EJEMPLO 2-5

Convierta el número $B2F_{16}$ en decimal.

Solución

$$\begin{aligned} B2F_{16} &= B \times 16^2 + 2 \times 16^1 + F \times 16^0 \\ &= 11 \times 256 + 2 \times 16 + 15 \\ &= 2863_{10} \end{aligned}$$

Resumen de las conversiones

En estos momentos es probable que su cabeza esté dando vueltas a medida que trata de mantener el sentido con todas estas distintas conversiones de un sistema numérico a otro. Tal vez se haya dado cuenta que muchas de estas conversiones pueden realizarse en forma *automática* en su calculadora con sólo oprimir una tecla, pero es importante que las domine para que pueda comprender el proceso. Además, ¿qué pasaría si su calculadora se quedara sin energía en un momento crucial y no tuviera un reemplazo a la mano? El siguiente resumen le ayudará, pero nada se compara con la práctica continua.

1. Al convertir de binario (o hexadecimal) a decimal, utilice el método de tomar la suma ponderada de la posición de cada bit.
2. Al convertir de decimal a binario (o hexadecimal), utilice el método de la división repetida entre 2 (o 16) y recolectar los residuos (figura 2-1).
3. Al convertir de binario a hexadecimal, divida el número en grupos de cuatro bits y convierta cada grupo en el dígito hexadecimal correcto.
4. Al convertir de hexadecimal a binario, convierta cada dígito en su equivalente de cuatro bits.

PREGUNTAS DE REPASO

1. Convierta el número $24CE_{16}$ en decimal.
2. Convierta el número 3117_{10} en hexadecimal y después de hexadecimal a binario.
3. Convierta el número 1001011110110101_2 en hexadecimal.
4. Escriba los siguientes cuatro números en esta secuencia de conteo hexadecimal: E9A, E9B, E9C, E9D, _____, _____, _____, _____.
5. Convierta el número 3527_{16} en binario.
6. ¿Qué rango de valores decimales puede representarse mediante un número hexadecimal de cuatro dígitos?

2-4 CÓDIGO BCD

Cuando se representan números, letras o palabras mediante un grupo especial de símbolos, decimos que están siendo codificados, y al grupo de símbolos se le llama *código*. Tal vez uno de los códigos más conocidos sea el Morse, en el cual una seria de puntos y rayas representan las letras del alfabeto.

Hemos visto que cualquier número decimal puede representarse mediante un número binario equivalente. El grupo de 1s y 0s en el número binario puede considerarse como un código que representa el número decimal. Cuando un número decimal se representa por su número binario equivalente, le llamamos **código binario directo**.

Todos los sistemas digitales utilizan cierta forma de números binarios para su operación interna, pero el mundo externo es decimal por naturaleza. Esto significa que con frecuencia se realizan conversiones entre los sistemas decimal y binario. Hemos visto que las conversiones entre decimal y binario pueden volverse extensas y complicadas cuando se manejan números grandes. Por esta razón, en ciertas situaciones se utiliza un medio para codificar números decimales que combina algunas características tanto del sistema decimal como del sistema binario.

Código decimal codificado en binario

Si *cada* dígito de un número decimal se representa mediante su equivalente binario, el resultado es un código que se conoce como **decimal codificado en binario** (que en lo sucesivo abreviaremos como BCD). Como un dígito decimal puede llegar hasta el 9, se requieren cuatro bits para codificar cada dígito (el código binario para el 9 es 1001).

Para ilustrar el código BCD, considere como ejemplo el número decimal 874. Cada *dígito* se cambia a su equivalente binario de la siguiente manera:

8	7	4	(decimal)
↓	↓	↓	
1000	0111	0100	(BCD)

Como segundo ejemplo, vamos a cambiar el número 943 a su representación en código BCD:

9	4	3	(decimal)
↓	↓	↓	
1001	0100	0011	(BCD)

Una vez más, cada dígito decimal se cambia a su equivalente binario directo. Observe que *siempre* se utilizan cuatro bits para cada dígito.

Así, el código BCD representa cada dígito del número decimal mediante un número binario de cuatro bits. Es evidente que sólo se utilizan los números binarios de cuatro bits del 0000 al 1001. El código BCD no utiliza los números 1010, 1011, 1100, 1101, 1110 y 1111. En otras palabras, sólo se utilizan 10 de los 16 posibles grupos de código binario de cuatro bits. Si llega a aparecer uno de los números "prohibidos" de cuatro bits en una máquina que utilice el código BCD, por lo general, es una indicación de que se produjo un error.

EJEMPLO 2-6

Convierta el número 0110100000111001 (BCD) en su equivalente decimal.

Solución

Divida el número BCD en grupos de cuatro bits y convierta cada grupo en decimal.

0110	1000	0011	1001
6	8	3	9

EJEMPLO 2-7

Convierta el número BCD 011111000001 en su equivalente decimal.

Solución

0111	1100	0001
7	↓	1

El grupo con el código prohibido indica un error en el número BCD

Comparación entre BCD y binario

Es importante entender que BCD no es otro sistema numérico como el binario, el decimal o el hexadecimal. De hecho, se utiliza el sistema decimal pero cada dígito está codificado en su equivalente binario. También es importante comprender que un número BCD *no* es lo mismo que un número binario directo. Un número binario directo toma el número decimal *completo* y lo representa en binario; el código BCD convierte *cada dígito* decimal en binario de manera individual. Para ilustrar esto, tome el número 137 y compare sus códigos binario directo y BCD:

$$137_{10} = 10001001_2 \quad \text{(binario)}$$

$$137_{10} = 0001\ \ 0011\ \ 0111 \quad \text{(BCD)}$$

Para representar el 137, el código BCD requiere 12 bits, mientras que el código binario directo sólo requiere de ocho bits. El código BCD requiere más bits que el binario directo para representar números decimales de más de un dígito, ya que no utiliza todos los grupos de cuatro bits posibles, como se indicó antes, y es, por lo tanto, algo ineficiente.

La principal ventaja del código BCD es la relativa facilidad de convertir BCD en decimal y viceversa. Sólo necesitan recordarse los grupos de código de cuatro bits para los dígitos decimales del 0 al 9. Esta facilidad de conversión es muy importante desde el punto de vista del hardware, ya que en un sistema digital son los circuitos lógicos los que realizan las conversiones hacia y desde decimal.

