2

Sistemas numéricos

Contenido

Objetivos

- Justificar la utilización de dígitos binarios como elementos del lenguaje interno de la computadora.
- Justificar el empleo de los sistemas hexadecimal y octal como medio para "empaquetar" entidades binarias.
- Incorporar las nociones de operaciones en sistema binario, suma hexadecimal, suma octal y complemento binario.

2.1 Introducción

En este capítulo abordaremos temas que nos muestran cómo se convierte un número binario a decimal, cómo se lo representa con menos símbolos en octal y en hexadecimal y los códigos más difundidos para la representación de información.

Se hará especial hincapié en el sistema binario, por ser el que utilizan las computadoras.

El sistema binario representa todos los símbolos del usuario –como letras y números– o los caracteres –como los signos de puntuación– o los comandos –como **<ENTER>**– con una secuencia de dígitos binarios llamados "bits" (BIT es el acrónimo de *binary digit* o dígito binario). Los bits de datos individuales pueden combinarse en agrupaciones de *8* bits llamadas "byte" (*1 byte = 8 bits*). No sólo los datos que se procesan están representados en binario, sino también las instrucciones de los programas. La computadora codifica los datos para realizar cálculos, ordenar alfabéticamente y demás transformaciones de información binaria. Por ejemplo, la letra "*A*" se codifica como "*01000001*" y la "*B*", como "*01000010*" en el método de codificación más usado para representar caracteres alfanuméricos (tanto para almacenarlos como para mostrarlos en el monitor e imprimirlos). El valor binario correspondiente a la letra "*A*" es menor que el correspondiente a la "*B*"; esto lleva a la conclusión de que si queremos ordenar el par de letras nos basta con saber cuál es la menor para conocer cuál es la primera. Las computadoras digitales y muchos otros equipos electrónicos se basan en un sistema binario formado por dos únicos símbolos: *0* y *1*. Desde que un usuario presiona las teclas de un teclado, el universo de su información se representará en binario. De este modo, debemos saber que dentro de la computadora se almacenan, transfieren y procesan sólo dígitos binarios. Desde el punto de vista físico, los bits pueden ser representados por una magnitud de voltaje o un campo magnético.

2.2 Sistemas de notación posicional

Los sistemas de notación posicional están formados por un juego de n cantidad de símbolos, cuya combinación representa valores diferentes. El concepto de criterio posicional es que cada dígito tiene un peso distinto según el lugar que ocupa; el **peso** es la base elevada a la posición que ocupa dentro del número. La suma de cada dígito multiplicado por su peso permitirá obtener el valor final del número.

En todo sistema posicional, dada una base B se tienen B digitos posibles, de *0* a B-*1*.

2.2.1 Expresión generalizada de un número en potencias de su base

Dado un número N en base B, expresado por n dígitos (d), éste será igual a la sumatoria de cada dígito (d) por la potencia de la base B que le corresponda a su posición.

$$N = d_n \cdot B^n + d_{n-1} \cdot B^{n-1} + ... + d_1 \cdot B^1 + d_0 \cdot B^0$$

$$N = d_n\, d_{n-1} ..... d_1\, d_0 = d_n \cdot 10^n + d_{n-1} \cdot 10^{n-1} + ... + d_1 \cdot 10^1 + d_1 \cdot 10^0$$

Más adelante podremos comprobar que la base (B) se representa con los dígitos *10* (*1*, *0*) para cualquier sistema.

2.2.2 Sistema decimal

El **sistema de numeración decimal** es un sistema de notación posicional formado por *10* símbolos (*0, 1, 2, 3, 4, 5, 6, 7, 8, 9*). Entonces, el número decimal *1.023* estará conformado por la suma de las siguientes cantidades:

$10^3 10^2 10^1 10^0$

$1\ 0\ 2\ 3_{(10)}$

$3 \cdot 10^0 = 3$

$2 \cdot 10^1 = 20$

$0 \cdot 10^2 = + 0$

$1 \cdot 10^3 = \underline{1.000}$

$1.023_{(10)}$

2.2.3 Sistema binario

El **sistema binario**, o de base *2*, también es un sistema de notación posicional, formado por dos símbolos (*0*, *1*) a los que se denomina **bits**. El sistema binario permite analogar sus dos símbolos (*0* y *1*) con dos "estados" posibles; por lo tanto, los valores *0* y *1* se pueden interpretar, por ejemplo, como:

OFF	*ON*
Falso	Verdadero
No	Sí

Un número binario está compuesto por un conjunto de bits y su equivalente decimal será igual a la suma que resulte de multiplicar cada bit por las potencias de *2* (la base B) que correspondan a su posición. Así, el número binario:

$1\ 1\ 0\ 1\ 1_{(2)}$

$1 \cdot 2^0 = 1$

$1 \cdot 2^1 = 2$

$0 \cdot 2^2 = + 0$

$1 \cdot 2^3 = 8$

$1 \cdot 2^4 = \underline{16}$

$27_{(10)}$

El número $11011_{(2)}$ se lee: uno, uno, cero, uno, uno, en base *2*.

Se debe dejar en claro que, por ejemplo, el número *1020* no pertenece al sistema binario, ya que el dígito *2* no pertenece al conjunto de símbolos binarios.

Sistema binario o de base 2: es un sistema de notación posicional, formado por dos símbolos (0,1) a los que se denomina bits.

2.2.4 Sistema octal

El **sistema octal**, o de base *8*, es un sistema posicional formado por ocho símbolos (*0*, *1*, *2*, *3*, *4*, *5*, *6*, *7*); el peso de cada cifra son las potencias sucesivas de *8*. De esta manera:

$2\ 7\ 5_{(8)}$

$5 \cdot 8^0 = 5$

$7 \cdot 8^1 = + 56$

$2 \cdot 8^2 = \underline{128}$

$189_{(10)}$

Sistema octal o de base 8: es un sistema posicional formado por ocho símbolos (0, 1, 2, 3, 4, 5, 6, 7); el peso de cada cifra son las potencias sucesivas de 8.

El número $275_{(8)}$ se lee: dos, siete, cinco, en base *8*.

El número *948* no pertenece al sistema octal, porque los dígitos *9* y *8* no pertenecen a su conjunto de símbolos.

2.2.5 Sistema hexadecimal

El **sistema hexadecimal**, o de base *16*, en general abreviado con las siglas "Hexa", "H" o "h", es un sistema posicional formado por dieciséis símbolos (*0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F*) que representan los valores decimales de *0* a *15*; el peso de cada cifra son las potencias sucesivas de *16*.

Luego, si queremos hallar el equivalente decimal del número hexadecimal *1A2*, lo haremos de la siguiente forma:

$$1\ A\ 2_{(16)}$$

$$2 \cdot 16^0 = 2$$

$$10 \cdot 16^1 = +\ 160$$

$$1 \cdot 16^2 = \underline{256}$$

$$418_{(10)}$$

El número $1A2_{(16)}$ se lee: uno, a, dos, en base *16*.

Es importante recalcar que un número hexadecimal suele identificarse con la letra "h" luego del dígito menos significativo, por ejemplo *1A2h*. En ese caso, "h" no es el dígito menos significativo del número, sino que cumple la función de identificar un sistema de numeración. De cualquier manera, "h" no podría nunca expresar un dígito hexadecimal, ya que no pertenece al conjunto de símbolos del sistema.

