonces es obvio que el registro puede almacenar cualquier número binario desde 0 a $2^{16} - 1$. Para el ejemplo particular que se expuso, el contenido del registro es el equivalente binario del número decimal 50 121. Si se supone que el registro almacena caracteres alfanuméricos en un código de ocho bits, el contenido del registro es cualquiera de dos caracteres significativos (combinaciones de bits no asignadas no representan información significativa). En el código EBCDIC, el ejemplo anterior representa los dos caracteres C (los ocho bits de la izquierda) e I (los ocho bits de la derecha). Por otra parte, si se interpreta que el contenido del registro es cuatro dígitos decimales representados por un código de cuatro bits, el contenido del registro es un número decimal de cuatro dígitos. En el código exceso-3, el ejemplo anterior es el número decimal 9 096. El contenido de registro no tiene significado en el BCD, ya que la combinación de bits 1100 no está asignada a cualquier dígito decimal. Por este ejemplo, es claro que un registro puede almacenar uno o más elementos discretos de información y que la misma configuración de bits puede interpretarse en forma diferente para distintos tipos de elementos de información. Es importante que el usuario almacene información con significado en los registros y que la computadora esté programada para procesar esta información de acuerdo con el *tipo* de información almacenada.

Transferencia de registro

Una computadora digital se caracteriza por sus registros. La unidad de memoria (Fig. 1-1) simplemente es una colección de miles de registros para almacenar información digital. La unidad de proceso está compuesta de diversos registros que almacenan operandos bajo los cuales se realizan operaciones. La unidad de control utiliza registros para mantener cuenta de diversas secuencias de computación y, cada dispositivo de entrada o salida debe tener cuando menos un registro para almacenar la información transferida al dispositivo o desde éste. Una operación de *transferencia interregistro*, una operación básica en sistemas digitales, consiste en la transferencia de información almacenada en un registro a otro. En la Fig. 1-2 se ilustra la transferencia de información entre registros y se demuestra en forma gráfica la transferencia de información binaria de un teletipo de teletipo a un registro en la unidad de memoria. Se supone que la unidad de teletipo de entrada tiene un teclado, un circuito de control y registros de entrada. Cada vez que se oprime una tecla, el control introduce al registro de entrada un carácter de código alfanumérico de ocho bits. Se supone que el código usado es el código ASCII con un octavo bit de paridad impar. La información del registro de entrada se transfiere en las ocho celdas menos significativas de un registro