PREGUNTAS DE REPASO

1. Represente el valor decimal 178 mediante su equivalente binario directo. Luego codifique el mismo número decimal en BCD.
2. ¿Cuántos bits se requieren para representar un número decimal de ocho dígitos en BCD?
3. ¿Cuál es la ventaja de codificar un número decimal en BCD, en lugar de hacerlo en binario directo? ¿Cuál es la desventaja?

2-5 CÓDIGO GRAY

Los sistemas digitales operan a velocidades muy elevadas y responden a los cambios que se producen en las entradas digitales. Al igual que en la vida real, cuando varias condiciones de entrada están cambiando al mismo tiempo la situación puede malinterpretarse, con lo cual se puede llegar a producir una reacción errónea. Cuando se ven los bits en una secuencia de conteo binario, a menudo hay varios bits que deben cambiar estados al mismo tiempo. Por ejemplo, considere cuando el número binario de tres bits correspondiente al 3 decimal cambia a 4: los tres bits deben cambiar de estado.

Para reducir la probabilidad de que un circuito digital malinterprete una entrada cambiante, se desarrolló el **código Gray** como una manera de representar una secuencia de números. El aspecto único del código Gray es que, entre dos números sucesivos en la secuencia sólo un bit cambia. La tabla 2-2 muestra la traducción entre el valor del código binario de tres bits y el código Gray. Para convertir de binario a Gray sólo hay que empezar en el bit más significativo y usarlo como el MSB de Gray, como muestra la figura 2-2(a). Después se compara el MSB binario con el siguiente bit binario (B1). Si son iguales, entonces G1 = 0; si son distintos, entonces G1 = 1. Para encontrar G0 se compara B1 con B0.

TABLA 2-2
Equivalencia entre el código binario de tres bits y el código Gray.

B_2	B_1	B_0	G_2	G_1	G_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

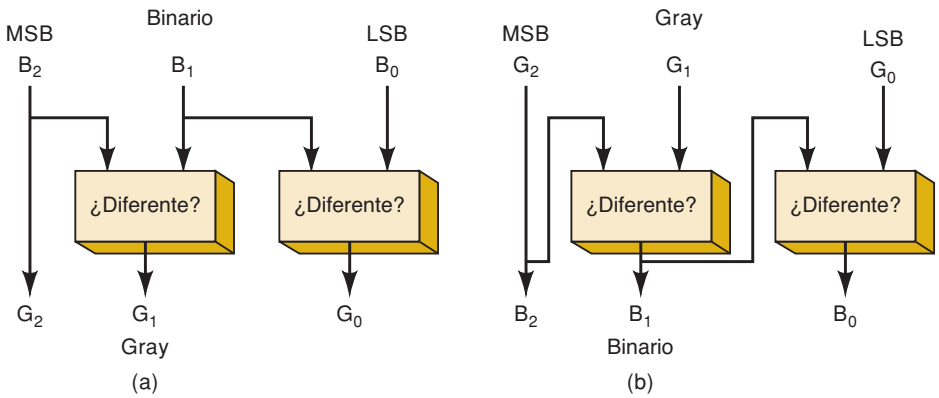
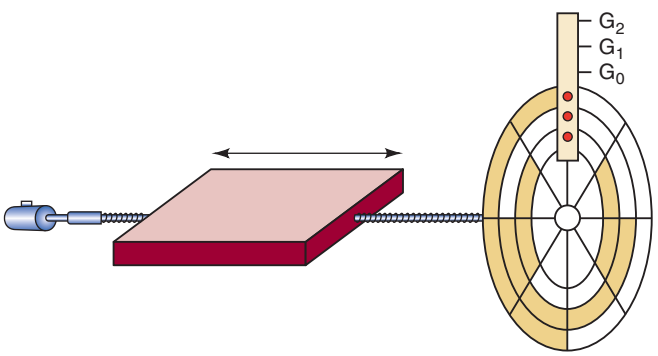


FIGURA 2-2 Conversión de (a) binario a Gray y de (b) Gray a binario.

La figura 2-2(b) muestra la conversión del código Gray a binario. Observe que el MSB en Gray siempre es el mismo que el MSB en binario. El siguiente bit binario se encuentra comparando el bit *binario* a la izquierda con el *bit correspondiente en código Gray*. Los bits similares producen un 0 y los bits distintos un 1. La aplicación más común del código Gray es en los codificadores de posición de eje, como muestra la figura 2-3. Estos dispositivos producen un valor binario que representa la posición de un eje mecánico giratorio. Un codificador de eje práctico utiliza mucho más de tres bits y divide la rotación en mucho más de ocho segmentos, por lo que puede detectar incrementos de rotación mucho más pequeños.

FIGURA 2-3 Un codificador de eje de ocho posiciones y tres bits.



PREGUNTAS DE REPASO

1. Convierta el número 0101 (binario) en su equivalente en código de Gray.
2. Convierta el número 0101 (código de Gray) en su número binario equivalente.

2-6 INTEGRACIÓN DE LOS SISTEMAS NUMÉRICOS

La tabla 2-3 muestra la representación de los números decimales del 1 al 15 en los sistemas numéricos binario y hexadecimal, y también en los códigos BCD y Gray. Examine esta tabla con cuidado y asegúrese de comprender de dónde proviene. Observe en especial cómo la representación en BCD siempre usa cuatro bits para cada dígito decimal.

TABLA 2-3

Decimal	Binario	Hexadecimal	BCD	GRAY
0	0	0	0000	0000
1	1	1	0001	0001
2	10	2	0010	0011
3	11	3	0011	0010
4	100	4	0100	0110
5	101	5	0101	0111
6	110	6	0110	0101
7	111	7	0111	0100
8	1000	8	1000	1100
9	1001	9	1001	1101
10	1010	A	0001 0000	1111
11	1011	B	0001 0001	1110
12	1100	C	0001 0010	1010
13	1101	D	0001 0011	1011
14	1110	E	0001 0100	1001
15	1111	F	0001 0101	1000

2-7 BYTE, NIBBLE Y PALABRA

Bytes

La mayoría de las microcomputadoras maneja y almacena datos binarios e información en grupos de ocho bits, por lo que una cadena de ocho bits tiene un nombre especial: **byte**. Un byte consiste de ocho bits y puede representar cualquier tipo de datos o de información. Los siguientes ejemplos ilustrarán este punto.

