2.1 TIPOS DE DATOS

En la actualidad los datos se presentan de diferentes maneras, por ejemplo números, texto, imágenes, audio y video (figura 2.1). La gente necesita procesar todos estos tipos de datos.

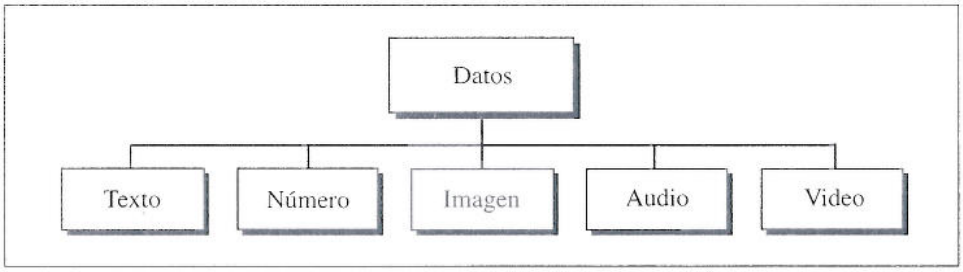

Figura 2.1 Diferentes tipos de datos

- Un programa de ingeniería utiliza una computadora principalmente para procesar números: hacer aritmética, resolver ecuaciones algebraicas o trigonométricas, encontrar las raíces de una ecuación diferencial, y así por el estilo.
- Un programa de procesamiento de palabras, por otra parte, utiliza una computadora más que nada para procesar texto: justificarlo, moverlo, eliminarlo, etcétera.
- Un programa de procesamiento de imágenes usa una computadora para manipular imágenes: crearlas, reducirlas, ampliarlas, rotarlas, etcétera.
- Una computadora también puede manejar datos de audio. Usted puede reproducir música en una computadora e introducir su voz como datos.
- Finalmente, una computadora puede usarse no sólo para mostrar películas, sino también para crear los efectos especiales que se ven en ellas.

La industria de la computación usa el término *multimedia* para definir información que contiene números, texto, imágenes, audio y video.

2.2 DATOS DENTRO DE LA COMPUTADORA

La pregunta es: ¿Cómo se manejan todos estos tipos de datos? ¿Se necesitan otras computadoras para procesar los distintos tipos de datos? Es decir, ¿se tiene una categoría de computadoras que procesan sólo números? ¿Hay una categoría de computadoras que procesan sólo texto?

Esta solución de diferentes computadoras para procesar distintos tipos de datos no es económica ni práctica porque los datos por lo general son una mezcla de tipos. Por ejemplo, aunque un banco procesa principalmente números, también necesita almacenar, como texto, los nombres de sus clientes. Como otro ejemplo, una imagen con frecuencia es una mezcla de gráficos y texto.

La solución más eficaz es usar una representación uniforme de los datos. Todo tipo de datos que entran del exterior a una computadora se transforman en esta representación uniforme cuando se almacenan en una computadora y se vuelven a transformar en su representación original cuando salen de la computadora. Este formato universal se llama *patrón de bits*.

BIT

Antes de continuar con el análisis de los patrones de bits, se debe definir un bit. Un **bit** (*binary digit:* **dígito binario**) es la unidad más pequeña de datos que puede almacenarse en una computadora; puede ser ya sea 0 o 1. Un bit representa el estado de un dispositivo que puede tomar uno de dos estados. Por ejemplo, un **interruptor** puede estar ya sea apagado o encendido. La convención es representar el estado de encendido como 1 y el estado de apagado como 0. Un interruptor electrónico puede representar un bit. En otras palabras, un interruptor puede almacenar un bit de información. Actualmente las computadoras utilizan varios dispositivos binarios de dos estados para almacenar datos.

PATRÓN DE BITS

Un solo bit no puede resolver el problema de la representación de datos, si cada pieza de datos pudiera representarse por un 1 o un 0, entonces sólo se necesitaría un bit. Sin embargo, usted necesita almacenar números más grandes, necesita almacenar texto, gráficos y otros tipos de datos.