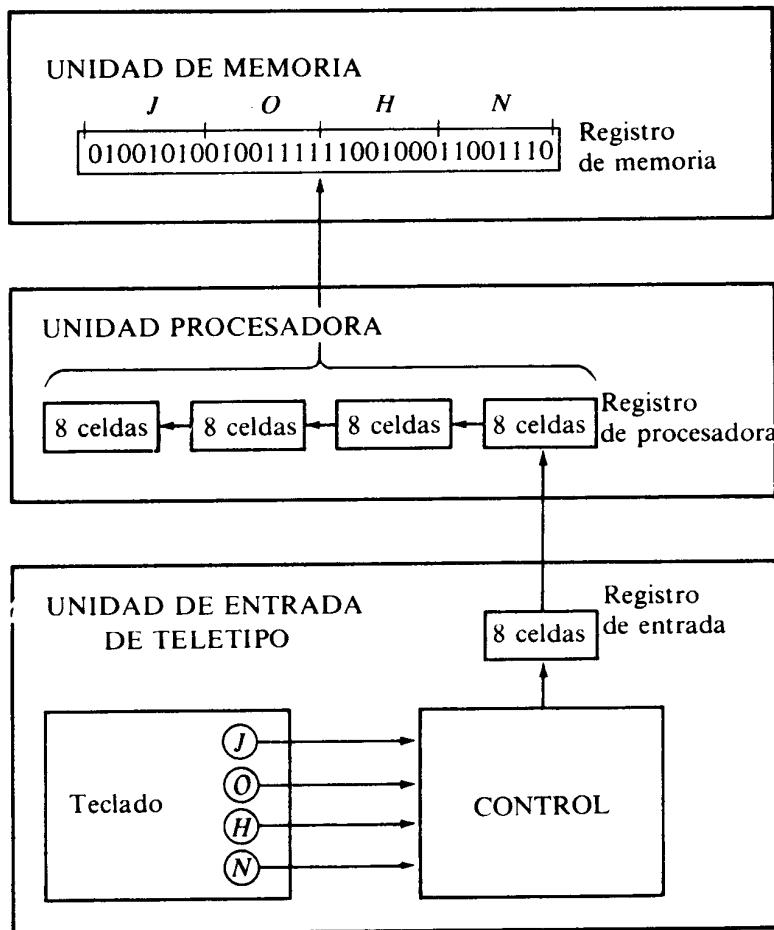


Figura 1-2 Transferencia de información con registros

de procesador. Después de cada transferencia, el registro de entrada se limpia para posibilitar que el control inserte un nuevo código de ocho bits cuando se oprimen otra vez las teclas. Cada carácter de ocho bits transferidos al registro del procesador está precedido por un corrimiento del carácter anterior a las siguientes ocho celdas a su izquierda. Cuando se completa una transferencia de cuatro caracteres, el registro del procesador está lleno y su contenido se transfiere a un registro de memoria. El contenido almacenado en el registro de memoria que se muestra en la Fig. 1-2 proviene de la transferencia de los caracteres JOHN después de que se han oprimido las cuatro teclas apropiadas.

Para procesar cantidades discretas de información en la forma binaria, una computadora debe estar provista con (1) dispositivos que retengan los datos que van a procesar y (2) elementos de circuito que manipulen los bits individuales de información. El dispositivo de uso más común para retener los datos es un registro. La manipulación de variables binarias se hace mediante circuitos lógicos digitales. En la Fig. 1-3 se ilustra el proceso de sumar dos números binarios de 10 bits. La unidad de memoria, que en forma normal consta de miles de registros, se muestra en el diagrama

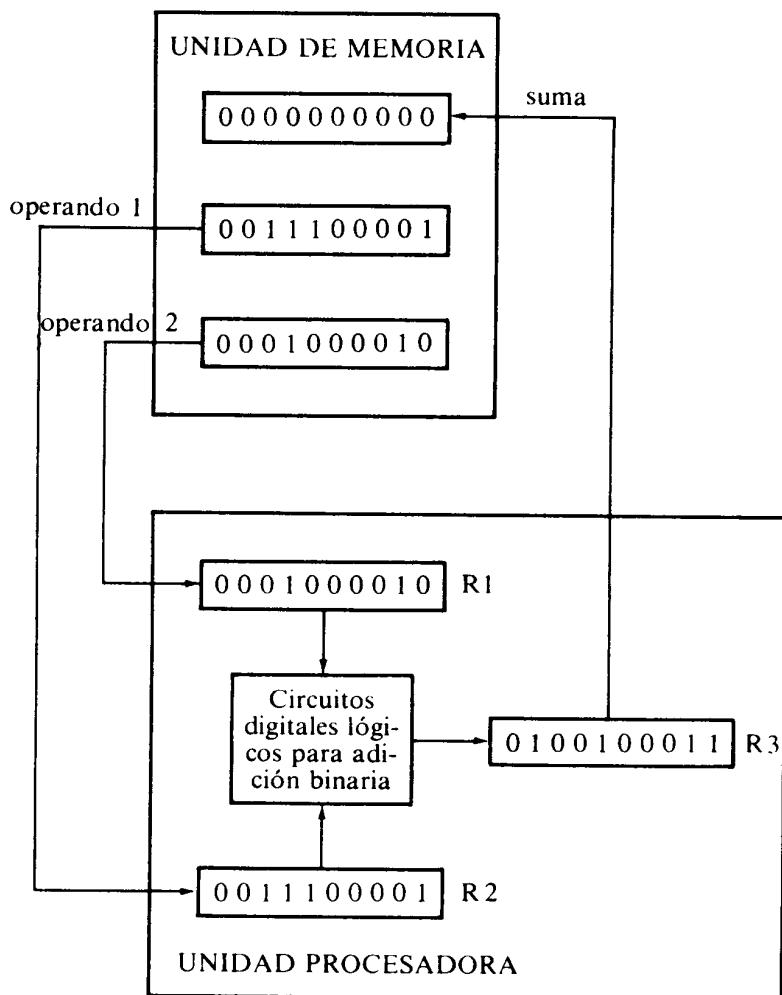


Figura 1-3 Ejemplo de procesamiento de información binaria.

con sólo tres de sus registros. La parte de la unidad de proceso que se presenta consta de tres registros, R1, R2 y R3, junto con los circuitos digitales que manipulan los bits de R1 y R2 y transfieren a R3 un número binario igual a su suma aritmética. Los registros de memoria almacenan información y son incapaces de procesar los dos operandos. Sin embargo, la información almacenada en memoria puede transferirse a los registros del procesador. Los resultados obtenidos en los registros del procesador pueden transferirse devolviéndolos a un registro de memoria para almacenamiento, hasta que se necesiten otra vez. El diagrama muestra los contenidos de los dos operandos transferidos de los dos registros de memoria a R1 y R2. Los circuitos lógicos digitales producen la suma, que se transfiere al registro R3. Los contenidos de R3 pueden transferirse ahora devolviéndolos a uno de los registros de memoria.