EJEMPLO 2-8

¿Cuántos bytes hay en una cadena de 32 bits?

Solución

32/8 = 4; por lo tanto, hay **cuatro** bytes en una cadena de 32 bits.

EJEMPLO 2-9

¿Cuál es el valor decimal más grande que puede representarse en binario si se utilizan dos bytes?

Solución

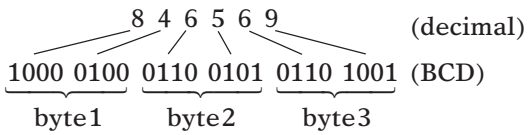
Dos bytes son 16 bits, por lo que el valor binario más grande será equivalente al número decimal $2^{16} - 1 = 65{,}535$.

EJEMPLO 2-10

¿Cuántos bytes se necesitan para representar el valor decimal 846,569 en BCD?

Solución

Cada dígito decimal se convierte en un código BCD de cuatro bits. Por ende, un número decimal de seis dígitos requiere 24 bits. Esos 24 bits corresponden a **tres** bytes. El diagrama para este caso se muestra a continuación:



Nibbles

A menudo los números binarios se descomponen en grupos de cuatro bits, como hemos visto con los códigos BCD y las conversiones a números hexadecimales. En los primeros días de los sistemas digitales surgió un término para describir un grupo de cuatro bits. Como abarca la mitad de un byte, se le denominó **nibble**. Los siguientes ejemplos ilustran el uso de este término.

EJEMPLO 2-11

¿Cuántos nibbles hay en un byte?

Solución

2

EJEMPLO 2-12

¿Cuál es el valor hexadecimal del nibble menos significativo del número binario 1001 0101?

Solución

1001 0101

El nibble menos significativo es 0101 = 5.

Palabras

Los términos bit, nibble y byte representan un número fijo de dígitos binarios. A medida que los sistemas han ido creciendo a través de los años, también ha crecido

su capacidad (¿apetito?) de manejar datos binarios. Una **palabra** es un grupo de bits que representa una cierta unidad de información. El tamaño de la palabra depende del tamaño de la ruta de datos en el sistema que utiliza la información. El **tamaño de palabra** puede definirse como el número de bits en la palabra binaria con el que opera un sistema digital. Por ejemplo, tal vez la computadora en su horno de microondas sólo pueda manejar un byte a la vez. Tiene un tamaño de palabra de ocho bits. Por otro lado, la computadora personal en su escritorio puede manejar ocho bytes a la vez, por lo que tiene un tamaño de palabra de 64 bits.

PREGUNTAS DE REPASO

1. ¿Cuántos bytes se necesitan para representar el número 235_{10} en binario?
2. ¿Cuál es el valor decimal más grande que puede representarse en BCD, si se utilizan dos bytes?
3. ¿Cuántos dígitos hexadecimales puede representar un nibble?
4. ¿Cuántos nibbles hay en un dígito BCD?

2-8 CÓDIGOS ALFANUMÉRICOS

Además de los datos numéricos, una computadora debe ser capaz de manejar información no numérica. En otras palabras, una computadora debe reconocer códigos que representen letras del alfabeto, signos de puntuación y otros caracteres especiales, además de los números. A estos códigos se les denomina **códigos alfanuméricos**. Un código alfanumérico completo debe incluir las 26 letras minúsculas, las 26 letras mayúsculas, los 10 dígitos numéricos, 7 signos de puntuación y de 20 a 40 caracteres adicionales, como +, /, #, %, *, y así sucesivamente. Podemos decir que un código alfanumérico representa a todos los caracteres y funciones que se encuentran en el teclado de una computadora.

Código ASCII

El código alfanumérico más utilizado es el **Código estándar estadounidense para el intercambio de información (ASCII)**. Este código es de siete bits, por lo cual tiene $2^7 = 128$ código posibles. Más que suficiente para representar todos los caracteres estándar del teclado, así como las funciones de control tales como retorno de carro (RETURN) y avance de línea (LINEFEED). La tabla 2-4 muestra un listado del código ASCII estándar de siete bits. La tabla proporciona los equivalentes en hexadecimal y decimal. Para obtener el código binario de siete bits para cada carácter hay que convertir el valor hexadecimal en binario.

EJEMPLO 2-13

Use la tabla 2-4 para encontrar el código ASCII de siete bits para el carácter de barra diagonal inversa (\).

Solución

El valor hexadecimal que aparece en la tabla 2-4 es 5C. Al traducir cada dígito hexadecimal en código binario de cuatro bits produce el valor 0101 1100. Los siete bits de menor peso representan el código ASCII para \, o 1011100.

TABLA 2-4 Código ASCII estándar.

Caracter	HEX	Decimal	Caracter	HEX	Decimal	Caracter	HEX	Decimal	Caracter	HEX	Decimal
NUL (nulo)	0	0	Espacio	20	32	@	40	64	`	60	96
Inicio del encabezado	1	1	!	21	33	A	41	65	a	61	97
Inicio del texto	2	2	“	22	34	B	42	66	b	62	98
Fin del texto	3	3	#	23	35	C	43	67	c	63	99
Fin de la transmisión	4	4	$	24	36	D	44	68	d	64	100
Consulta	5	5	%	25	37	E	45	69	e	65	101
Reconocimiento	6	6	&	26	38	F	46	70	f	66	102
Timbre	7	7	`	27	39	G	47	71	g	67	103
Retroceso	8	8	(	28	40	H	48	72	h	68	104
Tabulación horizontal	9	9	)	29	41	I	49	73	i	69	105
Avance de línea	A	10	*	2A	42	J	4A	74	j	6A	106
Tabulación vertical	B	11	+	2B	43	K	4B	75	k	6B	107
Avance de hoja de impresión	C	12	,	2C	44	L	4C	76	l	6C	108
Retorno de carro	D	13	-	2D	45	M	4D	77	m	6D	109
Mayúsculas desactivadas	E	14	.	2E	46	N	4E	78	n	6E	110
Mayúsculas activadas	F	15	/	2F	47	O	4F	79	o	6F	111
Escape de enlace de datos	10	16	0	30	48	P	50	80	p	70	112
Control directo 1	11	17	1	31	49	Q	51	81	q	71	113
Control directo 2	12	18	2	32	50	R	52	82	r	72	114
Control directo 3	13	19	3	33	51	S	53	83	s	73	115
Control directo 4	14	20	4	34	52	T	54	84	t	74	116
ACK (reconocimiento) negativo	15	21	5	35	53	U	55	85	u	75	117
Sincronía en estado inactivo	16	22	6	36	54	V	56	86	v	76	118
Fin de Bloque de Transmisión	17	23	7	37	55	W	57	87	w	77	119
Cancelar	18	24	8	38	56	X	58	88	x	78	120
Fin de medio	19	25	9	39	57	Y	59	89	y	79	121
Sustituir	1A	26	:	3A	58	Z	5A	90	z	7A	122
Escape	1B	27	;	3B	59	[	5B	91	{	7B	123
Separador de formas	1C	28	<	3C	60	\	5C	92	l	7C	124
Separador de grupos	1D	29	=	3D	61	]	5D	93	}	7D	125
Separador de registros	1E	30	>	3E	62	^	5E	94	~	7E	126
Separador de unidades	1F	31	?	3F	63		5F	95	Suprimir	7F	127