En la tabla 2-1 se presentan las equivalencias entre los sistemas decimal, binario, octal y hexadecimal:

Tabla 2-1. Equivalencias entre los sistemas de bases 10, 2, 8 y 16.

Decimal	*Binario*	*Octal*	*Hexadecimal*
0	*0*	*0*	*0*
1	*1*	*1*	*1*
2	*10*	*2*	*2*
3	*11*	*3*	*3*
4	*100*	*4*	*4*
5	*101*	*5*	*5*
6	*110*	*6*	*6*
7	*111*	*7*	*7*
8	*1000*	*10*	*8*
9	*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 *10* (uno, cero) es el número que en cualquier sistema representa la base, por ejemplo en base *5*

$$10_{(5)} = 5_{(10)}$$

luego $100_{(5)} = 5^2{}_{(10)}$, *o sea, la base al cuadrado.*

$1\ 0\ 0_{(5)}$

$0 \cdot 5^0 = 0$

$0 \cdot 5^1 = 0$

$1 \cdot 5^2 = 25$

$25_{(10)}$

$25_{(10)} = 5^2{}_{(10)}$

Como se puede observar en la tabla 1-1, cada sistema de notación posicional construye sus cantidades de la siguiente manera:

- Primero, utiliza los números que le pertenecen, en orden creciente, desde el primero (el cero) hasta agotar el último.
- Segundo, repite el paso anterior, pero incrementando el dígito de orden superior en una unidad.
- Tercero, repite los dos primeros pasos, tantas veces como sea necesario, para representar el número deseado.

Para analizar los contenidos de los registros internos de la computadora, se utilizan los sistemas octal y hexadecimal como "métodos taquigráficos", que permiten representar su contenido binario compactado. Esto es posible debido a que *8* y *16* **son potencias exactas de *2*** ($8 = 2^3$ y $16 = 2^4$); en consecuencia, es factible convertir números del sistema binario al octal y al hexadecimal tomando agrupaciones de *3* y *4* bits, respectivamente. Por ejemplo, el número binario $11110010_{(2)} = 362_{(8)}$ y $F2_{(16)}$.

Para lograr esta conversión, la cadena binaria *11110010* se divide en grupos de *3* bits, de derecha a izquierda y, si fueran necesarios para completar el grupo, se agregan ceros a la izquierda del último bit:

011	110	010	y luego se le asigna el dígito octal equivalente
3	6	$2_{(8)}$.	

De la misma manera, la cadena binaria *11110010* se divide en grupos de *4* bits:

1111	0010	y luego se le asigna el dígito hexadecimal equivalente
F	$2_{(16)}$.	

Queda claro que los sistemas octal y hexadecimal permiten "compactar" bits, de modo de hacer más sencilla la tarea de reconocerlos a simple vista. Sólo se debe asumir el esfuerzo de aprender cómo se convierte directamente desde binario a base *8* o *16*, y al revés. Por esta razón, cuando un usuario pretende visualizar la información "desde adentro", la computadora asume uno de estos dos sistemas y simplifica la tarea de leer ceros y unos. Si se comparan ambos métodos, se nota que el sistema hexadecimal compacta mejor la cadena binaria que el octal, esto es, indica lo mismo con menor cantidad de símbolos. Por lo tanto, se puede inferir que cuanto más grande es la base, más importante es la reducción. Cabe preguntarse, entonces, por qué no utilizar un sistema base *32* o *64*; la razón es simple, resultaría engorroso recordar *32* o *64* símbolos diferentes y se estaría ante un problema casi mayor que el de operar con bits.

Los sistemas octal y hexadecimal permiten "compactar" bits, de modo de hacer más sencilla la tarea de reconocerlos a simple vista.

2.2.6 Número de cifras. Cantidad decimal máxima

Repase de nuevo la tabla de equivalencias. Note que con **un** bit se pueden representar los dígitos decimales *0* y *1*. Con **dos** bits se pueden representar los dígitos decimales *0*, *1*, *2* y *3*. Con **tres** bits, los dígitos decimales *0* a *7*, con **cuatro** bits, los dígitos decimales *0* a *15*.

Si se llama **n** a la cantidad de bits en juego, se puede generalizar el concepto:

> *con* **n** *bits se pueden representar los valores decimales comprendidos entre 0 y* $2^n - 1$.

Nótese que de la misma tabla se deduce que:

> *con* **n** *dígitos octales se pueden representar los valores decimales comprendidos entre 0 y* $8^n - 1$.

De la misma manera,

> *con* **n** *dígitos hexadecimales se pueden representar los valores decimales comprendidos entre 0 y* $16^n - 1$.

2.3 Métodos de conversión de números enteros y fraccionarios

Ahora, es preciso establecer qué relación hay entre los distintos sistemas numéricos estudiados y el sistema decimal que estamos acostumbrados a utilizar. Después de cierto tiempo de práctica, puede resultar bastante simple reconocer directamente que $15_{(10)}$ es igual a $1111_{(2)}$ y que ambas entidades están identificando *15* elementos; sin embargo, es factible que surjan problemas cuando la cantidad de dígitos de un sistema no decimal es muy grande, por ejemplo, $1111100010101001 01_{(2)}$. Para lograrlo se pueden utilizar los siguientes métodos que permiten la conversión entre sistemas de bases distintas para números enteros y fraccionarios.

2.3.1 Método de conversión de números de otras bases a decimal

A continuación, se describen los métodos que permiten convertir números de cualquier sistema de notación posicional al sistema decimal.

2.3.1.1. Binario a decimal (2) a (10)

Enteros

Sea el número $10111_{(2)}$, se obtendrá su valor decimal multiplicando cada bit por la potencia de dos que corresponda a su posición y sumando los valores finales.

$2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$1\ 0\ 1\ 1\ 1_{(2)}$

$1 \cdot 2^0 = 1$

$1 \cdot 2^1 = 2$

$1 \cdot 2^2 = +\ 4$

$0 \cdot 2^3 = 0$

$1 \cdot 2^4 = 16$

$23_{(10)}$

Fracciones

Sea el número $0,101_{(2)}$, se obtendrá su valor decimal multiplicando cada bit por la potencia negativa de dos que corresponde a su posición, a partir de la coma y sumando los valores finales.

$2^0\ 2^{-1}\ 2^{-2}\ 2^{-3}$

$0,\ 1\ \ 0\ \ 1_{(2)}$

$1 \cdot 2^{-3} = 0,125$

$0 \cdot 2^{-2} = +0$

$1 \cdot 2^{-1} = \underline{0,5}$

$0,625_{(10)}$

Como la conversión de binario a decimal será con la que usted se enfrentará más a menudo, por lo menos en este texto, es conveniente que recuerde el equivalente decimal de las primeras potencias negativas de dos.

Para recordar:

$2^{-1} = 0,5$

$2^{-2} = 0,25$

$2^{-3} = 0,125$

$2^{-4} = 0,0625$

$2^{-5} = 0,03125$

2.3.1.2. Octal a decimal (8) a (10)

Sea el número $17.3_{(8)}$, se obtendrá su valor decimal multiplicando cada dígito octal por la potencia de ocho que corresponda a su posición y sumando los valores finales.