Los últimos dos ejemplos demuestran las capacidades de flujo de información de un sistema digital en una forma muy simple. Los registros del sistema son los elementos básicos para almacenar y retener la información binaria. Los circuitos lógicos digitales procesan la información. Los circuitos lógicos digitales y sus capacidades de manipulación se introducen en la sección siguiente. Los registros y la memoria se presentan en el Capítulo 7.

1-8 LOGICA BINARIA

La lógica binaria trata con variables que toman dos valores discretos y con operaciones que tienen significado lógico. Los dos valores que toman las variables pueden designarse con nombres diferentes (esto es, *verdadero* y *falso*, *si* y *no*, etc.), pero para este propósito no es conveniente pensar en términos de bits y asignarles los valores de 1 y 0. La lógica binaria se usa para describir, en forma matemática, la manipulación y el proceso de la información binaria. Es en particular adecuada para el análisis y diseño de sistemas digitales. Por ejemplo, los circuitos lógicos digitales en la Fig. 1-3 que llevan a cabo la aritmética binaria, son circuitos cuyo comportamiento se expresa en la forma más conveniente mediante variables binarias y operaciones lógicas. La lógica binaria se introduce en esta sección y es equivalente a un álgebra llamada booleana. La presentación formal de un álgebra booleana de dos valores se cubre con más detalle en el Capítulo 2. El propósito de esta sección es introducir el álgebra booleana en una forma heurística y relacionarla con los circuitos lógicos digitales y las señales binarias.

Definición de la lógica binaria

La lógica binaria consta de variables binarias y operaciones lógicas. Las variables se denotan con letras del alfabeto como *A*, *B*, *C*, *x*, *y*, *z*, etc., y cada variable tiene dos y sólo dos valores posibles distintos: 1 y 0. Hay tres operaciones lógicas básicas: AND, OR y NOT.

1. AND (Y): esta operación se representa mediante un punto o por la ausencia de operador. Por ejemplo, $x \cdot y = z$ o $xy = z$ se lee “*x Y y es igual a z*”. La operación lógica AND se interpreta con el significado $z = 1$ si y sólo si $x = 1$ y

$y = 1$; en cualquier otro caso $z = 0$. (Recuérdese que x , y y z son variables binarias y pueden ser iguales a 1 o 0 y a nada más.)

2. OR (O): Esta operación se representa mediante un signo de suma. Por ejemplo, $x + y = z$ se lee “ x O y es igual a z ”, lo cual significa que $z = 1$ si $x = 1$ o si $y = 1$ o si tanto $x = 1$ como $y = 1$. Si tanto $x = 0$ como $y = 0$, entonces $z = 0$.
3. NOT (NO): Esta operación está representada por una sola comilla (algunas veces por una barra). Por ejemplo, $x' = z$ ($\bar{x} = z$) se lee “ x no es igual a z ”, significa que z es lo que x no es. En otras palabras, si $x = 1$, entonces $z = 0$; pero si $x = 0$, entonces $z = 1$.

La lógica binaria es semejante a la aritmética binaria y, las operaciones AND y OR tienen ciertas similitudes con la multiplicación y la suma, respectivamente. De hecho, los símbolos que se utilizan para AND y OR son los mismos que se usan para la multiplicación y la suma. Sin embargo, la lógica binaria no debe confundirse con la aritmética binaria. Debe tomarse en cuenta que una variable aritmética denota un número que puede constar de muchos dígitos. Una variable lógica es siempre ya sea 1 o 0. Por ejemplo, en la aritmética binaria se tiene $1 + 1 = 10$ (se lee “uno más uno es igual a 2”), en tanto que en la lógica binaria se tiene $1 + 1 = 1$ (se lee: “uno OR uno es igual a uno”).

