Capítulo 2: SISTEMAS NUMÉRICOS DE CODIFICACIÓN

2.1 Introducción

Los sistemas numéricos de codificación son técnicas utilizadas para representar textos, flujos de datos o archivos mediante números enteros, utilizando los conceptos de la teoría de números. En la actualidad, tanto la teoría de números como los sistemas numéricos de codificación tienen gran auge en el mundo de la criptografía y en la seguridad computacional.

En este capítulo estudiaremos el sistema decimal, el binario, el octal y hexadecimal, y las conversiones de un sistema a otro.

2.2 Sistema numérico decimal

El número base de este sistema es 10; los dígitos utilizados son: 0, 1, 2, 3, 4, 6, 7, 8, 9. Es el sistema utilizado por los seres humanos para la realización de operaciones aritméticas. En el aspecto técnico con este sistema pueden calcular o representar la cantidad de posiciones o de localidades de memoria que ocupa un archivo en un dispositivo de almacenamiento.

Cualquier cadena numérica escrita con estos dígitos se denomina número decimal los cuales pueden tener o no punto flotante. Con ellas se realizan las operaciones aritméticas conocidas (suma, resta, multiplicación y división, potencia y radicación) que no serán realizadas en este texto, ya que generalmente se utilizan desde los inicios académicos. Un número decimal, en particular como los siguientes puede expresarse como sigue:

$6753= 6x10^3+7x10^2+5x10^1+3x10^0$
$67.53=6x10^1+7x10^0+5x10^{-1}+3x10^{-2}$

En general: sean c_i las cifras de la parte entera y d_i las cifras de parte fraccionaria, se tiene:
$r_n r_{n-1} \ldots r_2 r_1 r_0 . d_1 d_2 d_3 \ldots d_{n-1} d_n =$
$r_n x10^n + r_{n-1} x10^{n-1} + \ldots + r_2 x10^2 + r_1 x10^1 + r_0 x10^0 + d_1 x10^{-1} + d_2 x10^{-2} + \ldots + d_{n-1} x10^{-(n-1)} + d_n x10^{-n}$

2.3 Sistema numérico binario

La base de este sistema es el número 2; los únicos dígitos utilizados son: 0,1. Para diferenciar estos números se escribe la base como subíndice; por ejemplo 1101110_2.

Cada cifra binaria se denomina bit. Cualquier cadena de bits se denomina número binario, los cuales pueden tener o no punto flotante. Con ellas se realizan las operaciones aritméticas conocidas (suma, resta, multiplicación y división, potencia y radicación) que serán realizadas en este texto.

Cualquier número decimal se puede expresar como sigue:
$1011_2= 1x2^3+0x2^2+1x2^1+1x2^0$

$10.11_2=1x2^1+0x2^0+1x2^{-1}+1x2^{-2}$

En general: sean r_i las cifras de la parte entera y d_i las cifras de parte fraccionaria, se tiene:

$r_nr_{n-1}\ldots r_2r_1r_0.d_1d_2d_3\ldots d_{n-1}d_n=$

$r_nx2^n+r_{n-1}x2^{n-1}+\ldots+r_2x2^2+r_1x2^1+r_0x2^0+d_1x2^{-1}+d_2x2^{-2}+\ldots+d_{n-1}x2^{-(n-1)}+d_nx2^{-n}$

La realización de las operaciones indicadas da como resultado un número decimal; esta es la conversión de un número binario dado decimal.

Ejemplo 2.1: convierta a decimal el numero binario 111001.1011_2 usando la tabla 2.1

Numero	1	1	1	0	0	1	.	1	0	1	1_2
exponente	5	4	3	2	1	0		-1	-2	-3	-4

Tabla 2.1:

$111001.111_2=1x2^5+1x2^4+1x2^3+0x2^2+0x2^1+1x2^0+1x2^{-1}+0x2^{-2}+1x2^{-3}+1x2^{-4}$
$=32+16+8+0+0+1+0.5+0+0.125+0.0625$
$=57.6875$

Otra forma de hacer esta conversión se da con el siguiente procedimiento:

Ejemplo 2.2: convierta a binario el número 50, 2909 y 9102

- Construya la tabla 2.2 con potencias de 2 hasta donde sea necesario (vea tabla)

13	12	11	10	9	8	7	6	5	4	3	2	1	0
8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

Tabla 2.2: tabla de potencias de 2

- Escriba el bit 1 en la celda donde encuentra las potencias con las que se puede formar el número dado y aquellas celdas que no se utilicen, se llenan con el bit 0 (vea tabla 2.3).