El código ASCII se utiliza para la transferencia de información alfanumérica entre una computadora y los dispositivos externos, tales como una impresora u otra computadora. La computadora también utiliza ASCII en forma interna para almacenar la información que escribe un operador en el teclado. El siguiente ejemplo ilustra lo anterior.

EJEMPLO 2-14

Un operador está escribiendo un programa en lenguaje C en el teclado de cierta microcomputadora, la cual convierte cada pulsación de tecla en su código ASCII, y lo almacena como un byte en memoria. Determine las cadenas binarias que se introducirán en memoria cuando el operador escriba la siguiente instrucción en C:

```
if (x>3)
```

Solución

Localice cada carácter (incluyendo el espacio) en la tabla 2-4 y escriba su código ASCII.

i	69	0110	1001
f	66	0110	0110
espacio	20	0010	0000
(	28	0010	1000
x	78	0111	1000
>	3E	0011	1110
3	33	0011	0011
)	29	0010	1001

Observe que se agregó un 0 al bit más a la izquierda de cada código ASCII, ya que los códigos deben almacenarse como bytes (ocho bits). A este proceso de agregar un bit adicional se le denomina *rellenar con 0s*.

PREGUNTAS DE REPASO

1. Codifique el siguiente mensaje en código ASCII utilizando la representación hexadecimal: "COSTO = $72."
2. El siguiente mensaje en código ASCII con bits de relleno se almacena en ubicaciones contiguas de memoria en una computadora:

   01010011 01010100 01001111 01010000

¿Cuál es el mensaje?

2-9 MÉTODO DE PARIDAD PARA LA DETECCIÓN DE ERRORES

El movimiento de datos binarios y códigos de un lugar a otro es la operación más frecuente que se realiza con los sistemas digitales. A continuación se listan solo unos cuantos ejemplos:

- La transmisión de voz digitalizada a través de un enlace de microondas.
- El almacenamiento de datos y la recuperación de los mismos desde dispositivos de memoria externos, como el disco magnético y el disco óptico.
- La transmisión de datos digitales desde una computadora hacia otra computadora remota, a través de líneas telefónicas (mediante el uso de un módem). Ésta es una de las principales formas de enviar y recibir información en Internet.

Siempre que se transmite información desde un dispositivo (el transmisor) hasta otro (el receptor), existe la posibilidad de que puedan producirse errores tales

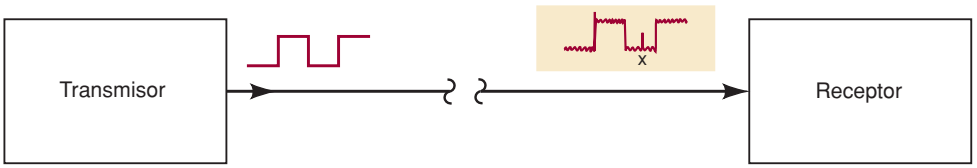


FIGURA 2-4 Ejemplo del ruido que produce un error en la transmisión de datos digitales.

que causen que el receptor no reciba la información idéntica tal y como la envió el transmisor. La principal causa de errores de transmisión es el *ruido eléctrico*, el cual consiste en fluctuaciones espurias en el voltaje o en la corriente, dichas señales de ruido están presentes en todos los sistemas electrónicos en diversos grados. La figura 2-4 es una ilustración simplificada de un tipo de error de transmisión.

El transmisor envía una señal digital en serie que está relativamente libre de ruido, a través de una línea de señal hacia un receptor. No obstante, para cuando la señal llega al receptor contiene un cierto grado de ruido acumulado a la señal original. En ocasiones, el ruido es tan grande en amplitud que altera el nivel lógico de la señal, como se ve en el punto *x*. Cuando esto ocurre, el receptor puede interpretar en forma incorrecta ese bit como un 1 lógico, que no corresponde con lo que el transmisor ha enviado.

La mayoría del equipo digital moderno está diseñado relativamente libre de errores, y la probabilidad de que ocurran errores como el que se muestra en la figura 2-4 es muy baja. No obstante, debemos considerar que los sistemas digitales frecuentemente transmiten miles, incluso millones de bits por segundo, por lo que incluso hasta una relación muy baja de ocurrencia de errores puede producir un error ocasional que podría llegar a ser molesto, si no es que desastroso. Por esta razón, muchos sistemas digitales emplean algún método para la detección (y en ocasiones corrección) de errores. Uno de los esquemas más simples y utilizados para este fin es el **método de paridad**.

Bit de paridad

Un **bit de paridad** es un bit que se agrega al grupo de bits del código que se está transfiriendo de un lugar a otro. El bit de paridad se hace 0 o 1, dependiendo del número de 1s que contenga el grupo de bits del código. Se utilizan dos métodos distintos.