$1\ 7\ ,\ 3_{(8)}$

$3 \cdot 8^{-1} = 3 \cdot \frac{1}{8} = 0,375$

$7 \cdot 8^{0} = 7 \cdot 1 = +7$

$1 \cdot 8^{1} = 1 \cdot 8 = \underline{\ 8}$

$15,375_{(10)}$

2.3.1.3. Hexadecimal a decimal (16) a (10)

Sea el número $1A,0F_{(16)}$, se obtendrá su valor decimal multiplicando cada dígito hexadecimal por la potencia de dieciséis que corresponda a su posición y sumando los valores finales.

$1\ A,0\ F_{(16)}$

$15 \cdot 16^{-2} = 15 \cdot \frac{1}{256} = 0,058...$

$0 \cdot 16^{-1} = 0 \quad = +\ 0$

$10 \cdot 16^{0} = 10 \cdot 1 = 10$

$1 \cdot 16^{1} = 1 \cdot 16 = \underline{\ 16}$

$26,058..._{(10)}$

2.3.2 Método de divisiones sucesivas (para convertir un número entero decimal a otras bases)

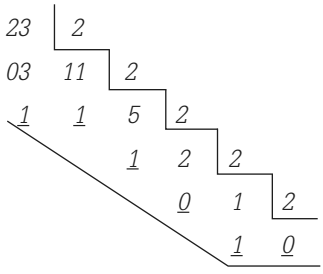
Para convertir un número entero decimal a cualquier sistema, se divide el número por la base que corresponda hasta hallar el último cociente (o hasta que el último cociente sea cero), que formará con todos los restos anteriores (desde el último hasta el primero) el número buscado.

Para convertir un número entero decimal a cualquier sistema, se divide el número por la base que corresponda hasta hallar el último cociente .

El seguimiento de los ejemplos siguientes permitirá eliminar dudas.

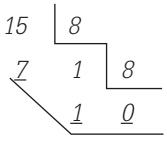
2.3.2.1. Decimal a binario (10) a (2)

$23_{(10)\ a\ (2)} = 10111_{(2)}$



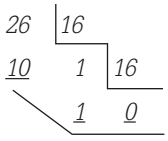
2.3.2.2. Decimal a octal (10) a (8)

$15_{(10)\ a\ (8)} = 17_{(8)}$



2.3.2.3. Decimal a hexadecimal (10) a (16)

$26_{(10)\ a\ (16)} = 1A_{(16)}$



En este caso, el primer resto es "10" y debe ser reemplazado por su equivalente hexadecimal "A".

2.3.3 Método de multiplicaciones (para convertir un número fraccionario decimal a otras bases)

Para convertir un número fraccionario decimal a cualquier sistema, primero se debe recordar que de una fracción pura se obtiene, después de la conversión, otra fracción pura. Por eso, cuando un número tiene parte entera y parte fraccionaria se lo separa y se aplica un método para la conversión de la parte entera y otro para la parte fraccionaria, como se describe a continuación.

Algoritmo para la conversión de un número fraccionario decimal a otras bases

1. Multiplicar la fracción por la base de conversión (en el caso de que la conversión se realice a sistema binario, la base es 2), cuyo resultado tiene una parte entera, que se considerará uno de los dígitos fraccionarios buscados.

2. Del resultado se toma sólo la parte fraccionaria y se vuelve a multiplicar por la base, para hallar el siguiente dígito fraccionario buscado. Este procedimiento se realiza tantas veces como sea necesario, hasta cancelar el método por alguno de los motivos que siguen:

- La parte fraccionaria en una iteración se hace cero.

- En la conversión comienza a repetirse una secuencia de dígitos fraccionarios, lo que implica que es una fracción periódica.
- Se hallaron varios dígitos y se considera que la fracción obtenida tiende al resultado esperado por la conversión, y el error es infinitesimal.

En consecuencia, para todos los casos, el nuevo número fraccionario estará compuesto por la parte entera resultante de cada multiplicación; éstas se toman en orden descendente, o sea, desde la primera hasta la última iteración. La flecha (↓) indica el orden en que se deberán tomar los dígitos. Veamos los ejemplos.

2.3.3.1. Decimal a binario (10) a (2)

1er. caso) $0{,}625_{(10)\ a\ (2)} = 0{,}101_{(2)}$

$$0{,}625 \cdot 2 = \underline{1}{,}250$$
$$0{,}25 \cdot 2 = \underline{0}{,}5$$
$$0{,}5 \cdot 2 = \underline{1}$$

Deben tomarse en cuenta las consideraciones siguientes:

- Cada parte entera así obtenida no debe superar el valor de la base del sistema; si esto se produce, hay un error en la multiplicación.
- Cada parte entera, formada por dos dígitos decimales, debe representarse con un solo símbolo del sistema al que se convierte. Así, el pasaje de decimal a hexadecimal de $0{,}625 \cdot 16 = 10{,}0$ y la cifra 10 es reemplazada por $A_{(16)}$.

2do. caso) $0{,}2_{(10)\ a\ (2)} = 0{,}\widehat{0011}_{(2)}$

$$0{,}2 \cdot 2 = \underline{0}{,}4$$
$$0{,}4 \cdot 2 = \underline{0}{,}8 \quad \text{período}$$
$$0{,}8 \cdot 2 = \underline{1}{,}6$$
$$0{,}6 \cdot 2 = \underline{1}{,}2$$
$$0{,}2 \cdot 2 = 0{,}4$$

El arco en el resultado señala el conjunto de números que se repiten, o sea, la periodicidad.

3er. caso) $0{,}122_{(10)\ a\ (2)} = 0{,}000111\ldots_{(2)}$

$$0{,}12 \cdot 2 = \underline{0}{,}24$$
$$0{,}24 \cdot 2 = \underline{0}{,}48$$
$$0{,}48 \cdot 2 = \underline{0}{,}96$$
$$0{,}96 \cdot 2 = \underline{1}{,}92$$
$$0{,}92 \cdot 2 = \underline{1}{,}84$$
$$0{,}84 \cdot 2 = \underline{1}{,}68$$
$$\ldots\ldots\ldots\ldots$$
$$\ldots\ldots\ldots\ldots$$

El método se cancela cuando la parte fraccionaria resultante de una multiplicación da *0*, cuando se obtiene periodicidad o cuando se suspende luego de varias iteraciones, de acuerdo con la aproximación de la conversión que se crea conveniente. En este caso, se debe calcular el error cometido y considerar si éste es tolerable o afecta demasiado el resultado del cálculo en el que interviene.

2.3.3.2. Decimal a octal (10) a (8)

1er. caso) $0{,}625_{(10)\ a\ (8)} = 0{,}5(8)$

$0{,}625 \cdot 8 = \underline{5},0$

2do. caso) $0{,}2_{(10)\ a\ (8)} = 0{,}\widehat{1463}_{(8)}$

$0{,}2 \cdot 8 = \underline{1},6$
$0{,}6 \cdot 8 = \underline{4},8$ | *período*
$0{,}8 \cdot 8 = \underline{6},4$
$0{,}4 \cdot 8 = \underline{3},2$ |
$0{,}2 \cdot 8 = 1{,}6$ |

3er. caso) $0{,}1_{(10)\ a\ (8)} = 0{,}\widehat{06314}_{(8)}$

$0{,}1 \cdot 8 = \underline{0},8$ |
$0{,}8 \cdot 8 = \underline{6},4$
$0{,}4 \cdot 8 = \underline{3},2$ | *período*
$0{,}2 \cdot 8 = \underline{1},6$
$0{,}6 \cdot 8 = \underline{4},8$ |
$0{,}8 \cdot 8 = 6{,}4$ |

4to. caso) $0{,}12_{(10)\ a\ (8)} = 0{,}07534\ldots_{(8)}$

$0{,}12 \cdot 8 = \underline{0},96$
$0{,}96 \cdot 8 = \underline{7},68$
$0{,}68 \cdot 8 = \underline{5},44$
$0{,}44 \cdot 8 = \underline{3},52$
$0{,}52 \cdot 8 = \underline{4},16$

Se suspende con una aproximación conveniente.