Para cada combinación de los valores de x y y , hay un valor de z especificado por la definición de la operación lógica. Estas definiciones pueden listarse en forma compacta usando las *tablas de verdad*. Una tabla de verdad es una tabla de todas las combinaciones posibles de las variables, que muestra la relación entre los valores que pueden tomar las variables y el resultado de la operación. Por ejemplo, las tablas de verdad para las operaciones AND y OR con las variables x y y se obtienen haciendo la lista de todos los valores posibles que pueden tener las variables cuando se combinan en pares. El resultado de la operación para cada combinación se lista entonces en una columna separada. Las tablas de verdad para AND, OR y NOT se listan en la Tabla 1-6. Estas tablas demuestran en forma clara las definiciones de las operaciones.

Circuitos con interruptores y señales binarias

El uso de las variables binarias y la aplicación de la lógica binaria se demuestra por los circuitos con interruptores simples en la Fig. 1-4. Permítase que los interruptores manuales A y B representan dos variables binarias cuyos valores son iguales a 0 cuando el interruptor está abierto y 1 cuando el interruptor está cerrado. En forma semejante, permítase que la lámpara L represente una tercera variable binaria igual a 1 cuando la luz esté encendida y 0 cuando esté apagada. Para los interruptores en serie, la luz se enciende y A y B están cerrados. Para los interruptores en paralelo, la luz se enciende si A o B está cerrado. Es obvio que los dos circuitos pueden expresarse mediante la lógica binaria con las operaciones Y y O, respectivamente:

$$L = A \cdot B \text{ para el circuito en la Fig. 1-4(a)}$$

$$L = A + B \text{ para el circuito en la Fig. 1-4(b)}$$

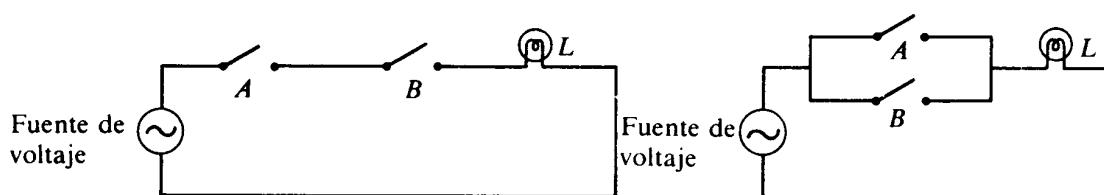
TABLA 1-6 Tablas de verdad de operaciones lógicas

AND		OR		NOT			
x	y	$x \cdot y$	x	y	$x + y$	x	x'
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

Algunas veces los circuitos digitales electrónicos se denominan *circuitos interruptores* (o circuitos conmutadores) debido a que se comportan como un interruptor, con el elemento activo, por ejemplo un transistor que conduce (interruptor cerrado) o bien no conduce (interruptor abierto). En lugar de cambiar manualmente el interruptor, un circuito interruptor electrónico usa señales binarias para controlar los estados de conducción o no conducción del elemento activo. Las señales eléctricas como voltajes o corriente existen a través de un sistema digital ya sea en uno de dos valores reconocibles (excepto durante la transición). Los circuitos operados por voltaje, por ejemplo, responden a dos niveles separados de voltaje que representan una variable binaria igual a lógica 1 o lógica 0. Por ejemplo, un sistema digital particular puede definir la lógica 1 como una señal con un valor nominal de 3 voltos y, lógica 0 como una señal con un valor nominal de 0 volt. Como se muestra en la Fig. 1-5, cada nivel de voltaje tiene una desviación aceptable de la nominal. La región intermedia entre las regiones permitidas se cruza sólo durante las transiciones de estado. Las terminales de entrada de los circuitos digitales aceptan señales binarias con las tolerancias permitidas y responden a la terminal de salida con señales binarias que caen dentro de las tolerancias especificadas.

Compuertas lógicas

Los circuitos digitales electrónicos también se denominan *circuitos lógicos* ya que, con la entrada apropiada, establecen trayectorias lógicas de manipulación. Cualquier información que se deseé para computación o control puede operarse por el paso de señales binarias a través de diversas combinaciones de circuitos lógicos, cada señal



(a) Interruptores en serie – lógica AND

(b) Interruptores en paralelo – lógica OR

Figura 1-4 Circuitos interruptores que demuestran la lógica binaria.