13	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
8192	**4096**	**2048**	**1024**	**512**	**256**	**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**
0	0	0	0	0	0	0	0	1	1	0	0	1	0
0	0	1	0	1	1	0	1	0	1	1	1	0	1
1	0	0	0	1	1	1	0	0	0	1	1	1	0

Tabla 2.3: tabla de potencias de 2 con los bits correspondientes de cada número

- El resultado obtenido corresponderá a toda la cadena de estos bits

$50=110010_2$
$2909=101101011101_2$
$9102=10001110001110_2$

2.4 Sistema numérico octal

La base de este sistema es el 8; los únicos dígitos utilizados son: 0,1, 2, 3, 4, 6, 7. Para diferenciar estos números se escribe la base como subíndice; por ejemplo 72_{8}.

Cualquier cadena numérica escrita con estos dígitos se denomina número octal los cuales pueden tener o no punto flotante. Con ellas se realizan las operaciones aritméticas conocidas (suma, resta, multiplicación y división) que serán realizadas en este texto.

Cualquier número octal queda expresado en decimal como sigue:

$6753_8 = 6x8^3+7x8^2+5x8^1+3x8^0$
$67.53_8 = 6x8^1+7x8^0+5x8^{-1}+3x8^{-2}$

En general: sean c_i las cifras de la parte entera y d_i las cifras de parte fraccionaria, se tiene:
$r_n.r_{n-1} \ldots r_2.r_1.r_0.d_1.d_2.d_3 \ldots d_{n-1}.d_n =$
$r_n x8^n+r_{n-1}x8^{n-1}+\ldots+r_2x8^2+r_1x8^1+r_0x8^0+d_1x8^{-1}+d_2x8^{-2}+\ldots+d_{n-1}x8^{-(n-1)}+d_nx8^{-n}$

2.5 Sistema numérico hexadecimal

La base de este sistema es 16; los únicos dígitos utilizados son: 0,1, 2, 3, 4, 6, 7, 8, 9, A, B, C, D, E, F, donde A es 10, B es 11, C es 12, D es 13, E es 14 y F es 15. Para diferenciar estos números se escribe la base como subíndice 16 o H; por ejemplo, $F2_{16}$ o $F2_H$.

El sistema hexadecimal tiene su aplicación en la determinación de direcciones de localidades de memoria en una microcomputadora y en la programación de computadoras en lenguaje de máquina. Las direcciones corresponden a números secuenciales e identifican cada circuito de la memoria. Algunos problemas de estas áreas se resuelven sumando o restando números hexadecimales.

Cualquier cadena numérica escrita con estas cifras se denomina número hexadecimal y puede tener o no punto flotante. Con ellas se realizan las operaciones aritméticas conocidas (suma, resta, multiplicación y división, potencia y radicación) que serán tratadas en este texto.

Cualquier número hexadecimal queda expresado en decimal, así:

$D7F3_{16} = Dx16^3+7x16^2+Fx16^1+3x16^0$
$D7.F3_{16} = Dx16^1+7x16^0+Fx16^{-1}+3x16^{-2}$

En general: sean c_i las cifras de la parte entera y d_i las cifras de parte fraccionaria, se tiene:

$r_n r_{n-1} \ldots r_2 r_1 r_0.d_1 d_2 d_3 \ldots d_{n-1} d_n =$
$r_n x16^n+r_{n-1}x16^{n-1}+\ldots+r_2x16^2+r_1x16^1+r_0x16^0+d_1x16^{-1}+d_2x16^{-2}+\ldots+d_{n-1}x16^{-(n-1)}+d_nx16^{-n}$

2.6 Aritmética binaria

Las operaciones aritméticas de una computadora o una calculadora se procesan en la ALU (Arithmetic Logic Unit) del microprocesador. Allí se combinan compuertas lógicas y otros dispositivos de manera que puedan realizar las operaciones aritméticas a grandes velocidades (en menos de un microsegundo). Las operaciones binarias aritméticas fundamentales son similares las realizadas con números decimales, pero teniendo en cuenta que el acarreo en la suma o lo que presta en la resta corresponde a la base del número.