Para representar diferentes tipos de datos se utiliza un **patrón de bits**, una secuencia o, como a veces se le llama, una cadena de bits. La figura 2.2 muestra un patrón de bits formado por 16 bits, es una combinación de ceros (0) y unos (1). Esto significa que si usted quiere almacenar un patrón de bits formado por 16 bits, necesita 16 interruptores electrónicos. Si quiere almacenar 1 000 patrones de bits, cada uno de 16 bits, necesita 16 000 bits y así sucesivamente.

```
1000101010111111
```

Figura 2.2 Patrón de bits

Ahora la pregunta es: ¿Cómo sabe la memoria de la computadora qué tipo de datos representa el patrón de bits? No lo sabe. La memoria de la computadora sólo almacena los datos como patrones de bits. Es responsabilidad de los dispositivos de entrada/salida o de los programas interpretar un patrón de bits como un número, texto o algún otro tipo de datos. En otras palabras, los datos se codifican cuando entran a la computadora y se decodifican cuando se presentan al usuario (figura 2.3).

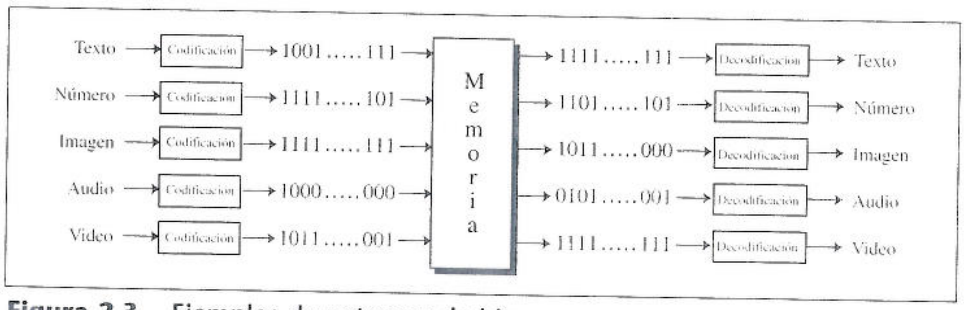

Figura 2.3 Ejemplos de patrones de bits

BYTE

Por tradición, un patrón de bits con una longitud de 8 bits se llama **byte**. Este término también se ha utilizado para medir el tamaño de la memoria o de otros dispositivos de almacenamiento. Por ejemplo, se dice que la memoria de una computadora que puede almacenar 8 millones de bits de información es una memoria de 1 millón de bytes.

2.3 REPRESENTACIÓN DE DATOS

Ahora podemos explicar cómo pueden representarse diferentes tipos de datos usando patrones de bits.

TEXTO

Una pieza de **texto** en cualquier idioma es una secuencia de símbolos usados para representar una idea en ese idioma. Por ejemplo, el idioma inglés utiliza 26 símbolos (A, B, C, . . . , Z) para representar las letras mayúsculas, 26 símbolos (a, b, c, . . . , z) para representar las letras minúsculas, 9 símbolos (0, 1, 2, . . . , 9) para los caracteres numéricos (no números; la diferencia se verá más adelante) y símbolos (., ?, :, ; . . . , !) para representar la puntuación. Otros símbolos como el espacio en blanco, la línea nueva y el tabulador se usan para alineación de texto y legibilidad.

Usted puede representar cada símbolo con un patrón de bits. Dicho de otra forma, texto como la palabra "BYTE", formada por cuatro símbolos, puede representarse como 4 patrones de bits, en los que cada patrón define un solo símbolo (figura 2.4).