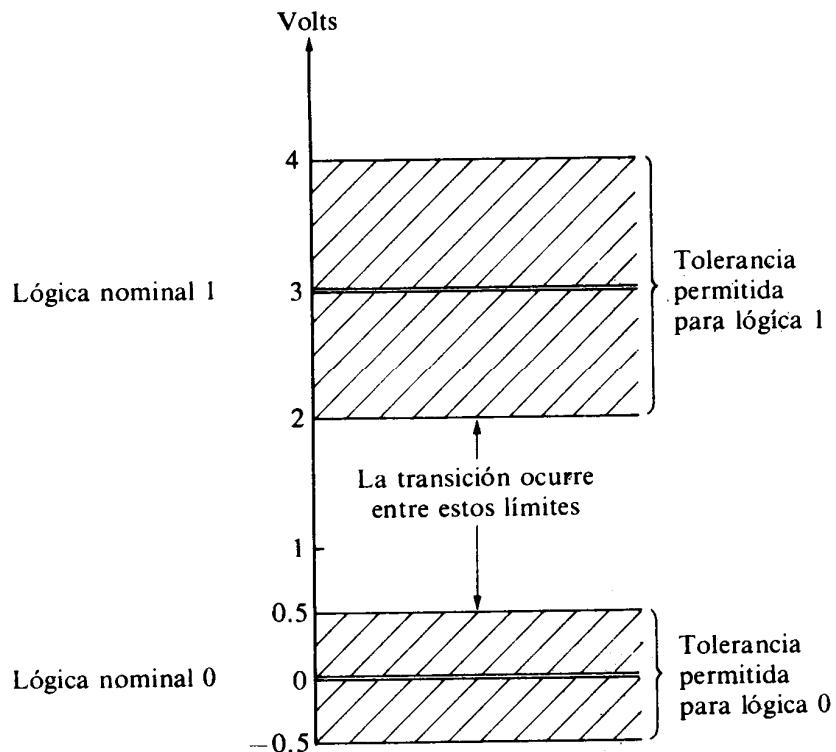


Figura 1-5 Ejemplo de señales binarias.

representa una variable y lleva un bit de información. Los circuitos lógicos que realizan las operaciones lógicas de AND, OR y NOT se muestran con sus símbolos en la Fig. 1-6. Estos circuitos, llamados *compuertas*, son bloques de hardware que producen una señal de salida lógica 1 o lógica 0 y se satisfacen los requisitos de la entrada lógica. Obsérvese que se han utilizado cuatro nombres diferentes para el mismo tipo de circuitos: circuitos digitales, circuitos interruptores, circuitos lógicos y compuertas. Todos los cuatro nombres tienen uso amplio, pero aquí se hará referencia a los circuitos como compuertas AND, OR y NOT. Algunas veces la compuerta NOT se denomina *circuito inversor* ya que invierte una señal binaria.

Las señales de entrada x y y en las dos compuertas de entrada en la Fig. 1-6 pueden existir en uno de cuatro estados posibles: 00, 10, 11 o 01. Estas señales de entrada se muestran en la Fig. 1-7, junto con las señales de salida para las compuertas AND y OR. Los diagramas de tiempo en la Fig. 1-7 ilustran la respuesta de cada circuito a cada una de las cuatro combinaciones binarias de entrada posibles. La razón del nombre “inversor” para la compuerta NOT es aparente por la comparación de la señal x (entrada del inversor) y la de x' (salida del inversor).

Las compuertas AND y OR pueden tener más de dos entradas. Una compuerta AND con tres entradas y una compuerta OR con cuatro entradas se muestran en la Fig. 1-6. La compuerta de tres entradas AND responde con una salida lógica 1 si todas las tres señales de entrada son de lógica 1. La salida produce una señal de lógica 0 si cualquier entrada es lógica 0. Las cuatro entradas en la compuerta OR responden con una lógica 1 cuando cualquier entrada es lógica 1. Su salida llega a ser lógica 0 si todas las señales de entrada son lógica 0.

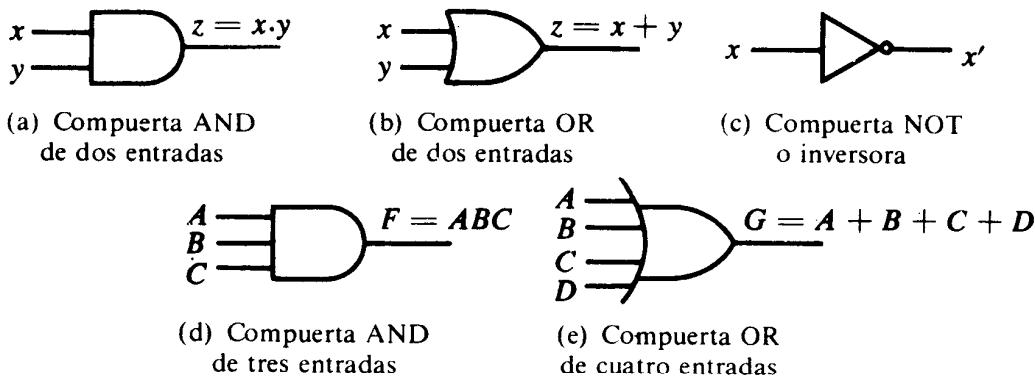


Figura 1-6 Símbolos para los circuitos digitales lógicos.