2.6.1 Suma binaria

La adición binaria utiliza el mismo algoritmo que utilizan los números decimales; pero debe tener en cuenta la anotación hecha en la sección acerca del acarreo. (ver tabla 2.4)

+	**0**	**1**
0	0	1
1	1	0, acarreo= 1

Tabla 2.4: tabla de la suma

Ejemplo 2.3: Efectúe

a) $110011_2+11010_2+1100111_2$

Acarreos	**1**	**1**	**1**	**1**	**1**	**1**	**1**	
						1		
			1	1	0	0	1	1
				1	1	0	1	0
		1	1	0	0	1	1	1
	1	0		1	0	1	0	0

b) $11_2+111_2+1111_2+11111_2+111111_2+1_2$

Acarreos	**1**	**1**	**1**	**1**	**1**	**1**	**1**	
		1	**1**	**1**	**1**	**1**	**1**	
			1	**1**	**1**	**1**	**1**	
					1	**1**		
							1	1
						1	1	1
					1	1	1	1
				1	1	1	1	1
			1	1	1	1	1	1
								1
	1	0	1	1	1	0	0	0

c) $1.1_2+11.1_2+111.1_2+11.111_2+1111.11_2+1_2$

Acarreos	**1**	**1**	**1**	**1**	**1**	**1**		**1**		
		1	**1**	**1**	**1**	**1**				
			1	**1**	**1**	**1**				
				1	**1**					
						1	**.**	1	0	0

$$
\begin{array}{cccccccccc}
 & & & & 1 & 1 & . & 1 & 0 & 0 \\
 & & & 1 & 1 & 1 & . & 1 & 0 & 0 \\
 & & & & 1 & 1 & . & 1 & 1 & 1 \\
 & & 1 & 1 & 1 & 1 & . & 1 & 1 & 0 \\
 & & & & & 1 & . & 0 & 0 & 0 \\
\hline
1 & 1 & 0 & 0 & 0 & 1 & . & 0 & 0 & 1_2
\end{array}
$$

2.6.2 Sustracción binaria

La adición binaria se realiza de igual manera que con los números decimales; pero debe tener en cuenta la anotación hecha en la sección acerca del préstamo binario. Para restar números binarios se pueden tener dos casos:

Caso 1: Resta sin signo. En este caso utilice la tabla 2.5

-	**0**	**1**
0	0	1, presta 1
1	1	0

Tabla 2.5: Tabla de la resta

Ejemplo 2.10: efectúe las siguientes operaciones

a) 111001_2-1101_2
b) 1000011_2-111111_2

Solución:

a) 111001_2-1101_2

Presta

$$
\begin{array}{cccccc}
 & \mathbf{0} & \mathbf{2} & \mathbf{2} & & \\
 & & \mathbf{\cancel{0}} & & & \\
1 & \cancel{1} & \cancel{1} & \cancel{0} & 0 & 1 \\
 & & 1 & 1 & 0 & 1 \\
\hline
1 & 0 & 1 & 1 & 0 & 0
\end{array}
$$

b) 1000011_2-111111_2

Presta

$$
\begin{array}{ccccccc}
\mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{2} & & \\
 & \mathbf{\cancel{2}} & \mathbf{\cancel{2}} & \mathbf{\cancel{2}} & & & \\
1 & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & 1 & 1 \\
 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
0 & 0 & 0 & 0 & 1 & 0 & 0
\end{array}
$$

Caso 2: Sustracción con signo. En este caso puede convertir la resta a suma, así:

$$a-b=a+(-b)$$

Luego aplique el concepto del máximo número de bits que deben tener los números, que generalmente se logra haciendo la suma y calculando el intervalo de valores.