En el método de *paridad par*, el valor del bit de paridad se elije de manera que el número total de 1s en el grupo de bits del código, incluyendo el bit de paridad, sea *par*. Por ejemplo, suponga que el grupo es 1000011. Éste es el carácter "C" en ASCII. El código tiene *tres* 1s. Por ende, agregaremos un bit de paridad de 1 para que el número total de 1s sea par. El *nuevo* código *incluyendo el bit de paridad* sería entonces:

1 1 0 0 0 0 1 1
↑
bit de paridad agregado*

Si el grupo de bits del código contiene un número par de 1s para empezar, el bit de paridad recibe un valor de 0. Por ejemplo, si el código fuera 1000001 (el código ASCII para la "A"), el bit de paridad asignado sería 0 y el nuevo código *incluyendo el bit de paridad* sería 01000001.

El método de *paridad impar* se utiliza de la misma forma, sólo que el bit de paridad se elije de manera que el número total de 1s, incluyendo el bit de paridad, sea *impar*. Por ejemplo, para el código 1000001, el bit de paridad asignado sería un 1. Para el código 1000011, el bit de paridad sería un 0.

* El bit de paridad puede colocarse en cualquier extremo del grupo de código, pero, por lo general, se coloca a la izquierda del MSB.

Ya sea que se utilice la paridad par o impar, el bit de paridad se convierte en parte de la palabra de código. Por ejemplo, al agregar un bit de paridad al código ASCII de siete bits se produce un código de ocho bits. Por lo tanto, el bit de paridad se trata justo igual que cualquier otro bit en el código.

El bit de paridad se transmite para detectar cualquier error de *un solo bit* que ocurra durante la transmisión de un código de un lugar a otro. Por ejemplo, suponga que el carácter "A" se va a transmitir y se va a utilizar paridad *impar*. El código transmitido sería

1 1 0 0 0 0 0 1

Cuando el circuito receptor reciba el código, verificará que éste contenga un número impar de 1s, incluyendo el bit de paridad. De ser así, el receptor supondrá que el código se ha recibido de manera correcta. Ahora suponga que debido a cierto ruido o falla el receptor en realidad recibe el siguiente código:

1 1 0 0 0 0 0 0

El receptor descubrirá que este código tiene un número *par* de 1s. Esto indica al receptor que debe haber un error en el código, ya que se presume que el transmisor y el receptor han acordado utilizar paridad impar. Sin embargo, no hay forma de que el receptor sepa cuál bit tiene error, ya que no sabe cuál se supone que va a ser el código.

Podemos asumir que este método de paridad no funciona si *dos* bits tienen error, ya que dos errores no cambiarían la característica de "par" o "impar" en el número de 1s en el código. En la práctica, el método de paridad se utiliza sólo en situaciones en las que la probabilidad de un solo error es muy baja y la probabilidad de doble error es prácticamente cero.

Cuando se utiliza el método de paridad, el transmisor y el receptor deben acordar antes de la transmisión si se va a utilizar la paridad par o impar. No hay ventaja de un método sobre el otro, aunque parece que la paridad par se utiliza más a menudo. El transmisor debe agregar el bit de paridad apropiado a cada unidad de información que transmita. Por ejemplo, si el transmisor está enviando datos codificados en ASCII, deberá agregar el bit de paridad a cada grupo de código ASCII de siete bits. Cuando el receptor examine los datos que reciba del transmisor, comprobará cada grupo de código para ver si el número total de 1s, incluyendo el bit de paridad, es consistente con el tipo de paridad acordado. A esto se le conoce comúnmente como *comprobar la paridad* de los datos. En caso de que detecte un error, el receptor puede enviar un mensaje al transmisor para pedirle que vuelva a transmitir el último conjunto de datos. El procedimiento a seguir cuando se detecta un error depende de cada sistema.

EJEMPLO 2-15

Es común que las computadoras se comuniquen con otros equipos remotos a través de líneas telefónicas. Así es como se lleva a cabo la comunicación por acceso telefónico a través de Internet. Cuando una computadora transmite un mensaje a otra, esa información, por lo general, se codifica en ASCII. ¿Cuáles serían las cadenas de bits que transmitiría una computadora para enviar el mensaje HOLA, utilizando ASCII con paridad par?

Solución

Primero, vea los códigos ASCII para cada carácter del mensaje. Después, para cada código cuente el número de 1s. Si es un número par añada un 0 como el MSB, o un 1

si es un número impar. Por ende, los códigos de ocho bits (bytes) resultantes tendrán un número par de 1s (incluyendo la paridad).

		bits de paridad agregados ↓							
H	=	0	1	0	0	1	0	0	0
O	=	1	1	0	0	0	1	0	1
L	=	1	1	0	0	1	1	0	0
A	=	0	1	0	0	0	0	0	1

PREGUNTAS DE REPASO

1. Agregue un bit de paridad impar al código ASCII para el símbolo $ y exprese el resultado en hexadecimal.
2. Adjunte un bit de paridad par al código BCD para el 69 decimal.
3. ¿Por qué el método de paridad no puede detectar un doble error en los datos transmitidos?

2-10 APLICACIONES

A continuación se presentan algunas aplicaciones que servirán como repaso de algunos conceptos que vistos en este capítulo. El objetivo de estas aplicaciones es que usted se dé una idea de cómo se utilizan los diversos sistemas y códigos numéricos en el mundo digital. En los problemas al final del capítulo presentaremos más aplicaciones.

APLICACIÓN 2-1

Un CD-ROM ordinario puede almacenar 650 megabytes de datos digitales. Como mega = 2^{20}, ¿cuántos bits de datos puede almacenar un CD-ROM?

Solución

Recuerde que un byte tiene ocho bits. Por lo tanto, 650 megabytes son $650 \times 2^{20} \times 8 =$ **5,452,595,200 bits**.

APLICACIÓN 2-2

Para poder programar la mayoría de microcontroladores, las instrucciones binarias se almacenan en un archivo de una computadora personal, de una manera especial que se conoce como Formato Hexadecimal Intel. La información hexadecimal se codifica en caracteres ASCII, de manera que pueda mostrarse con facilidad en la pantalla de una PC, imprimirse o transmitirse de una manera sencilla, carácter por carácter, a través de un puerto COM serial de la PC. A continuación se muestra una línea de un archivo en Formato Hexadecimal Intel:

:10002000F7CFFFCF1FEF2FEF2A95F1F71A95D9F7EA

El primer carácter enviado es el código ASCII correspondiente a los dos puntos, seguido de un 1. A cada uno se le adjunta un bit de paridad como el bit más significativo. Un instrumento de prueba captura el patrón binario de bits a medida que pasa a través del cable hacia el microcontrolador.