El sistema matemático de la lógica binaria es mejor conocido como álgebra booleana. Esta álgebra se usa en forma conveniente para describir la operación de redes complejas de circuitos digitales. Los diseñadores de sistemas digitales utilizan el álgebra booleana para transformar los diagramas de circuitos en expresiones algebraicas y viceversa. Los Capítulos 2 y 3 se dedican al estudio del álgebra booleana, sus propiedades y capacidades de manipulación. En el capítulo 4 se muestra cómo puede usarse el álgebra booleana para expresar en forma matemática las interconexiones entre redes de compuertas.

1-9 CIRCUITOS INTEGRADOS

Los circuitos digitales en forma invariable se construyen con circuitos integrados. Un circuito integrado (abreviado IC) es un cristal semiconductor pequeño de silicio, llamado *pastilla*, que contiene componentes eléctricos como transistores, diodos, resistores y capacitores. Los diversos componentes están interconectados dentro de la pastilla para formar un circuito electrónico. La pastilla se monta en un paquete de metal o plástico y se soldan conexiones a las clavijas externas para formar el IC. Los circuitos integrados difieren de otros circuitos electrónicos compuestos de componentes desprendibles en que los componentes individuales de un IC no pueden separarse o

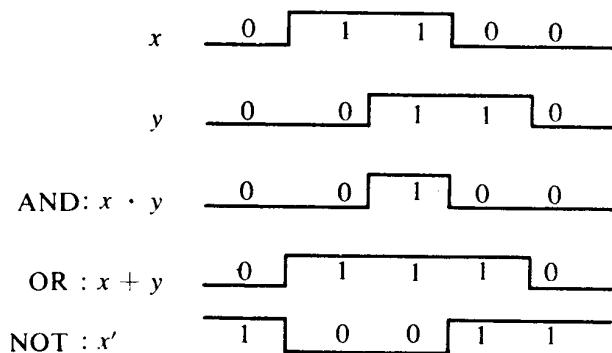


Figura 1-7 Señales de entrada-salida para las compuertas (a), (b) y (c) en la Fig. 106.

desconectarse y el circuito en el interior del paquete es accesible sólo a través de las clavijas externas.

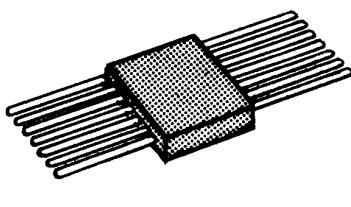
Los circuitos integrados se obtienen en dos tipos de paquetes: el paquete *plano* y el paquete *dual en línea* (DIP) como se muestra en la Fig. 1-8. El paquete dual en línea es el tipo de mayor uso debido a su precio bajo y fácil instalación en tableros para conectar circuitos. La envolvente del paquete IC se hace de plástico o cerámica. La mayoría de los paquetes tienen tamaño estándar y el número de clavijas varía desde 8 a 64. Cada IC tiene una denominación numérica impresa en la superficie del paquete para su identificación. Cada vendedor publica un libro o catálogo con información que proporciona los datos necesarios que conciernen a los diversos productos.

El tamaño de los paquetes IC es muy pequeño. Por ejemplo, cuatro compuertas AND están encerradas dentro de un paquete dual en línea de 14 clavijas con dimensiones de $20 \times 8 \times 3$ milímetros. Un microprocesador entero se encuentra dentro de un paquete dual en línea de 40 clavijas con dimensiones de $50 \times 15 \times 4$ milímetros.

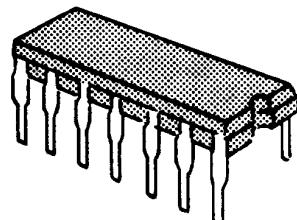
A parte de una reducción sustancial en tamaño, los IC ofrecen otras ventajas y beneficios en comparación con los circuitos electrónicos hechos de componentes discretos. El costo de los IC es muy bajo, lo que los hace económicos para su utilización. Su consumo reducido de potencia hace que el sistema digital tenga una operación más económica. Tienen una alta confiabilidad contra fallas, de modo que el sistema digital necesita menos reparaciones. La velocidad de operación es más alta, lo cual los hace adecuados para operaciones de alta velocidad. El uso de los IC reduce el número de conexiones de alambrado externas, debido a que muchas de las conexiones están en el interior del paquete. Debido a todas estas ventajas, los sistemas digitales siempre se construyen con circuitos integrados.