Complementos binarios. Se conocen dos tipos de complemento binario: complemento a 1 y complemento a 2 y que son de gran utilidad para la realización de operaciones binarias con signo.

En los sistemas digitales se usan frecuentemente los Flip-Flops que son elementos de memoria de 1 bit (binary digit) conformados por componentes lógicos combinatorios conectados en una configuración con retroalimentación. Los Flip-Flops son circuitos electrónicos que alternan entre dos estados; pueden mantener un estado de salida después de desaparecer la condición de entrada que produjo ese estado. Los sistemas digitales, en particular las computadoras, almacenan los números binarios en Flip-Flops para almacenar el signo de estos números; en efecto, se introduce a la izquierda del binario un bit adicional, así:

- almacenará 0, si el número es positivo,.
- almacenará 1, si el número es negativo,.

Complemento a 1. Permite cambiar los bits o cifras binarias, así: 0 por 1 y 1 por 0.

Ejemplo 2.4: el complemento a 1 de 1011100101_2 es 0100011010_2

Complemento a 2. Consiste en sumar 1 a la cifra más significativa del número binario obtenido en complemento a 1

Ejemplo 2.5: el complemento a 2 de 1011100101_2 es

Número	:	1	0	1	1	1	0	0	1	0	1
Complemento 1:		0	1	0	0	0	1	1	0	1	0
Adiciona 1	:										1
Complemento 2:		0	1	0	0	0	1	1	0	1	1

El complemento a 2 es la técnica más utilizada para representar números binarios con signo, permitiendo convertir la sustracción en una adición. Lo anterior, hace que el sistema digital utilice solamente un circuito para realizar ambas operaciones.

Ejemplo 2.6: exprese con 6 bits más el signo a los siguientes números

a) +25
b) –25

Solución:

a) Proceda así:
   - Convierta a binario el 25. En efecto, $25= 11001_2$
   - Complete los bits pedidos y agregue el bit del signo; por consiguiente el numero quedará como sigue: 0011001_2

b) Como el número es negativo, proceda como en el caso anterior, pero hallando complemento a 2 a ese resultado, así:
   $+25=0011001_2$
   $\Rightarrow$ complemento 1: 1100110_2.
   $\Rightarrow$ complemento 2: 1100111_2
   Por lo tanto, $-25=1100111_2$

Nota: el intervalo de valores que debe tener el número binario del resultado de una operación con signo va desde -2^n hasta $+(2^n -1)$ con n= número de bits. Es de tal manera que quienes crean los compiladores tienen que tener en cuenta tal intervalo.

Ejemplo 2.7: ¿cuál es la cantidad de bits que tienen los números enteros de un lenguaje de programación, dado que su intervalo es –32768 hasta 32767?

$$-32768=-2^n \Leftrightarrow -2^{15}=-2^n$$
$$\Leftrightarrow n=15 \text{ bits sin signo}$$

Por lo tanto, en algunos lenguajes de programación los números enteros con signo se escriben con máximo 16 bits.

Ejemplo 2.8: exprese con signo +49, -49, +25 y -25

Solución:

El máximo número entero que se puede formar con estos números es 74 que se encuentra entre $64=2^6$ y $128=6^7$. Luego, se necesitan 7 bits para expresar a dichos enteros.
$49=110001_2 \Rightarrow +49=00110001_2$ (1)

Si se quiere obtener –49, en efecto se tendría que hallar complemento a 2 al resultado (1):

$-49=11001111_2$ (2)

Ahora, $25=0011001_2 \Rightarrow +25=00011001_2$ (3)

Similarmente, para hallar el binario de –25 se toma el resultado (3). Por consiguiente quedaría:

$-25=11100111_2$ (4)