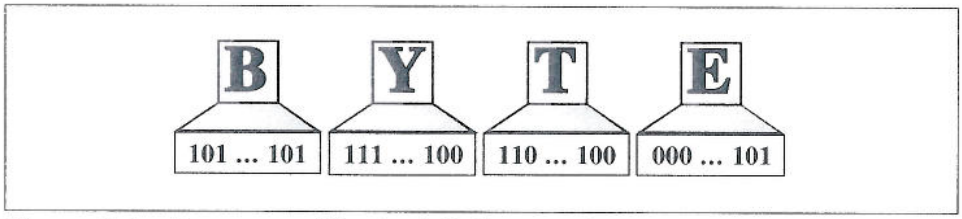

Figura 2.4 Representación de símbolos usando patrones de bits

La pregunta es: ¿Cuántos bits se necesitan en un patrón de bits para representar un símbolo en un idioma? Depende de cuántos símbolos haya en la secuencia. Por ejemplo, si usted crea un idioma imaginario que utilice sólo las letras mayúsculas del idioma inglés, sólo necesita 26 símbolos. Un patrón de bits en este idioma requiere representar al menos 26 símbolos. Para otro idioma, como el chino, pueden necesitarse muchos símbolos más. La longitud del patrón de bits que representa un símbolo en un idioma depende del número de símbolos usados en ese idioma. Más símbolos significan un patrón de bits más grande.

Aunque la longitud del patrón de bits depende del número de símbolos, la relación no es lineal; es logarítmica. Si se requieren dos símbolos, la longitud es 1 bit (el $\log_2 2$ es 1). Si se necesitan cuatro símbolos, la longitud es 2 bits ($\log_2 4$ es 2). La tabla 2.1 muestra esta relación, la cual es fácilmente perceptible. Un patrón de bits de 2 bits puede tomar cuatro formas diferentes: 00, 01, 10 y 11; cada una de las cuales representa un símbolo. Del mismo modo, un patrón de tres bits puede tomar ocho formas diferentes: 000, 001, 010, 011, 100, 101, 110 y 111.

Número de símbolos	Longitud del patrón de bits
2	1
4	2
8	3
16	4
. . .	. . .
128	7
256	8
. . .	. . .
65 536	16

Tabla 2.1 Número de símbolos y longitud de un patrón de bits

Códigos

Se han diseñado diferentes secuencias de patrones de bits para representar símbolos de texto. A cada secuencia se le conoce como **código** y al proceso de representar los símbolos se le llama codificación. En esta sección explicamos los códigos comunes.

ASCII El **Instituto Nacional Norteamericano de Estándares** (**ANSI**: *American Nacional Standards Institute*) desarrolló un código llamado **Código norteamericano de estándares para intercambio de información** (**ASCII**: *American Standard Code for Information Interchange*). Este código utiliza siete bits para cada símbolo. Esto significa que 128 (2^7) símbolos distintos pueden definirse mediante este código. Los patrones de bits completos para el código ASCII están en el apéndice A. La figura 2.5 muestra cómo se representa la palabra "BYTE" en código ASCII.

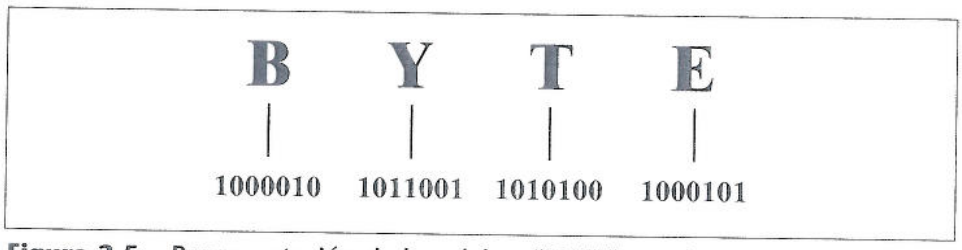

Figura 2.5 Representación de la palabra "BYTE" en código ASCII

La lista siguiente destaca algunas de las características de este código:

- ASCII utiliza un patrón de siete bits que varía de 0000000 a 1111111.
- El primer patrón (0000000) representa el carácter nulo (la ausencia de carácter).
- El último patrón (1111111) representa el carácter de eliminación.
- Hay 31 caracteres de control (no imprimibles).
- Los caracteres numéricos (0 a 9) se codifican antes que las letras.
- Hay varios caracteres de impresión especiales.
- Las letras mayúsculas (A ... Z) están antes que las letras minúsculas (a ... z).
- Los caracteres en mayúsculas y en minúsculas se distinguen sólo por un bit. Por ejemplo, el patrón para A es 1000001; el partón para a es 1100001. La única diferencia es el sexto bit a partir de la derecha.
- Hay seis caracteres especiales entre las letras mayúsculas y minúsculas.