Los circuitos integrados se clasifican en dos categorías generales, *lineales* y *digitales*. Los IC lineales operan con señales continuas para proporcionar funciones electrónicas como amplificadores y comparadores de voltaje. Los circuitos integrados digitales operan con señales binarias y están hechos de compuertas digitales interconectadas. Aquí el interés se centra sólo en los circuitos integrados digitales.

Conforme ha mejorado la tecnología de los IC, el número de compuertas que pueden colocarse dentro de una sola pastilla de silicio ha aumentado en forma considerable. La diferenciación entre los IC que tienen unas cuantas compuertas internas y los que tienen decenas o cientos de compuertas, se hace por una referencia acostumbrada de que un paquete es un dispositivo de pequeña, mediana o gran escala



Paquete plano



Paquete dual en línea

Figura 1-8 Paquetes de circuitos integrados.

de integración. Varias compuertas lógicas en un solo paquete hacen un dispositivo con integración a pequeña escala (SSI). Para calificar como un dispositivo de integración a media escala (MSI), el IC debe realizar una función lógica completa y tener una complejidad de 10 a 100 compuertas. Un dispositivo de integración a gran escala (LSI) lleva a cabo una función lógica con más de 100 compuertas. También hay dispositivos de integración a muy alta escala (VLSI) que contienen miles de compuertas en una sola pastilla.

Muchos de los diagramas de circuitos digitales que se consideran en este libro se muestran en detalle hasta las compuertas individuales y sus conexiones. Dichos diagramas son útiles para demostrar la construcción lógica de una función particular. Sin embargo, debe tenerse en cuenta que, en la práctica, la función puede obtenerse por un dispositivo MSI o LSI, y el usuario tiene acceso a las entradas y salidas externas pero no a las entradas y salidas de las compuertas intermedias. Por ejemplo, un diseñador que desea incorporar un registro en su sistema es más probable que escoja una función de esta clase de un circuito MSI disponible, en lugar de diseñarlo con circuitos digitales individuales como puede mostrarse en un diagrama.

BIBLIOGRAFIA

1. Richard, R. K., *Arithmetic Operations in Digital Computers*. New York: Van Nostrand Co., 1955.
2. Flores, I., *The Logic of Computer Arithmetic*. Englewood Cliffs, N. J.: Prentice-Hall, Inc., 1963.
3. Chu, Y., *Digital Computer Design Fundamentals*. New York: McGraw-Hill Book Co., 1962, Caps. 1 y 2.
4. Kostopoulos, G. K., *Digital Engineering*. New York: John Wiley & Sons, Inc., 1975, Cap. 1.
5. Rhyne, V. T., *Fundamentals of Digital Systems Design*. Englewood Cliffs, N. J.: Prentice-Hall, Inc., 1973, Cap. 1.

PROBLEMAS

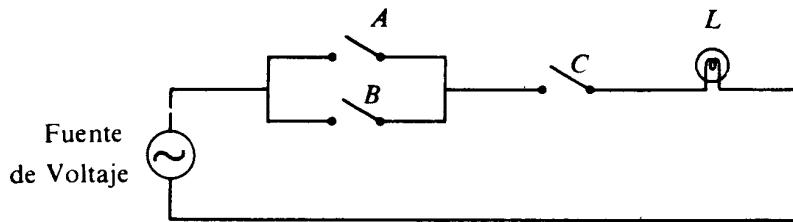
- ✓ 1-1. Escriba los primeros 20 dígitos decimales en base 3.
- ✓ 1-2. Sume y multiplique los siguientes números en la base dada sin convertirlos en decimales.
 - (a) $(1230)_4$ y $(23)_4$
 - (b) $(135.4)_6$ y $(43.2)_6$
 - (c) $(367)_8$ y $(715)_8$
 - (d) $(296)_{12}$ y $(57)_{12}$
- ✓ 1-3. Convierta el número decimal 250.5 en base 3, base 4, base 7, base 8 y base 16.
- ✓ 1-4. Convierta los siguientes números decimales en binarios: 12.0625, 10^4 , 673.23 y 1998.
- ✓ 1-5. Convierta los siguientes números binarios en decimales:
10.10001, 101110.0101, 1110101.110, 1101101.111.
- ✓ 1-6. Convierta los siguientes números de la base dada en las bases indicadas:
 - (a) decimal 225.225 en binario, octal y hexadecimal