2.3.3.3. Decimal a hexadecimal (10) a (16)

1er. caso) $0{,}25_{(10)\ a\ (16)} = 0{,}A_{(16)}$

$0{,}625 \cdot 16 = \underline{10},0 = 0{,}A_{(16)}$

El supuesto "resto" es *10* y debe ser reemplazado por el símbolo hexadecimal A.

2do. caso) $0{,}2_{(10)\ a\ (16)} = 0{,}\widehat{3}_{(16)}$

$0{,}2 \cdot 16 = \underline{3},2$ | *período*
$0{,}2 \cdot 16 = 3{,}2$

3er. caso) $0{,}12_{(10)\ a\ (16)} = 0{,}1EB85\ldots_{(16)}$

$0{,}12 \cdot 16 = \underline{1},92$
$0{,}92 \cdot 16 = \underline{E},72$
$0{,}72 \cdot 16 = \underline{B},52$
$0{,}52 \cdot 16 = \underline{8},32$
$0{,}32 \cdot 16 = \underline{5},12$
.....
.....

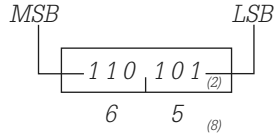
Se suspende con una aproximación conveniente.

2.3.4 Pasaje directo entre las bases 2 a 8 y 2 a 16

El pasaje directo entre los sistemas binario y octal, y entre binario y hexadecimal, se puede realizar porque *8* y *16* son potencias exactas de *2*.

2.3.4.1. Binario a octal (2) a (8)

Si se observa la tabla de equivalencias entre sistemas, se ve que todo símbolo octal se representa, a lo sumo, con una combinación de *3* bits. Así, el número $110101_{(2)}$ podría representarse en octal como:



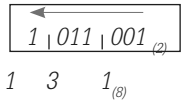
Llamaremos **LSB** (*Least Significant Bit* o de menor peso) al primer bit de la derecha y **MSB** (*Most Significant Bit* o de mayor peso) al primer bit de la izquierda del número binario. Luego, el procedimiento para convertir un número binario a octal en forma "directa" es el que se describe a continuación.

Conversión directa de binario a octal

1. Se divide el número binario en grupos de *3* bits, en el sentido siguiente: para enteros desde el LSB hacia el MSB y para fracciones en sentido inverso.
2. Luego se sustituye cada grupo de *3* bits por el correspondiente dígito octal, con cuidado de completar con ceros a la derecha la parte fraccionaria, de ser necesario para obtener el grupo de *3*.

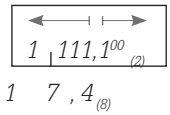
Enteros

Sea el número $1011001_{(2)}$, tomado como ejemplo para la conversión de binario a octal, el seguimiento del procedimiento da como resultado:



Fracciones

En primer término se deben formar grupos de *3* dígitos binarios, teniendo presente que en la parte fraccionaria cambia el sentido de la agrupación, que se encuentra representado por la orientación de las flechas.



El motivo de formar grupos de *3* dígitos binarios se debe a que la base es potencia exacta de dos, $8 = 2^3$, luego el exponente indica el tamaño de la agrupación.

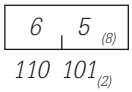
Los dos ceros de la fracción se agregan para completar el grupo y se señalaron como superíndices para marcar el procedimiento. El agregado de ceros para completar el grupo es imprescindible para lograr la conversión correcta. Obsérvese que *0,1* en sistema binario (*1/2*) no equivale a *0,1* en sistema octal (*1/8*), y es muy diferente aun al resultado obtenido si se rellena el grupo con ceros a la derecha, como lo indica el procedimiento, que permite un resultado final de *0,4* en octal.

2.3.4.2 Octal a binario (8) a (2)

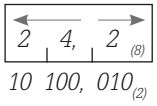
En forma inversa a la enunciada antes, se puede convertir directamente un número octal a su correspondiente en base *2*, reemplazando cada dígito octal por el equivalente binario en grupos de *3* bits.

Enteros

Así, si se toma el ejemplo anterior, $65_{(8)}$ se convertirá directamente a binario reemplazando cada dígito octal por el grupo de tres dígitos binarios que le correspondan.



Fracciones



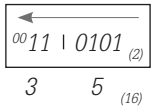
El sentido de las agrupaciones para números fraccionarios es siempre el mismo.

2.3.4.3 Binario a hexadecimal (2) a (16)

La conversión en forma directa de un binario a un hexadecimal sigue los mismos pasos enunciados para la conversión a un octal.

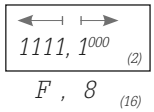
Enteros

Nótese que cada símbolo hexadecimal se representa, a lo sumo, con una combinación de *4* bits. Si se toma el número $110101_{(2)}$ y se lo divide en grupos de *4* bits empezando por el LSB, al sustituir cada grupo por su dígito hexadecimal correspondiente, se obtendrá:



El criterio de formar grupos de *4* bits es el mismo que se tomó para la conversión a octal, teniendo en cuenta que en hexadecimal la base es potencia exacta de $2,16 = 2^4$, y el exponente *4* es el que indica la cantidad de dígitos binarios que se deben agrupar para esta conversión.

Fracciones

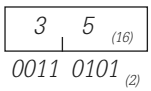


2.3.4.4 Hexadecimal a binario (16) a (2)

De manera inversa, se podrá convertir directamente un hexadecimal a binario, reemplazando cada dígito hexadecimal por su correspondiente binario en grupos de *4* bits.

Entero

Sea el número $35_{(16)}$ la conversión directa a binario dará como resultado:



Fracciones

Si se desea convertir una fracción hexadecimal a binario en forma directa, esto se realizará de manera semejante a la conversión octal-binaria, sólo que los grupos se deberán armar de *4* dígitos binarios.

$F,\ 8_{(16)}$

$1111,1000_{(2)}$

De idéntica forma se pueden realizar pasajes directos entre distintas bases, donde una sea potencia exacta de la otra.

2.4 Operaciones fundamentales en binario

2.4.1 Suma

La suma entre *2* bits tiene cuatro combinaciones de operación posibles:

0 + 0 = 0

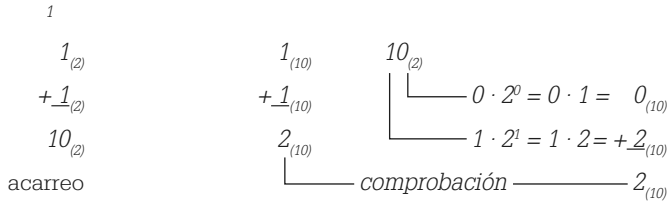
0 + 1 = 1

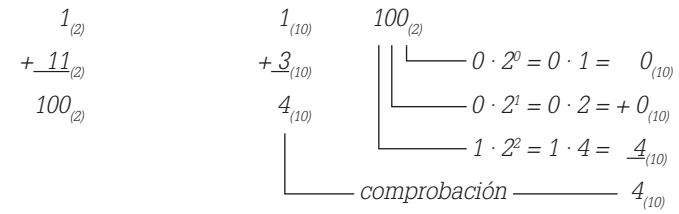
1 + 0 = 1

1 + 1 = 0 **y me llevo 1; o acarreo 1**

Observemos algunos ejemplos para entender el mecanismo de la suma.