ASCII extendido Para hacer que el tamaño de cada patrón sea de 1 byte (8 bits), a los patrones de bits ASCII se les aumenta un 0 más a la izquierda. Ahora cada patrón puede caber fácilmente en un byte de memoria. En otras palabras, en **ASCII extendido** el primer patrón es 00000000 y el último es 01111111.

Algunos fabricantes han decidido usar el bit de más para crear un sistema de 128 símbolos adicional. Sin embargo, este intento no ha tenido éxito debido a la secuencia no estándar creada por cada fabricante.

EBCDIC A principios de la era de las computadoras, IBM desarrolló un código llamado **Código extendido de intercambio decimal codificado en binario** (**EBCDIC**: *Extended Binary Coded Decimal Interchange Code*). Este código utiliza patrones de ocho bits, de manera que puede representar hasta 256 símbolos. Sin embargo, este código no se utiliza más que en computadoras mainframe de IBM.

Unicode Ninguno de los códigos anteriores representa símbolos que pertenecen a idiomas distintos al inglés. Por eso, se requiere un código con mucha más capacidad. Una coalición de fabricantes de hardware y software ha diseñado un código llamado **Unicode** que utiliza 16 bits y puede representar hasta 65 536 (2^{16}) símbolos. Diferentes secciones del código se asignan a los símbolos de distintos idiomas en el mundo. Algunas partes del código se usan para símbolos gráficos y especiales. El lenguaje Java™ utiliza este código para representar caracteres. Microsoft Windows usa una variación de los primeros 256 caracteres. En el apéndice B hay un pequeño conjunto de símbolos Unicode.

ISO La **Organización Internacional para la Estandarización** (*Internacional Standard Organization*), conocida como ISO, ha diseñado un código que utiliza patrones de 32 bits. Este código representa hasta 4 294 967 296 (2^{32}) símbolos, definitivamente lo suficiente para representar cualquier símbolo en el mundo actual.

NÚMEROS

En una computadora, los números se representan usando el **sistema binario.** En este sistema, un patrón de bits (una secuencia de ceros y unos) representa un número. Sin embargo, un código como el ASCII no se usa para representar datos. La razón para ello y un análisis de la representación de números se presentan en el capítulo 3.

IMÁGENES

Hoy día las **imágenes** se representan en una computadora mediante uno de dos métodos: **gráficos de mapa de bits** o **gráficos de vectores** (figura 2.6).

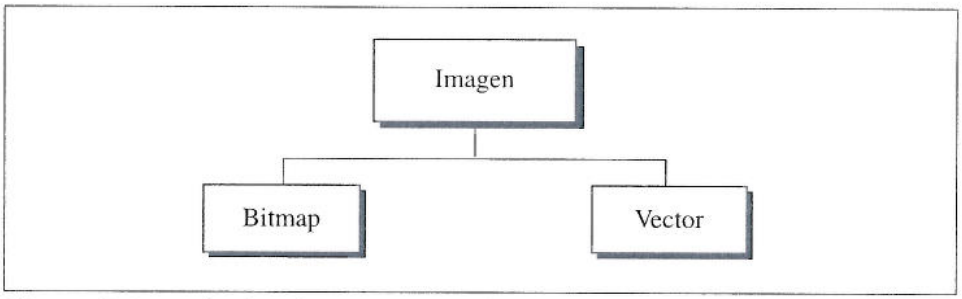

Figura 2.6 Métodos de representación de imágenes

Gráficos de mapa de bits

En este método, una imagen se divide en una matriz de **pixeles** (*picture elements:* **elementos de imagen**), donde cada pixel es un pequeño punto. El tamaño del pixel depende de lo que se conoce como *resolución*. Por ejemplo, una imagen puede dividirse en 1000 pixeles o 10 000 pixeles. En el segundo caso, aunque hay una mejor representación de la imagen (mejor resolución), se necesita más memoria para almacenarla.