Ejemplo 2.10: efectúe las siguientes operaciones con signo

a) $110010_2-11000_2$
b) +25+49
c) 49-25
d) 25-49
e) –25-49

Solución:
a) $110010_2-11000_2 = 110010_2+(-11000_2)$
Convierta -11000 a complemento a dos para lo cual se debe igualar la cantidad de bits 0011000_2 para hallar el binario con signo $\Rightarrow 1101000_2$.
El signo de $110010_2 \Rightarrow 0110010_2$
$=0110010_2+(1101000_2)$
$=10011010_2$
$=0011010_2$ descartando el acarreo del signo

b) +25+49 $=00011001_2+00110001_2$
$=01001010$
$= +74$

c) 49-25 $=49+(-25)$
$=00110001_2+11100111_2$
$=100011000_2$
$=00011000_2$
$= +24$

d) 25-49 $=25+(-49)$
$=00011001_2+11001111_2$
$=11101000_2$
$= -(0011001_2)$ hallando complemento a 2, porque el resultado es negativo
$= -24$

e) -25+(-49) $=11100111_2+11001111_2$
$=110110110$ el primer bit (el de la izquierda) se descarta, porque corresponde a un acarreo de signo
$=10110110_2$
$= -(0110110_2)$
$= -(1001010_2)$ hallando complemento a 2, porque el resultado es negativo
$= -74$

2.6.3 Multiplicación binaria

Para multiplicar binarios se realiza el mismo procedimiento que con números enteros. Para tal fin, a continuación se presenta la tabla 2.6 de la multiplicación que servirá de base para efectuar dicha operación.

*	**0**	**1**
0	0	0
1	0	1

Tabla 2.6: tabla de la multiplicación

Ejemplo 2.11: efectúe a)$10111_2 x 101_2$ b)$11.001_2 x 1010_2$
Solución:

10111_2	11.001_2
$x101_2$	$x1010_2$
10111	11001
10111	11001
1110011	11111.010

a)$10111_2 x 101_2 = 1110011_2$

b)$11.001_2 x 1010_2 = 11111.01_2$

2.6.4 División binaria

Para dividir binarios se realiza el mismo procedimiento que con números enteros. Para tal fin, realice la división utilizando el método de resta, en la que a la parte del dividendo se sustrae el producto de las veces que cabe divisor en la parte del residuo por el divisor.

Ejemplo 2.12: efectúe a)$1110011_2 \div 101_2$ b) $10011.01_2 \div 1010_2$
Solución:

$1110011_2 \mid 101_2$

-101 | 10111
01000
-101
00111
-101
0101
-101
000

$10011.01_2 \mid 1010_2$

-1010 | 1.111011001
010010
-1010
10001
-1010
001110
-1010
010000
-1010
0011000
-1010
0010000
-1010
1100

a) $1110011_2 \div 101_2 = 10111_2$

b) $10011.01_2 \div 1010_2 = 1.11101(1001)_2$

La parte que va en paréntesis $(1001)_2$ indica que se repite; es decir, corresponde a un decimal periódico.

2.7 Conversión de números de un sistema a otro

2.7.1 Conversión de número decimal a binario

Para convertir un número decimal sin fracción decimal a binario se pueden utilizar 2 técnicas:

Técnica 1: divisiones sucesivas por 2

En efecto, se divide el número sucesivamente por 2 hasta que el cociente sea cero y a partir del último residuo hasta último se van escribiendo estos residuos y este será el número binario.

Si el número tiene fracción decimal tome la parte entera y proceda como en el caso anterior; luego escriba el punto que separa la parte entera de la fracción binaria. Ahora proceda con la parte fraccionaria, así: multiplique este parte sucesivamente por 2 y vaya escribiendo la cifra obtenida en la parte entera, dejando siempre en cero a dicha parte, antes de volver a multiplicar. Continúe este proceso hasta que la parte fraccionaria sea cero. El resultado definitivo se obtiene escribiendo el resultado obtenido de la parte entera y a continuación los resultados de la parte fraccionaria (las partes enteras de los productos).

Ejemplo 2.13: convierta a binario los números
a) 47
b) 47.375.

Solución:

Apliquemos el algoritmo de Euclides $a=q*b+r$, con dividendo y b divisor, para realizar las divisiones y guardar los residuos:

a) $47=23x2+1 \Rightarrow r=1$ ↑
$23=11x2+1 \Rightarrow r=1$
$11=5x2+1 \Rightarrow r=1$
$5=2x2+1 \Rightarrow r=1$
$2=1x2+0 \Rightarrow r=0$
$1=0x2+1 \Rightarrow r=1$
$\Rightarrow 47=111101_2$

b) $0.375x2=0.75 \Rightarrow d=0$ ↓
$0.75x2=1.5 \Rightarrow d=1$
$0.5x2=1.0 \Rightarrow d=1$
$\Rightarrow 47.375=101111.011_2$

Las cifras se escriben en el sentido de las flechas.