Si quiere sumar *1* más *1* en binario, el resultado será *10*, o sea, la base binaria (ya que tiene sólo dos elementos). Expresado de otra manera, el resultado es *0* y me llevo *1* (o el acarreo es *1*). Si ahora, para comprobar que el resultado es el correcto, se halla el equivalente de $10_{(2)}$ en decimal, se obtiene $2_{(10)}$, que es el mismo resultado que surge de sumar *1* más *1* en decimal.





En el caso en que se desee sumar *1* más *11* en binario, el resultado será $100_{(2)}$.

Encuentre un simulador que permite realizar conversiones y operaciones entre sistemas numéricos en la Web de apoyo.

2.4.2 Resta o sustracción

Es común que la operación de resta en una computadora se realice con una suma, esto es, se obtenga una resta al sumar el minuendo y el complemento a la base del sustraendo. Sin embargo, a efectos de explicar la operatoria manual, y teniendo en cuenta que será de gran utilidad para el lector (por ejemplo, en el seguimiento de la ejecución de un programa cuyos resultados no son los esperados), se explicarán otros mecanismos que se pueden aplicar para la resta de dos números binarios:

La operación de resta en una computadora se calcula mediante la suma del minuendo y el complemento a la base del sustraendo.

- Uno de ellos tiene en cuenta las combinaciones siguientes para la resta de cada par de dígitos:

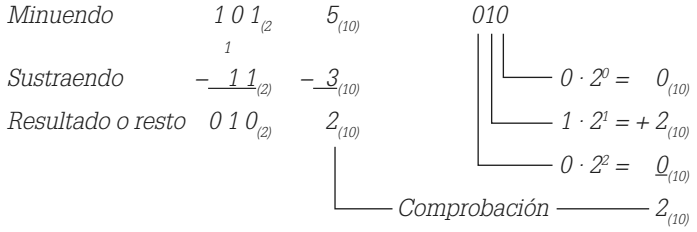
0 – 0 = 0

0 – 1 = 1 **le pide 1 a la posición siguiente (produce un acarreo)**

1 – 0 = 1

1 – 1 = 0.

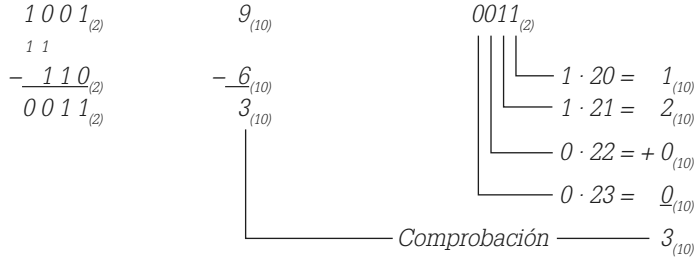
Supongamos que se debe efectuar la resta $101_{(2)} - 11_{(2)}$:



Si se aplica este mecanismo a cada par de dígitos, en la primera columna el resultado de restar *1* menos *1* es *0*. En la segunda columna, *0* menos *1* es *1*, y proviene de haberle "pedido *1*" a la posición siguiente (indicado con un ^{1} en la tercera columna). Por último, en la tercera columna, *1* menos ^{1} es *0*. De la misma forma que se hizo para la suma, se puede comprobar que el resultado es el correcto, hallando el equivalente de $010_{(2)}$, que es *2* en decimal.

- Otro mecanismo que se utiliza, si no se puede memorizar el anterior, consiste en preguntar para cada par de dígitos: ¿cuál será el dígito que sumado al sustraendo me da el minuendo?, o ¿cuánto le falta al sustraendo para llegar al minuendo? Tratemos de utilizarlo, realizando las restas que se presentan a continuación.

  1er. caso)



Para realizar la resta $1001_{(2)} - 110_{(2)}$, si en la primera columna preguntamos cuánto le falta a *0* para llegar a *1*, la respuesta es *1*.

En la segunda columna, la pregunta ¿cuánto le falta a *1* para llegar a *0*? no podrá responderse, ya que el minuendo es menor que el sustraendo y, por lo tanto, le debe "pedir *1*" a la columna siguiente, con lo que la pregunta pasa a ser ¿cuánto le falta a *1* para llegar a *10*? La respuesta es *1*, y el pedido de una unidad a la columna siguiente se encuentra reflejado con un ^{1} entre el minuendo y el sustraendo en esa columna.

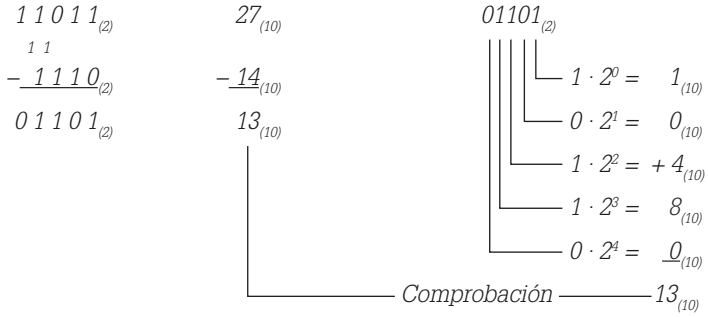
Es importante mencionar que "pedir *1*" significa pedirle a la base siguiente una unidad, que "siempre" va a ser equivalente a *10*, y sumársela al minuendo.

Para llegar al resultado de la tercera columna, en este caso, hay que realizar un paso previo antes de hacer la pregunta, que consiste en sumar al sustraendo la unidad que se había "pedido" con anterioridad y luego preguntar. De esta manera, *1* más ^{1} es *0* con acarreo a la columna siguiente. El resultado de esta columna es *0* y el pedido de unidad se encuentra reflejado en la columna siguiente.

La pregunta de la cuarta columna se contesta rápido: ¿cuánto le falta a *1* para llegar a *1*? La respuesta es *0*.

No desespere, es más difícil explicar el desarrollo de estos métodos que aplicarlos.

2do. caso)



En la resta $11011_{(2)} - 1110_{(2)}$, si preguntamos, para la primera columna, ¿cuánto le falta a *0* para llegar a *1*?, la respuesta es simple: *1*.

En la segunda columna, si hacemos la pregunta ¿cuánto le falta a *1* para llegar a *1*?, la respuesta es *0*.

En la tercera columna, para la pregunta ¿cuánto le falta a *1* para llegar a...?, ahora ya sabemos que tiene que ser *10*, la respuesta es *1* y en la columna siguiente se indicó con un [1] la solicitud de una unidad para hacer efectiva la resta.

En la cuarta columna, un paso previo estipula que se debe sumar el sustraendo con la unidad que se había "pedido" y el resultado restarlo del minuendo. Luego, *1* más [1] es *10* y la pregunta para hacer es ¿cuánto le falta a *10* para llegar a...?, en este caso *11*. Usted podrá preguntarse ¿por qué *11*? El mecanismo es siempre el mismo: si se le pide una unidad a la columna siguiente, entrega el equivalente a la base (siempre *10*) y al sumárselo, en este caso a *1*, queda *11*. No olvide reflejar el pedido en la columna de orden superior.