Después de dividir una imagen en pixeles, a cada pixel se asigna un patrón de bits. El tamaño y el valor del patrón dependen de la imagen. Para una imagen formada sólo por puntos blancos y negros (por ejemplo, un tablero de ajedrez), un patrón de un bit es suficiente para representar un pixel. Un patrón de 0 representa un pixel negro y uno de 1 representa un pixel blanco. Luego los patrones se registran uno tras otro y se almacenan en la computadora. La figura 2.7 muestra una imagen de este tipo y su representación.

Si una imagen no se forma de pixeles puramente blancos y pixeles puramente negros, usted puede aumentar el tamaño del patrón de bits para representar escalas de grises. Por ejemplo, para mostrar cuatro niveles de la escala de grises, se puede usar un patrón de dos bits. Un pixel negro puede representarse por 00, un gris oscuro por 01, un pixel gris claro por 10 y un pixel blanco por 11.

Para representar imágenes a color, cada pixel coloreado se descompone en tres colores primarios: rojo, verde y azul (RGB). Luego se mide la intensidad de cada color y se le asigna un

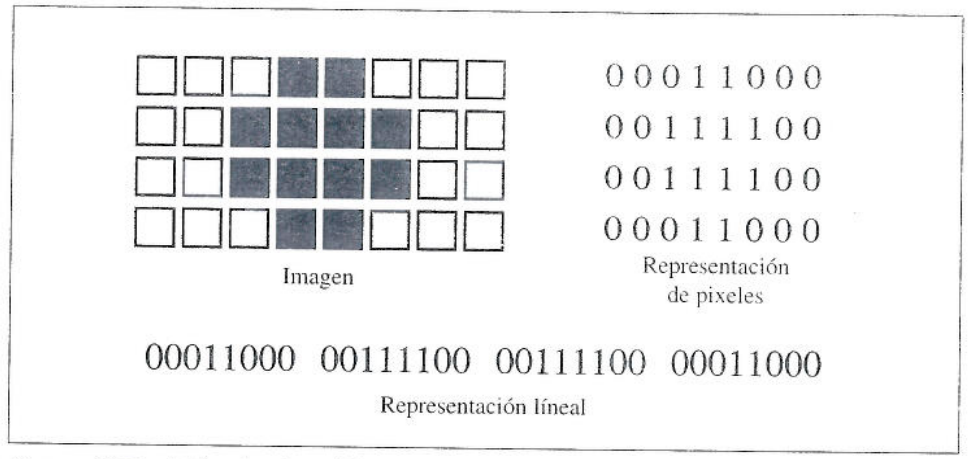

Figura 2.7 Método de gráficos de mapa de bits de una imagen blanca y negra

patrón de bits (por lo general ocho bits). En otras palabras, cada pixel tiene tres patrones de bits: uno para representar la intensidad del color rojo, uno para la intensidad del color verde y uno para la intensidad del color azul. Por ejemplo, la figura 2.8 muestra cuatro patrones de bits para algunos pixeles en una imagen a color.