Técnica 2: tabla de potencias de 2

Ejemplo 2.14: escriba el número binario correspondiente al decimal 2461. Escriba 1 en la potencia de 2 que puede utilizar para construir el número (ver tabla 2.7).

11	10	9	8	7	6	5	4	3	2	1	0
2048	1024	512	256	128	64	32	16	8	4	2	1
1	0	0	1	1	0	0	1	1	1	0	1

Tabla 7: ejemplo 2.14

Ponga 1 en la casilla del 2048, 256, 128, 16, 8, 4 y en 1. En efecto, el número será la suma de las potencias de 2.

Por consiguiente $2461=100110011101_2$

2.7.2 Conversión de número decimal a octal

Para convertir un número decimal sin fracción decimal a octal, se divide el número sucesivamente por 8 hasta que el cociente sea cero y a partir del último residuo se van escribiendo estos residuos; la cadena resultante será el número octal.

Si el número tiene fracción decimal tome la parte entera y proceda como en el caso anterior; luego escriba el punto que separa la parte entera de la fracción octal. Ahora proceda con la parte fraccionaria, así: multiplique este parte sucesivamente por 8 y vaya escribiendo la cifra obtenida en la parte entera, dejando siempre en cero a dicha parte, antes de volver a multiplicar. Continúe este proceso hasta que la parte fraccionaria sea cero. El resultado definitivo se obtiene escribiendo el resultado obtenido de la parte entera y a continuación los resultados de la parte fraccionaria (las partes enteras de los productos).

Ejemplo 2.13: 348 a octal

a) $348=43x8+4 \Rightarrow r=4$ ↑
$43=5x8+3 \Rightarrow r=3$ ↑
$5=0x8+5 \Rightarrow r=5$ ↑
$\Rightarrow 347=534_2$

b) $0.015625x8=0.125 \Rightarrow d=0$ ↓
$0.125x8=1.0 \Rightarrow d=1$ ↓

$\Rightarrow 0.015625=0.01_2$

Las cifras se escriben en el sentido de las flechas.

2.7.3 Conversión de número octal a binario

Para convertir un número octal a binario se toma cada cifra octal se convierte a binario, aplicando la técnica dada en la sección 8.8.2 y luego se escriba la cadena de binarios.

Ejemplo 2.14: convierta a binario a) 763_8 y b) 67.54_8

Solución:

7			6			3		
1	1	1	1	1	0	0	1	1

$763_8=111110011_2$

6	7.	5	4
1 1 0	1 1 1.	1 0 1	1 0 0

$67.54_8=110111.101100_2=110111.1011_2$

2.7.4 Conversión de número binario a octal

Para convertir un número binario a octal se reparten las cifras binarias en grupos de tres cifras binarias, así: parte entera se toma de derecha a izquierda; en caso que no se hayan suficientes cifras para formar esos grupos, se completa con ceros a la izquierda el último grupo; la parte fraccionaria se toma de izquierda a derecha; en caso tal que no tenga suficientes cifras para dichos grupos, se completa el último grupo con ceros a la derecha.

Ejemplo 2.15: convierta a binario a) 111100101_2 b) 1010111.0011_2

Solución:

1 1 1	1 0 0	1 0 1
7	4	5

$111100101_2=745_8$

0 0 1	0 1 0	1 1 1.	0 0 1	1 0 0
1	2	7.	1	4

$1010111.0011_2=127.14_8$

2.7.5 Conversión de número decimal a hexadecimal

Para convertir un número decimal sin fracción decimal a hexadecimal, se divide el número sucesivamente por 16 hasta que el cociente sea cero y a partir del último residuo hasta último se van escribiendo estos residuos y este será el número hexadecimal.