Para finalizar, la última columna es sencilla. A la pregunta ¿cuánto le falta a *1* para llegar a *1*?, respondemos *0*.

Los mecanismos propuestos para la suma y la resta quizá resulten un tanto tediosos, sobre todo si usted ya aprendió a operar con binarios; no obstante, es probable que le den un porcentaje de seguridad en la operatoria que otros mecanismos no poseen.

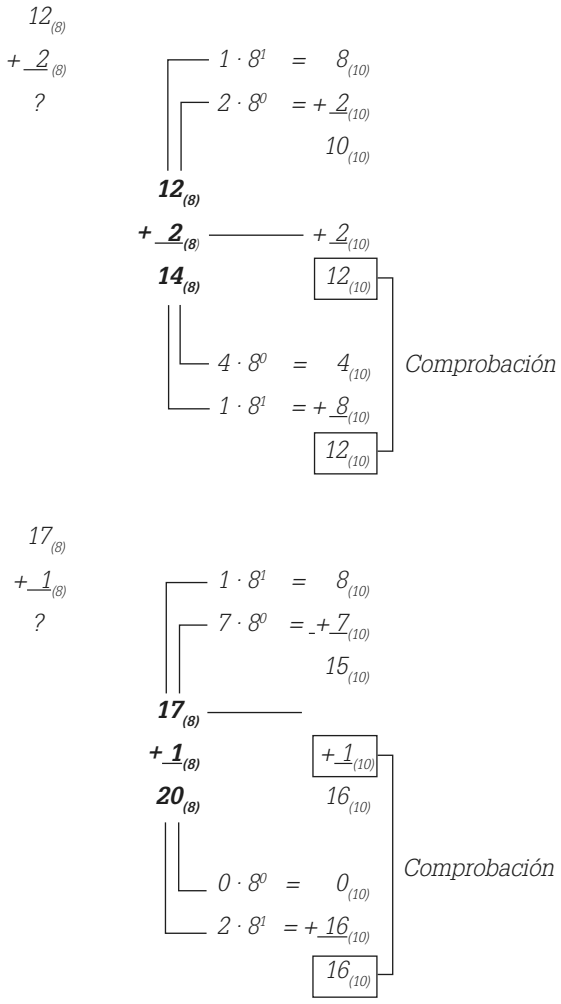
2.5 Operaciones fundamentales en octal y hexadecimal

La técnica para las operaciones en octal o hexadecimal es la misma que para cualquiera de las operaciones en un sistema posicional, por lo tanto, sólo se desarrollará la suma para ambos sistemas. Tenga en cuenta que lo único que cambia es la base del sistema. Además, en relación con la aplicación que usted pueda darle en su ámbito laboral, lo más probable es que tenga que sumar o restar direcciones de memoria expresadas en estos sistemas. Es muy probable que jamás tendrá que realizar una multiplicación u operaciones más complejas; por lo tanto, su estudio se considera innecesario para los fines prácticos de este libro.

Para analizar los ejemplos que se brindan, siga en forma detallada los esquemas presentados, que le permitirán reconocer el mecanismo empleado.

2.5.1 Suma octal

Suponga las siguientes sumas y su comprobación en decimal:



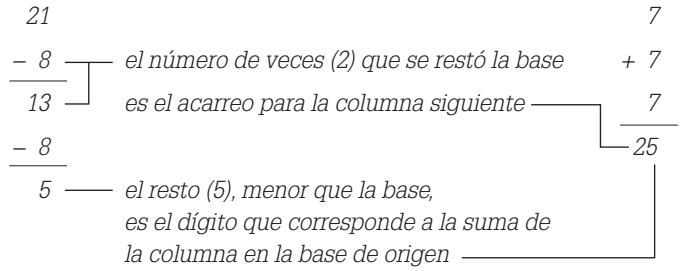
2.5.2 Técnica para sumar números grandes en cualquier base

Si, por ejemplo, se desea realizar la siguiente suma, $7_{(8)} + 7_{(8)} + 7_{(8)}$, la técnica para sumar números grandes en cualquier base es la que se describe a continuación:

- Poner en columnas los números y sumarlos, en base decimal:

$$\begin{array}{r} 7 \\ +\ 7 \\ \hline 7 \\ \hline 27_{(10)} \end{array}$$

- Al resultado de la suma de la columna anterior (*21*), restarle la base de origen (*8*) las veces necesarias hasta que el resto sea menor que esa base:



Luego, el resultado de la suma, en la base de origen, se obtendrá de la siguiente manera:

1. El último resto (dígito menor que la base) será el dígito que corresponda a la resta final de las sucesivas restas, en este caso *5*.
2. La cantidad de veces que se haya restado la base será el acarreo para la columna siguiente, en este caso *2*.

Si los números para sumar tienen más de una columna, la técnica se repetirá para cada una de ellas y el acarreo se sumará a la columna siguiente.

2.5.3 Suma hexadecimal

Suponga las siguientes sumas y su comprobación en decimal:

$12_{(16)}$
$+ 2_{(16)}$
$14_{(16)}$

$1 \cdot 16^1 = 16_{(10)}$
$2 \cdot 16^0 = + 2_{(10)}$
$18_{(10)}$

$\mathbf{12_{(16)}}$
$\mathbf{+ 2_{(16)}}$ — $+ 2_{(10)}$
$\mathbf{14_{(16)}}$ — $20_{(10)}$

$4 \cdot 16^0 = 4_{(10)}$
$1 \cdot 16^1 = + 16_{(10)}$
$20_{(10)}$

Comprobación

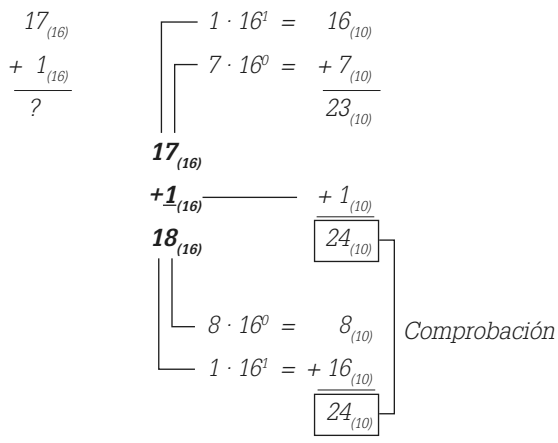
$17_{(16)}$
$+ 1_{(16)}$
$?$

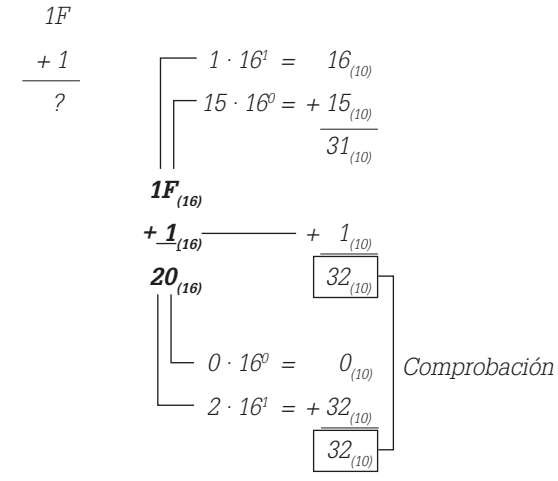
$1 \cdot 16^1 = 16_{(10)}$
$7 \cdot 16^0 = + 7_{(10)}$
$23_{(10)}$