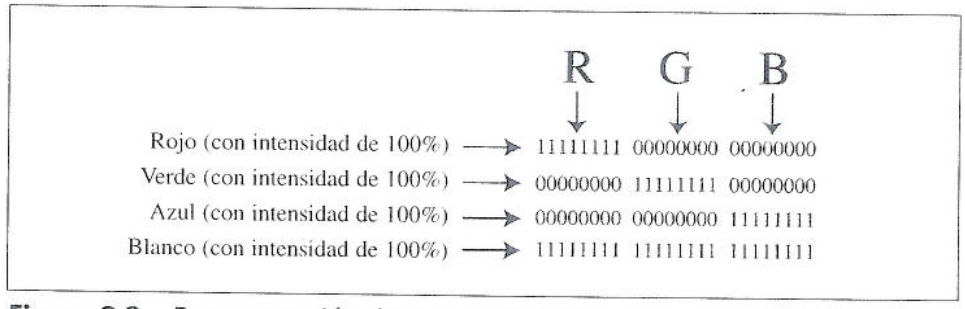

Figura 2.8 Representación de pixeles de color

Gráficos de vectores

El problema con el método de los gráficos de mapa de bits es que los patrones de bits exactos para representar una imagen particular deben guardarse en una computadora. Posteriormente, si usted desea cambiar el tamaño de la imagen debe cambiar el tamaño de los pixeles, lo cual crea una apariencia difusa y granulada. No obstante, el método de gráficos de vector no guarda los patrones de bits. Una imagen se descompone en una combinación de curvas y líneas. Cada curva o línea se representa por medio de una fórmula matemática. Por ejemplo, una línea puede describirse mediante las coordenadas de sus puntos extremos y un círculo puede describirse mediante las coordenadas de su centro y la longitud de su radio. La combinación de estas fórmulas se almacena en una computadora. Cuando la imagen se va a desplegar o imprimir, el tamaño de la imagen se proporciona al sistema como una entrada. El sistema rediseña la imagen con el nuevo tamaño y usa la misma fórmula para dibujar la imagen. En este caso, cada vez que una imagen se dibuja, la fórmula se vuelve a evaluar.

AUDIO

El audio es una representación de sonido o música. Aunque no hay un estándar para almacenar el sonido o la música, la idea es convertir el audio a datos **digitales** y usar patrones de bits. El audio por naturaleza es información **análoga**. Es continuo (análogo), no discreto (digital). La figura 2.9 muestra los pasos a seguir para cambiar los datos de audio a patrones de bits. Estos pasos son los siguientes:

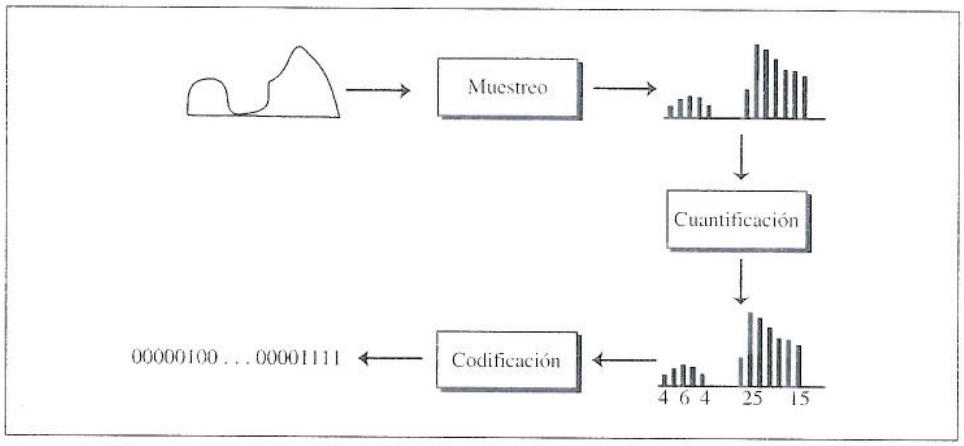

Figura 2.9 Representación de audio

1. La señal análoga se muestrea. El **muestreo** significa medir el valor de la señal a intervalos iguales.
2. Las muestras se cuantifican. La **cuantificación** significa asignar un valor (de un conjunto) a una muestra. Por ejemplo, si el valor de una muestra es 29.2 y el conjunto es el conjunto de enteros entre 0 y 63, se asigna un valor de 29 a la muestra.
3. Los valores cuantificados se cambian a patrones binarios. Por ejemplo, el número 25 se cambia al patrón binario 00011001 (consulte el capítulo 3 para la transformación de números en patrones).
4. Los patrones binarios se almacenan.