(a) ¿Qué apariencia debe tener el patrón binario de bits incluyendo la paridad? (MSB-LSB.)

(b) El valor 10, que va después de los dos puntos, representa en número hexadecimal el total de bytes que se van a cargar en la memoria del microcontrolador. ¿Cuál es el número decimal de bytes que se van a cargar?

(c) El número 0020 es un valor hexadecimal de cuatro dígitos que representa la dirección en la que se va a almacenar el primer byte. ¿Cuál es la mayor dirección posible? ¿Cuántos bits se requerirían para representar esta dirección?

(d) El valor del primer byte de datos es F7. ¿Cuál es el valor (en binario) del nibble menos significativo de este byte?

FFFF 1111 1111 1111 1111 16 bits

Solución

(a) Los códigos ASCII son 3A (para el :) y 31 (para el 1) 00111010 10110001
bit de paridad par

(b) 10 hex = $1 \times 16 + 0 \times 1 = 16$ bytes decimales.

(c) FFFF es el mayor valor posible. Cada dígito hexadecimal es de 4 bits, por lo que necesitamos 16 bits.

(d) El nibble menos significativo (4 bits) se representa mediante el 7 hexadecimal. En binario sería 0111.

APLICACIÓN 2-3

Una pequeña computadora de control de procesos utiliza códigos hexadecimales para representar sus direcciones de 16 bits de memoria.

(a) ¿Cuántos dígitos hexadecimales se requieren?

(b) ¿Cuál es el intervalo de direcciones en hexadecimal?

(c) ¿Cuántas localidades de memoria hay?

Solución

(a) Como 4 bits se convierten en un dígito hexadecimal, se necesitan 16/4 = 4 dígitos hexadecimales.

(b) El intervalo binario es de 0000000000000000_2 a 1111111111111111_2. En hexadecimal sería de 0000_{16} a $FFFF_{16}$.

(c) Con 4 dígitos hexadecimales, el número total de direcciones es $16^4 = \mathbf{65{,}536}$.

APLICACIÓN 2-4

En un sistema basado en microcontrolador, los números se introducen en BCD pero se almacenan en binario directo. Como programador, usted debe decidir si necesita una ubicación de almacenamiento de un byte o de dos bytes.

(a) ¿Cuántos bytes necesita si el sistema recibe una entrada decimal de dos dígitos?

(b) ¿Qué pasaría si tuviera que introducir tres dígitos?

Solución

(a) Con dos dígitos puede introducir valores hasta el 99 ($1001\ 1001_{BCD}$). En binario este valor es 01100011, el cual cabe en una ubicación de memoria de ocho bits. También puede utilizar un solo bit.

(b) Tres dígitos pueden representar hasta 999 (1001 1001 1001). En binario este valor es 1111100111 (10 bits). Por lo tanto, no puede usar un solo byte; necesita dos bytes.

APLICACIÓN 2-5

Cuando hay que transmitir caracteres ASCII entre dos sistemas independientes (como una computadora y un módem), debe haber una manera de indicar al receptor cuándo va a llegar un nuevo carácter. A menudo también se tiene la necesidad de detectar errores en la transmisión. El método de transferencia se llama comunicación asíncrona de datos. El estado normal de inactividad de la línea de transmisión es un 1 lógico. Cuando el transmisor envía un carácter ASCII, debe "encapsularse" para que el receptor sepa en dónde comienzan y terminan los datos. El primer bit debe ser siempre un bit de inicio (0 lógico). A continuación se envía el código ASCII, en donde el LSB va primero y el MSB al último. Después del MSB se adjunta un bit de paridad para comprobar errores en la transmisión. Para terminar la transmisión se envía un bit de paro (1 lógico). En la figura 2-5 se muestra una transmisión asíncrona ordinaria del código ASCII de siete bits para el signo # (23 Hex) con paridad par.

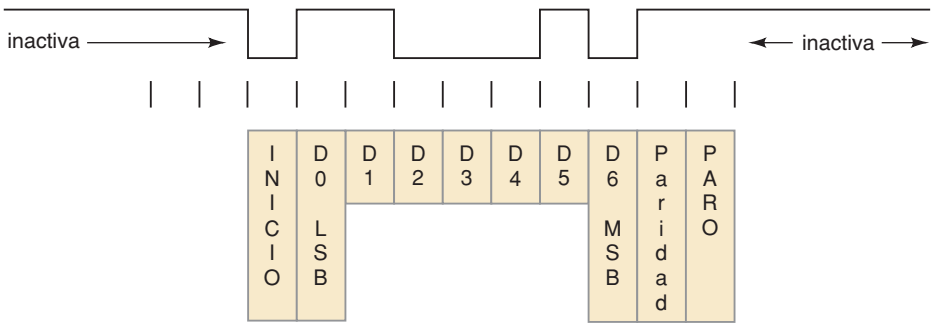


FIGURA 2-5 Datos seriales asíncronos con paridad par.

RESUMEN

1. El sistema numérico hexadecimal se utiliza en los sistemas y las computadoras digitales como una manera eficiente de representar cantidades binarias.
2. En las conversiones entre hexadecimal y binario, cada dígito hexadecimal corresponde a cuatro bits.
3. El método de división repetida se utiliza para convertir números decimales en binario o hexadecimal.
4. Mediante el uso de un número binario de N bits podemos representar valores decimales desde el 0 hasta $2^N - 1$.
5. Para formar el código BCD de un número decimal se convierte cada dígito del número decimal en su equivalente binario de cuatro bits.
6. El código Gray define una secuencia de patrones de bits, en los que sólo un bit cambia entre patrones sucesivos en la secuencia.
7. Un byte es una cadena de ocho bits. Un nibble es de cuatro bits. El tamaño de una palabra depende del sistema.
8. Un código alfanumérico utiliza grupos de bits para representar a todos los caracteres y funciones que forman parte del teclado ordinario de una computadora. El código ASCII es el código alfanumérico más utilizado.
9. En el método de paridad para la detección de errores se adjunta un bit de paridad especial a cada grupo de bits que se transmite.