$\mathbf{17_{(16)}}$
$\mathbf{+ 1_{(16)}}$ — $+ 1_{(10)}$
$\mathbf{18_{(16)}}$ — $24_{(10)}$

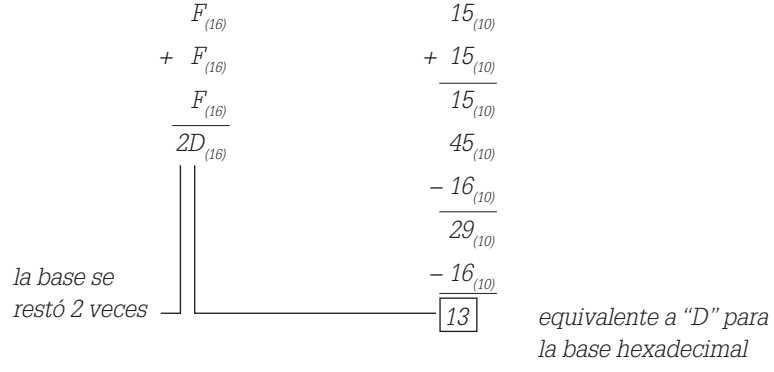
$8 \cdot 16^0 = 8_{(10)}$
$1 \cdot 16^1 = + 16_{(10)}$
$24_{(10)}$

Comprobación





Utilizando la técnica de la suma para números grandes:



2.6 Complemento de un número

Dado que muchas computadoras digitales basan su operatoria aritmética sobre todo en la suma, la mayoría de las operaciones aritméticas debe transformarse a una técnica que la incluya. Por eso, es necesario inferir la idea de complemento de un número, que permitirá, entre otras cosas hacer restas mediante sumas, así como representar números signados. Hay dos métodos que permiten representar el complemento de un número y se detallan a continuación.

2.6.1 Complemento a la base, a la raíz o auténtico

En todo sistema numérico de notación posicional, para un número x de n dígitos existe un número x', también de n dígitos, que es su complemento a la base; tal que

$$x + x' = 10^n, \qquad \text{de lo cual se deduce que}$$

$$\boxed{x' = 10^n - x,} \qquad \text{siendo 10 la base en cualquier sistema.}$$

Otra forma de definirlo es indicar que el complemento a la base de un número N es el resultado de elevar la base a la cantidad de cifras del número, menos el número dado:

$$\boxed{C_b N = B^n - N}$$

En el sistema binario el complemento a la base suele llamarse "complemento a *2*", lógicamente por ser *2* la base del sistema, y suele abreviarse C_2.

Si, por ejemplo, queremos obtener el complemento a la base de los números $0_{(2)}$, $1_{(2)}$, $10_{(2)}$ y $11_{(2)}$, debemos restar estos números de sus respectivas bases, elevadas a la cantidad de dígitos que ellos poseen, o sea de $10^1 = 10$ y de $10^2 = 100$.

10	10	100	100
− 0	− 1	− 10	− 11
10	1	10	01 (*)

Obsérvese que en todos los casos el complemento de un número es su inverso más *1*, o sea, el número inverso del sustraendo + *1*.

2.6.2 Su utilización para la representación de binarios negativos complementados a "2"

Si se desea convertir el decimal *–3* a un binario con signo de *3* bits, se debe considerar la siguiente estructura:

- El bit de extrema izquierda representa con un "*1*" el signo negativo.
- Los *2* bits restantes representan el complemento a *2* del valor binario real, que es *11*, que invertido es *00* y más *1* es *01*.

1	0 1

Signo — *Complemento (calculado en *)*

La cantidad de bits **n**, utilizados para la representación de un número binario con signo, siempre **incluye** el signo.

Para descomplementar un número, o sea, para hallar de nuevo su valor absoluto, el método es el mismo que para hallar el complemento (complemento de complemento).

2.6.3 Complemento a la base –1 o restringido

En todo sistema numérico de notación posicional para un número y de n dígitos, existe un número y', también de n dígitos, que es su complemento restringido; tal que

$$y + y' = 10^n - 1, \qquad \text{de lo cual se deduce que}$$

$$\boxed{y' = (10^n - 1) - y,} \qquad \text{siendo 10 la base en cualquier sistema.}$$

Otra manera de definirlo sería indicar que el complemento a la base *–1* de un número N es el resultado de elevar la base a la potencia dada por la cantidad de cifras del número, menos *1*, y luego restarle el número dado.

$$\boxed{C_{b-1} N = (B^n - 1) - N}$$

En el sistema binario el complemento a la base *–1* es el complemento a *1*; obviamente si la base es *2*, la base *–1* es *1* y suele abreviarse C_1. Coincide con el complemento booleano, o sea que para hallarlo se cambian los *1* por *0* y al revés.

Si al complemento restringido se le suma 1
se obtiene el complemento auténtico.

Si se quiere hallar el complemento restringido de los números $0_{(2)}$ y $11_{(2)}$ resulta:

1	*11*	
– 0	*–11*	*nótese que es el*
1	*00*	*inverso del sustraendo*

2.6.4. Su utilización para la representación de binarios negativos complementados a "1"

Si se desea convertir el decimal *–3* a un binario con signo de *3* bits, se debe considerar la estructura siguiente:

- El bit de extrema izquierda representa con un "*1*" el signo negativo.
- Los *2* bits restantes expresan el complemento a *1* del valor binario real.

Regla práctica para hallar el complemento de un número binario

Restringido (C1): *se invierten todos los dígitos.*

$C_{(1)}\ 0 = 1$ *el complemento restringido de 0 es 1.*

$C_{(1)}\ 11 = 00$ *el complemento restringido de 11 es 00.*

Auténtico (C2): *Para hallar el complemento a 2 se invierten todos los dígitos y se le suma 1.*

$C_{(2)}\ 11 = 00 + 1 = 01$

$C_{(2)}\ 101 = 010 + 1 = 011$

Comprobación: *1000*
– 101
011

2.7 Resumen

El sistema de números binarios ofrece muchas ventajas en el manejo interno de la información en las computadoras, respecto de la representación decimal que utilizamos los usuarios. Esto se debe en primer término, a la mayor simplicidad de diseño de una unidad aritmética binaria o a la facilidad de transferir y almacenar magnitudes físicas que representen bits. No obstante, los números decimales son esenciales para la comunicación entre el usuario y la máquina, y, por lo tanto, se justifica el aprendizaje de los métodos de conversión entre ambos sistemas.

Revisar la información textual del contenido de las largas cadenas binarias alojadas en una memoria puede resultar complicado, por lo tanto, para abreviar secuencias de ceros y unos, se utilizan los sistemas numéricos octal y hexadecimal. Esto es posible debido a que el pasaje directo entre las bases *2* y *8*, y el pasaje directo entre las bases *2* y *16* son factibles, ya que *8* es potencia exacta de *2 (2^3 es 8)* y *16* también lo es *(2^4 es 16)*; por lo tanto, en el primer caso una cadena de tres bits puede ser reemplazada por un único dígito octal, y en el segundo, una cadena de cuatro bits, por un único dígito hexadecimal. Un informático debe poder leer información hexadecimal y pensarla en binario en forma automática. Por ejemplo, una impresora "X" puede enviar el siguiente código hexadecimal, correspondiente a su estado actual "**D0**", que debe ser interpretado por el software que lee ese byte de estado como *1 1 0 1 0 0 0 0*; por lo tanto, el envío de un byte puede "programarse" para su envío o no, de acuerdo con la interpretación que se establezca para cada uno de los bits. Por ejemplo, si:

D				*0*			
1	*1*	*0*	*1*	*0*	*0*	*0*	*0*
7	*6*	*5*	*4*	*3*	*2*	*1*	*0*

El bit *7* significa encendida.