Si el número tiene fracción decimal tome la parte entera y proceda como en el caso anterior; luego escriba el punto que separa la parte entera de la fracción hexadecimal. Ahora proceda con la parte fraccionaria, así: multiplique este parte sucesivamente por 16 y vaya escribiendo la cifra obtenida en la parte entera, dejando siempre en cero a dicha parte, antes de volver a multiplicar. Continúe este proceso hasta que la parte fraccionaria sea cero. El resultado definitivo se obtiene escribiendo el resultado obtenido de la parte entera y a continuación los resultados de la parte fraccionaria (las partes enteras de los productos).

Ejemplo 2.16: 12127 a hexadecimal

a) $12127 = 757 \times 16 + 15 \Rightarrow r=15=F$

$757 = 47 \times 16 + 5 \Rightarrow r=5$

$47 = 2 \times 16 + 15 \Rightarrow r=15=F$

2=0x16+2 ⇒ r=2

⇒ 12127=2F5F$_H$

b) 0.000244140625x16=0.00390625 ⇒ d=0

0.00390625x16=0.0625 ⇒ d=0

0.0625x16=1.0 ⇒ d=1

⇒ 0.000244140625=0.001$_H$

Las cifras se escriben en el sentido de las flechas.

2.7.6 Conversión de número hexadecimal a binario

Para convertir un número hexadecimal a binario se toma cada cifra hexadecimal y se convierte a binario, aplicando la técnica ofrecida en la sección 8.8.2 y luego se escriba la cadena de binarios.

Ejemplo 2.17: convierta a binario a) B6$_H$ y b) A7.5D$_H$

Solución:

B				6			
1	0	1	1	0	1	1	0

B6$_H$=10110110$_2$

A				7**.**				5				D			
1	0	1	0	0	1	1	1**.**	0	1	0	1	1	1	0	1

A7**.**5D$_H$= 10100111.01011101$_2$

2.7.7 Conversión de número binario a hexadecimal

Para convertir un número binario a hexadecimal se agrupan de a cuatro cifras binarias de derecha a izquierda en parte entera binaria y de izquierda a derecha, si tiene parte fraccionaria. Si no se tienen suficientes cifras para formar esos grupos se completa con ceros, a la izquierda en su parte entera y a la derecha en su parte fraccionaria.

Ejemplo 2.18: convierta a hexadecimal a) 10011110$_2$ b) 111011.1101011$_2$

1	0	0	1	1	1	1	0
9				E			

10011110$_2$=9E$_H$

0	0	1	1	1	0	1	1**.**	1	1	0	1	0	1	1	0
3				B**.**				D				6			

111011.1101011$_2$=3B**.**D6**$_H$**

2.8 Aritmética octal y hexadecimal

Las diferentes operaciones aritméticas se pueden realizar en el sistema octal y en sistema hexadecimal de igual manera que en el sistema decimal o que en el sistema binario; basta con convertir el número octal o hexadecimal a binario o al decimal y,

posteriormente, se efectúa la operación en el sistema intermediario (binario o decimal).

Ejemplo 2.19: Un dispositivo tiene disponible desde la dirección $D5F_H$; se quiere almacenar un archivo que ocuparía en octal 6723_8 posiciones y se almacena luego otro archivo que ocuparía 1335 posiciones. ¿Es posible realizar esas operaciones en tal dispositivo? ¿En caso afirmativo, a partir de cuál posición quedó espacio disponible? ¿En caso negativo cuántas posiciones quedaron faltando?

Veamos. Efectúe la operación $D5F_H$ - 6723_8 -1335 y exprese el resultado en binario, hexadecimal y finalmente decimal.

1 1 0 1	0 1 0 1	1 1 1 1
D	5	F

$D5F_H = 110101011111_2$

1 1 0	1 1 1	0 1 0	0 1 1
6	7	2	3

$6723_8 = 110111010011_2$

$-6723_8 = 11001000101101_2$ con 14 bits incluyendo el signo, ¿por qué?

$1335 = 10100110111_2$

-1335=11101011001001 con 14 bits incluyendo el signo, ¿por qué?