VIDEO

El **video** es una representación de imágenes (llamadas cuadros o *frames*) en el tiempo. Una película es una serie de cuadros desplegados uno tras otro para crear la ilusión de movimiento. Así que si usted sabe cómo almacenar una imagen dentro de una computadora, también sabe cómo almacenar un video; cada imagen o cuadro cambia a una serie de patrones de bits y se almacena. La combinación de las imágenes representa el video. Observe que el video actual se comprime normalmente. En el capítulo 15 estudiaremos MPEG, una técnica de compresión de video común.

2.4 NOTACIÓN HEXADECIMAL

El patrón de bits se diseñó para representar datos cuando éstos se almacenan dentro de una computadora. Sin embargo, para la gente es difícil manipular los patrones de bits. Escribir una serie de números 0 y 1 es tedioso y propenso al error. La notación hexadecimal ayuda.

La **notación hexadecimal** se basa en 16 (*hexadec* es la palabra griega para 16). Esto significa que hay 16 símbolos (dígitos hexadecimales): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F. La importancia de la notación hexadecimal se hace evidente cuando se convierte un patrón de bits a notación hexadecimal.

Cada dígito hexadecimal puede representar cuatro bits y cuatro bits pueden representarse mediante un dígito hexadecimal. La tabla 2.2 muestra la relación entre un patrón de bits y un dígito hexadecimal.

> Un patrón de 4 bits puede representarse mediante un dígito hexadecimal, y viceversa.

Patrón de bits	Dígito hexadecimal	Patrón de bits	Dígito hexadecimal
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

Tabla 2.2 Dígitos hexadecimales

CONVERSIÓN

La conversión de un patrón de bits a notación hexadecimal se realiza por medio de la organización del patrón en grupos de cuatro y luego hallar el valor hexadecimal para cada grupo de cuatro bits. Para una conversión de hexadecimal a patrón de bits se convierte cada dígito hexadecimal a su equivalente de cuatro bits (figura 2.10).

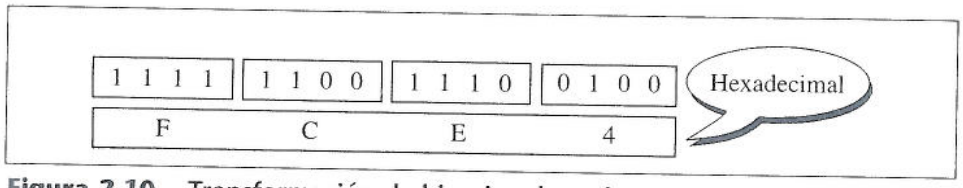

Figura 2.10 Transformación de binario a hexadecimal y de hexadecimal a binario

Observe que la notación hexadecimal se escribe en dos formatos. En el primer formato, usted añade una x minúscula (o mayúscula) antes de los dígitos para mostrar que la representación está en hexadecimal. Por ejemplo, xA34 representa un valor hexadecimal en esta convención. En otro formato, usted indica la base del número (16) como el subíndice después de cada notación. Por ejemplo, $A34_{16}$ muestra el mismo valor en la segunda convención. En este libro se usan ambas convenciones.

EJEMPLO 1

Determine el hexadecimal equivalente del patrón de bits 110011100010.

SOLUCIÓN

Cada grupo de cuatro bits se traduce a un dígito hexadecimal. El equivalente es xCE2. ■

EJEMPLO 2

Determine el hexadecimal equivalente del patrón de bits 0011100010.

SOLUCIÓN

El patrón de bits se divide en grupos de cuatro bits (a partir de la derecha). En este caso, se añaden dos 0 más a la izquierda para hacer el número total de bits divisible entre cuatro. Así que usted tiene 000011100010, lo cual se traduce a x0E2. ■

EJEMPLO 3

¿Cuál es el patrón de bits para x24C?

SOLUCIÓN

Cada dígito hexadecimal se escribe como su patrón de bits equivalente y se obtiene 001001001100. ■