El bit *6* significa no ocupada.

El bit *5* significa sin papel.

El bit *4* significa seleccionada entre otras impresoras.

El bit *3* significa atasco de papel.

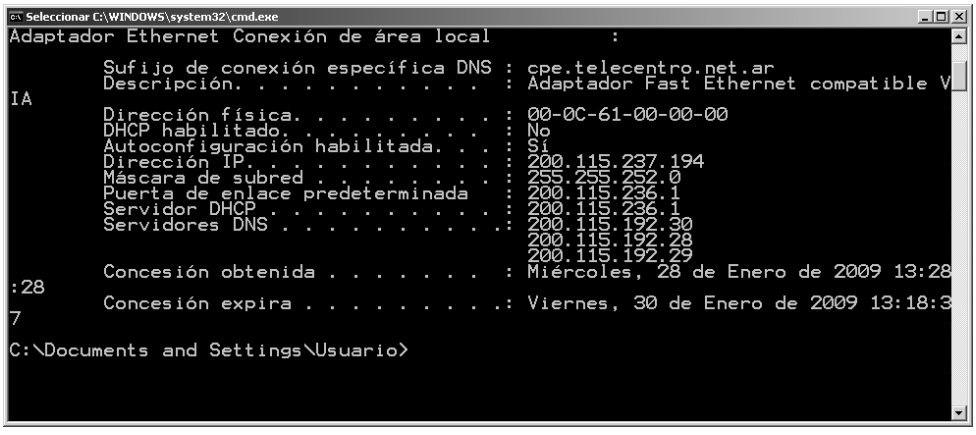
El bit *2* significa bajo nivel de tinta.

El bit *1* significa error de dispositivo.

El bit *0* significa tiempo excedido para enviar el byte.

Entonces, dada la combinación obtenida del pasaje directo de hexadecimal a binario, se interpreta que la impresora está encendida, no ocupada y seleccionada entre otras impresoras. A partir de allí el programa puede enviar el byte para imprimir, puesto que las condiciones establecidas le permiten la impresión. Por esta razón, se justifica el esfuerzo de aprender ambos sistemas numéricos y los métodos de sus conversiones a binario.

Otro ejemplo es la manera en que se utilizan ambos sistemas –el decimal y el hexadecimal– en una configuración de acceso a la red:



donde la dirección física del adaptador está indicada en hexadecimal y la dirección IP fue convertida a decimal por la propia computadora. Si esto no es así, hay que convertir los hexadecimales de esa dirección IP a decimal y obtener, de esta manera, lo que se muestra como "*200*","*115*","*237*" y "*194*" en **Dirección IP 200.115.237.194**.

Por último, el complemento de un número binario permite operar restas mediante sumas de una forma muy simple para los componentes electrónicos.

2.8 Ejercicios propuestos

1) Defina qué significan las siglas LSB y MSB.

2) Explique por qué cada dígito tiene un valor distinto y relativo a su posición en un sistema numérico de notación posicional.

3) Convierta los números binarios siguientes a sus correspondientes en sistema decimal:

   a) 10111

   b) 11111

   c) 10000001

   d) 10,1

   e) 10,101

   f) 100,011

   g) 1011,0101

   h) 1100,011

4) Convierta los números decimales siguientes a sus correspondientes en sistema binario:

   a) 55

   b) 48

   c) 204

   d) 237

   e) 255,75

   f) 10,4

   g) 83,45

5) Sin hacer ningún cálculo, convierta, si es posible, los números binarios siguientes a sus equivalentes en sistemas decimal, octal y hexadecimal:

   a) 0001

   b) 0101

   c) 1000

   d) 1011

   e) 1111

   f) 0111

   g) 10000

6) ¿La notación hexadecimal se usa ampliamente en el ámbito de la computación como método taquigráfico para representar números binarios, decimales o de ambos sistemas?

7) Convierta los números hexadecimales siguientes a sus correspondientes en sistema decimal:

   a) 7E

   b) DB

   c) 12A3

   d) 34CF

8) Convierta los números decimales siguientes a sus correspondientes en sistema hexadecimal:

   a) 48.373

   b) 217

   c) 16

   d) 3,25

   e) 101,55

9) ¿Cuántos enteros positivos se pueden expresar con K dígitos, usando números en base r?

10) Convierta "directamente", si es posible, los números siguientes. Justificar verbalmente en ambos casos.

   a) $310,10_{(16)—(2)}$

   b) $310,10_{(2)—(16)}$

   c) $310,10_{(8)—(2)}$

   d) $310,10_{(8)—(10)}$

   e) $103_{(16)—(4)}$

   f) $3,2_{(3)—(24)}$

   g) $10210_{(5)—(25)}$

11) Convierta "directamente" los números hexadecimales siguientes a sus equivalentes binarios.

a) 2EA,F65

b) 57,24

c) 3A,B

d) 10,10

e) F,003

12) Convierta los números hexadecimales siguientes a sus equivalentes binarios, pasando previamente por la base 10 (base 16 base 10 base 2), tanto los enteros como las fracciones.

a) E2

b) 3D

c) A0

d) 2,1

e) F,01

f) 1,F

g) H,A

13) Convierta:

$19,75_{(10)—(2)}$

$19,3_{(10)—(8)}$

$13,9_{(10)—(16)}$

14) Indique los números enteros anterior y posterior, en el mismo sistema:

a) $59F_{(16)}$

b) $777_{(8)}$

15) Calcule el complemento auténtico y el restringido de los números $99_{(10)}$ y $101_{(2)}$.

16) Dado el complemento auténtico siguiente, si es posible, indicar x para las bases 9, 3 y 16:

$$\begin{array}{r} 1000_{(...)} \\ -\quad x_{(...)} \\ \hline 681_{(...)} \end{array}$$

17) Deduzca cómo efectuaría la resta, por complemento a la base del sustraendo, de los números siguientes. Resuelva.

$210_{(4)} - 311_{(4)}$

18) Convierta a base 2 el número decimal 83,1. Reconviértalo a base 10, teniendo en cuenta 6 bits fraccionarios. ¿A qué conclusiones arriba respecto de la parte fraccionaria?

19) 10101001 es un número negativo expresado en complemento a 2. Convierta este número en binario positivo y su equivalente decimal y luego exprese el número negativo binario en decimal y hexadecimal.

20) Deduzca cómo representar el número negativo 10011110 con una longitud de 16 bits. Expréselo en sistemas binario y hexadecimal.

2.9 Contenido de la página Web de apoyo

El material marcado con asterisco (*) sólo está disponible para docentes.

Resumen gráfico del capítulo

Simulación

Herramienta interactiva que permite realizar conversiones y operaciones entre sistemas numéricos.

Autoevaluación

Evaluaciones Propuestas*

Presentaciones*