- (b) binario 11010111.110 en decimal, octal y hexadecimal
- (c) octal 623.77 en decimal, binario y hexadecimal
- (d) hexadecimal 2AC5.D en decimal, octal y binario

- ✓ 1-7. Convierta los siguientes números en decimales:
- (a) $(1001001.011)_2$
 - (e) $(0.342)_6$
 - (b) $(12121)_3$
 - (f) $(50)_7$
 - (c) $(1032.2)_4$
 - (g) $(8.3)_9$
 - (d) $(4310)_5$
 - (h) $(198)_{12}$
- 1-8. Obtenga los complementos de 1 y de 2 de los siguientes números binarios: 1010101, 0111000, 0000001, 10000, 00000.
- 1-9. Obtenga los complementos de 9 y de 10 de los siguientes números decimales: 13579, 09900, 90090, 10000, 00000.
- 1-10. Encuentre el complemento de 10 de $(935)_{11}$.
- 1-11. Lleve a cabo la resta con los siguientes números decimales usando (1) el complemento de 10 y (2) el complemento de 9. Verifique la respuesta por resta directa.
- (a) $5250 - 321$
 - (c) $753 - 864$
 - (b) $3570 - 2100$
 - (d) $20 - 1000$
- 1-12. Haga la resta con los siguientes números binarios usando (1) el complemento de 2 y (2) el complemento de 1. Verifique la respuesta por resta directa.
- (a) $11010 - 1101$
 - (c) $10010 - 10011$
 - (b) $11010 - 10000$
 - (d) $100 - 110000$
- 1-13. Demuestre el procedimiento establecido en la Sección 1-5 para la sustracción de dos números con complemento de $(r - 1)$.
- 1-14. Para los códigos pesados (a) 3, 3, 2, 1 y (b) 4, 4, 3, – 2 para dígitos decimales determine todas las tablas posibles, de modo que el complemento de 9 de cada dígito decimal se obtenga por el cambio de 1 a 0 y de 0 a 1.
- 1-15. Represente el número decimal 8620 (a) en BCD, (b) en el código exceso-3, (c) en el código 2, 4, 2, 1 y (d), como un número binario.
- 1-16. Un código binario usa diez bits para representar cada uno de los diez dígitos decimales. Cada dígito está asignado a un código de nueve números 0 y un 1. El código para el dígito 6, por ejemplo, es 0001000000. Determine el código binario para los dígitos decimales restantes.
- 1-17. Obtenga el código pesado binario para los dígitos en base 12 usando pesos de 5421.
- 1-18. Determine el bit de paridad-impar generado cuando el mensaje consta de diez dígitos decimales en el código 8, 4, – 2, – 1.
- 1-19. Determine otras dos combinaciones para un código reflejado diferente al que se muestra en la Tabla 1-4.
- 1-20. Obtenga un código binario para representar todos los dígitos base 6 de modo que el complemento de 5 se obtenga por el reemplazo de 1 por 0 y 0 por 1 en los bits del código.
- 1-21. Asigne un código binario en cierta manera ordenada de los 52 naipes de la baraja. Utilice el número mínimo de bits.
- 1-22. Escriba su primer nombre, la inicial del segundo y su apellido paterno en un código de ocho bits hecho con los siete bits ASCII de la Tabla 1-5 y un bit de paridad par en la

posición más significativa. Incluya espacios en blanco entre nombres y un punto después de la inicial del segundo nombre.

- 1-23. Muestre la configuración de bits de un registro de 24 celdas cuando su contenido representa (a) el número $(295)_{10}$ en binario, (b) el número decimal 295 en BCD y (c), los caracteres XY5 en EBCDIC.
- 1-24. El estado de un registro de 12 celdas es 010110010111. ¿Cuál es su contenido si representa (a) tres dígitos decimales en BCD, (b) tres dígitos decimales en el código exceso-3, (c) tres dígitos decimales en el código 2, 4, 2, 1 y (d), dos caracteres en el código interno de la Tabla 1-5?
- 1-25. Muestre el contenido de todos los registros en la Fig. 1-3 si los dos números binarios que se agregan tienen el equivalente decimal de 257 y 1050. (Suponga que hay registros con 11 celdas.)
- 1-26. Exprese el siguiente circuito de interruptores en notación lógica binaria.



- 1-27. Muestre las señales (mediante un diagrama similar al de la Fig. 1-7) de las salidas F y G en la Fig. 1-6. Utilice señales binarias arbitrarias para las entradas A, B, C y D.