TÉRMINOS IMPORTANTES

bit de paridad
byte
códigos alfanuméricos
código binario directo
código decimal codificado en binario (BCD)
Código estándar estadounidense para el intercambio de información (ASCII)
código Gray
método de paridad
nibble
palabra
sistema numérico hexadecimal
tamaño de palabra

PROBLEMAS

SECCIONES 2-1 Y 2-2

2-1. Convierta los siguientes números binarios en decimales.

(a)* 10110 (d) 01101011 (g)* 1111010111
(b) 10010101 (e)* 11111111 (h) 11011111
(c)* 100100001001 (f) 01101111

2-2. Convierta los siguientes valores decimales en binarios.

(a)* 37 (d) 1000 (g)* 205
(b) 13 (e)* 77 (h) 2133
(c)* 189 (f) 390 (i)* 511

2-3. ¿Cuál es el valor decimal más grande que puede representarse mediante (a)* un número binario de ocho bits? (b) un número de 16 bits?

SECCIÓN 2-4

2-4. Convierta cada número hexadecimal en su equivalente decimal.

(a)* 743 (d) 2000 (g)* 7FF
(b) 36 (e)* 165 (h) 1204
(c)* 37FD (f) ABCD

2-5. Convierta cada uno de los siguientes números decimales en hexadecimales.

(a)* 59 (d) 1024 (g)* 65,536
(b) 372 (e)* 771 (h) 255
(c)* 919 (f) 2313

2-6. Convierta cada uno de los valores hexadecimales del problema 2-4 en números binarios.

2-7. Convierta los números binarios del problema 2-1 en hexadecimales.

2-8. Liste los números hexadecimales en secuencia, desde 195_{16} hasta 280_{16}.

2-9. Cuando se va a convertir un número decimal grande en binario, algunas veces es más fácil convertirlo primero en hexadecimal y después en binario. Pruebe este procedimiento para el número 2133_{10} y compárelo con el procedimiento usado en el problema 2-2(h).

2-10. ¿Cuántos dígitos hexadecimales se requieren para representar los números decimales del 0 hasta el 20,000?

2-11. Convierta los siguientes valores hexadecimales en decimales.

(a)* 92 (d) ABCD (g)* 2C0
(b) 1A6 (e)* 000F (h) 7FF
(c)* 37FD (f) 55

* Encontrará las respuestas a los problemas marcados con un asterisco al final del libro.

2-12. Convierta los siguientes valores decimales en hexadecimales.

(a)* 75	(d) 24	(g)* 25,619
(b) 314	(e)* 7245	(h) 4095
(c)* 2048	(f) 498	

2-13. Tome cada número binario de cuatro bits en el orden en el que están escritos y escriba el dígito hexadecimal equivalente sin realizar ningún cálculo manual o mediante la calculadora.

(a) 1001	(e) 1111	(i) 1011	(m) 0001
(b) 1101	(f) 0010	(j) 1100	(n) 0101
(c) 1000	(g) 1010	(k) 0011	(o) 0111
(d) 0000	(h) 1001	(l) 0100	(p) 0110

2-14. Tome cada dígito hexadecimal y escriba su valor binario de cuatro bits sin realizar ningún cálculo manual ni mediante la calculadora.

(a) 6	(e) 4	(i) 9	(m) 0
(b) 7	(f) 3	(j) A	(n) 8
(c) 5	(g) C	(k) 2	(o) D
(d) 1	(h) B	(l) F	(p) 9

2-15.* Convierta los números binarios del problema 2-1 en hexadecimales.

2-16.* Convierta los valores hexadecimales del problema 2-11 en binarios.

2-17.* Liste los números hexadecimales en secuencia, desde 280 hasta 2A0.

2-18. ¿Cuántos dígitos hexadecimales se requieren para representar números decimales hasta 1 millón?

SECCIÓN 2-5

2-19. Codifique los siguientes números decimales en BCD.

(a)* 47	(d) 6727	(g)* 89,627
(b) 962	(e)* 13	(h) 1024
(c)* 187	(f) 529	

2-20. ¿Cuántos bits se requieren para representar los números decimales en el intervalo de 0 a 999 si se utiliza: (a) código binario directo, y (b) código BCD?

2-21. Los siguientes números están en BCD. Conviértalos en decimales.

(a)* 1001011101010010	(d) 0111011101110101
(b) 000110000100	(e)* 010010010010
(c)* 011010010101	(f) 010101010101

SECCIÓN 2-7

2-22.* (a) ¿Cuántos bits hay en ocho bytes?

(b) ¿Cuál es el número hexadecimal más grande que puede representarse en cuatro bytes?

(c) ¿Cuál es el valor decimal codificado en BCD más grande que puede representarse en tres bytes?

2-23. (a) Consulte la tabla 2-4. ¿Cuál es el nibble más significativo del código ASCII para la letra X?

(b) ¿Cuántos nibbles pueden almacenarse en una palabra de 16 bits?

(c) ¿Cuántos bytes se requieren para formar una palabra de 24 bits?

SECCIONES 2-8 Y 2-9

2-24. Represente la instrucción "X = 3 × Y" en código ASCII. Adjunte un bit de paridad impar.

2-25.* Adjunte un bit de paridad *par* a cada uno de los códigos ASCII del problema 2-24, y muestre los resultados en hexadecimal.

2-26. Los siguientes bytes (mostrados en hexadecimal) representan el nombre de una persona según como se almacenaría en la memoria de una computadora. Cada byte es código ASCII con relleno. Determine el nombre de cada persona.

(a)* 42 45 4E 20 53 4D 49 54 48

(b) 4A 6F 65 20 47 72 65 65 6E

2-27. Convierta los siguientes números decimales en código BCD y después adjunte un bit de paridad *impar*.

(a)* 74 (c)* 8884 (e)* 165

(b) 38 (d) 275 (f) 9201

2-28.* En cierto sistema digital, los números decimales del 000 al 999 se representan en código BCD. También se incluye un bit de paridad *impar* al final de cada grupo. Examine cada uno de los códigos que se muestran a continuación y suponga que cada uno acaba de transferirse de un lugar a otro. Algunos de los grupos contienen errores. Suponga que *no se han producido más de dos errores* en cada grupo. Determine cuáles de los siguientes casos tienen un solo error y cuáles tienen *en definitiva* un error doble. (*Sugerencia:* recuerde que éste es código BCD).

(a) 1001010110000

(b) 0100011101100

(c) 0111110000011

(d) 1000011000101

2-29. Suponga que el receptor recibió los siguientes datos del transmisor del ejemplo 2-16:

0 1 0 0 1 0 0 0
1 1 0 0 0 1 0 1
1 1 0 0 1 1 0 0
1 1 0 0 1 0 0 0
1 1 0 0 1 1 0 0

¿Qué errores puede determinar el receptor en los datos que recibió?

PREGUNTAS DE PRÁCTICA

2-30.* Realice cada una de las siguientes conversiones. Si desea puede probar varios métodos en algunas de ellas para ver con cuál se adapta mejor. Por ejemplo, una conversión de binario a decimal puede realizarse en forma directa, o también mediante una conversión de binario a hexadecimal seguida de una conversión de hexadecimal a decimal.

(a) 1417_{10} = _____$_2$

(b) 255_{10} = _____$_2$

(c) 11010001_2 = _____$_{10}$

(d) 1110101000100111_2 = _____$_{10}$

(e) 2497_{10} = _____$_{16}$
(f) 511_{10} = _____ (BCD)
(g) 235_{16} = _____$_{10}$
(h) 4316_{10} = _____$_{16}$
(i) $7A9_{16}$ = _____$_{10}$
(j) $3E1C_{16}$ = _____$_{10}$
(k) 1600_{10} = _____$_{16}$
(l) $38{,}187_{10}$ = _____$_{16}$
(m) 865_{10} = _____ (BCD)
(n) 100101000111 (BCD) = _____$_{10}$
(o) 465_{16} = _____$_{2}$
(p) $B34_{16}$ = _____$_{2}$
(q) 01110100 (BCD) = _____$_{2}$
(r) 111010_2 = _____ (BCD)

2-31.* Represente el valor decimal 37 en cada una de las siguientes formas.
(a) Binario directo.
(b) BCD.
(c) Hexadecimal.
(d) ASCII (es decir, trate cada dígito como un carácter).

2-32.* Llene los espacios en blanco con la palabra o palabras correctas.
(a) Para convertir de decimal a _____ se requiere de la división repetida entre 16.
(b) Para convertir de decimal a binario se requiere de la división repetida entre _____.
(c) En el código BCD, cada _____ se convierte en su equivalente binario de cuatro bits.
(d) El código _____ tiene la característica de que sólo cambia un bit al avanzar de un paso al siguiente.
(e) Un transmisor adjunta un _____ a un código para permitir que el receptor detecte _____.
(f) El código _____ es código alfanumérico más común que se utiliza en los sistemas computacionales.
(g) _____ se utiliza a menudo como una manera conveniente de representar números binarios extensos.
(h) Una cadena de ocho bits se llama _____.

2-33. Escriba el número binario que se produce cuando cada uno de los siguientes números se incrementa en uno.
(a)* 0111 (b) 010011 (c) 1011

2-34. Decremente cada uno de los siguientes números binarios.
(a)* 1110 (b) 101000 (c) 1110

2-35. Escriba el número que se produce cuando se incrementa cada una de las siguientes cifras.
(a)* 7779_{16} (c)* $0FFF_{16}$ (e)* $9FF_{16}$
(b) 9999_{16} (d) 2000_{16} (f) $100A_{16}$

2-36.* Repita el problema 2-35 para la operación de decremento.

EJERCICIOS AVANZADOS

2-37.* En una microcomputadora, las *direcciones* de las localidades de memoria son números binarios que identifican cada uno de los circuitos de memoria en donde se almacena un byte. El número de bits que forman cada dirección depende de cuántas localidades de memoria haya. Como el número de bits puede ser muy extenso, a menudo las direcciones se especifican en hexadecimal, en lugar de binario.

(a) Si una microcomputadora utiliza una dirección de 20 bits, ¿cuántas localidades de memoria distintas hay?

(b) ¿Cuántos dígitos hexadecimales se necesitan para representar la dirección de una localidad de memoria?

(c) ¿Cuál es la dirección hexadecimal de la localidad de memoria número 256? (*Nota:* la primera dirección siempre es 0.)

2-38. En un CD de audio, la señal de voltaje de audio, por lo general, se muestrea aproximadamente 44,000 veces por segundo, y el valor de cada muestra se graba en la superficie del CD como número binario. En otras palabras, cada número binario que se graba representa un punto de voltaje individual en la forma de onda de la señal de audio.

(a) Si los números binarios tienen una longitud de seis bits, ¿cuántos valores de voltaje distintos pueden representarse mediante un solo número binario? Repita para ocho y diez bits.

(b) Si se utilizan números de diez bits, ¿cuántos bits se grabarán en el CD en un segundo?

(c) Si un CD puede almacenar, por lo general, 5 mil millones de bits, cuántos segundos de audio pueden grabarse si se utilizan diez bits?

2-39.* Una cámara digital en blanco y negro coloca una rejilla fina sobre una imagen para después medir y registrar un número binario que representa el nivel de gris que ve en cada celda de la rejilla. Por ejemplo, si se utilizan números de cuatro bits el valor de negro se establece en 0000 y el valor de blanco en 1111, y cualquier nivel de gris puede tener algún valor entre 0000 y 1111. Si se utilizan números de seis bits, el negro es 000000 y el blanco es 111111, y todos los grises se encuentran entre estos dos valores.

Suponga que queremos diferenciar entre 254 niveles de gris dentro de cada una de las celdas de la rejilla. ¿Cuántos bits necesitaríamos usar para representar estos niveles de gris?

2-40. Una cámara digital de 3 megapixeles almacena un número de ocho bits para el brillo de cada uno de los colores primarios (rojo, verde, azul) que se encuentran en cada elemento de imagen (pixel). Si se almacenan todos los bits, sin compresión de datos, ¿cuántas imágenes pueden almacenarse en una tarjeta de memoria de 128 Megabytes? (Nota: en los sistemas digitales, Mega significa 2^{20}).

2-41. Construya una tabla que muestre las representaciones en binario, hexadecimal y BCD de todos los números decimales del 0 al 15. Compare sus resultados con la tabla 2-3.

RESPUESTAS A LAS PREGUNTAS DE REPASO DE LAS SECCIONES

SECCIÓN 2-1

1. 2267 2. 32768

SECCIÓN 2-2

1. 1010011 2. 1011011001 3. 20 bits