	0	0	1	1	0	1	0	1	0	1	1	1	1	1
	1	1	0	0	1	0	0	0	1	0	1	1	0	1
	1	1	1	0	1	0	1	1	0	0	1	0	0	1
~~1~~	1	1	1	0	1	0	0	1	0	1	0	1	0	1

Por lo tanto, se descarta el acarreo del signo y se realiza complemento 2 a la parte restante al signo (bit 14), porque el resultado del número es negativo.

En efecto, el resultado es:

$$- \quad 0010110101011_2 = -2653_8 = -5AB_H = -1451$$

Efectivamente, como el resultado es negativo, significa que no cupieron los archivos, pues quedaron faltando 1451 posiciones.

2.9 Sistemas de codificación computacional

Una sucesión de bits conforman un número binario y corresponden a lo que se denomina “codificación” de un dato. Este código puede ser procesado por el computador.

¿Qué pueden codificar los computadores? Pueden codificar lo siguiente:

- **Las instrucciones** que conforman un programa, las cuales son fijadas o determinadas por el fabricante y no pueden ser modificadas por el usuario; además, no tiene por qué hacerlo.

- **Los datos** que son de dos tipos: datos numéricos y los datos alfanuméricos. Dichas codificaciones se utilizan dependiendo del computador que se tenga, las cuales pueden tener 2^n combinaciones con n= cantidad de bits.

Los datos alfanuméricos (caracteres literales: 26 letras minúsculas y 26 letras mayúsculas y, 26 caracteres especiales); además, los 10 dígitos tienen que codificarse para que sean representados en el computador.

El estándar de las máquinas de computación trabaja con 8 bits (códigos de 7 bits que producen 128 combinaciones para representar cualquier carácter y 1 bit de paridad[1] que ayuda a las transmisiones de datos).

[1] Cuando se transmiten datos mediante el computador se suman los bits del carácter enviado; si la suma no coincide con el bit de verificación, según la paridad del computador (par o impar),

Se conocen varios sistemas de codificación que se utilizan para intercambiar información, entre otros son ellos: BCD, EBCDIC y ASCII.

	000 0	001 1	010 2	011 3	100 4	101 5	110 6	111 7
0000 0	0	16	- 32	48	SP 64	80	96	& 112
0001 1	1 1	17	33	A 49	65	/ 81	J 97	113
0010 2	2 2	18	34	B 50	66	S 82	K 98	114
0011 3	3	T 19	L 35	51	3 67	83	99	C 115
0100 4	4	20	36	D 52	4 68	U 84	M 100	116
0101 5	5	V 21	N 37	53	5 69	85	101	E 117
0110 6	6	W 22	O 38	54	6 70	86	102	F 118
0111 7	7 7	23	39	G 55	71	X 87	P 103	119
1000 8	8 8	24	40	H 56	72	Y 88	Q 104	120
1001 9	9	Z 25	R 41	57	9 73	89	105	I 121
1010 A	10	26	! 42	58	0 74	90	106	? 122
1011 B	# 11	27	43	. 59	75	, 91	$ 107	123
1100 C	12	% 28	44	60	' 76	92	108	) 124
1101 D	: 13	29	45	| 61	77	~ 93	109	125
1110 E	> 14	30	46	< 62	@ 78	94	; 110	126
1111 F	15	31	Δ 47	63	79	95	111	* 127

Tabla 2.7: tabla de código BCD de 7 bits

2.9.1 Sistema de codificación EBCDIC

Es una sigla que proviene del inglés Extended Binary Coded Decimal Interchange Code (Código de Intercambio Extendido Decimal Codificado en Binario). Es el código de los computadores IBM o compatibles con IBM como extensión del código BCD. Este sistema de codificación fue desarrollado por la IBM con un código de 8 bits (256 combinaciones) que representan caracteres alfanuméricos y los dígitos decimales dentro de 1 byte (8 bits). Los 4 primeros bits corresponden a los bits de zona que cambian así: la letras A-I con 1100, J-R con 1101, S-Z con 1110; los dígitos decimales con 1111 y, los caracteres especiales con diferente representación (0100, 0101, 0110, 0111). Los otros bits corresponden al código propio del carácter o dígito.

	1000	1001	1010	1011	1100	1101	1110	1111
0000 0					{ 192	} 208	\ 224	**0** 240
0001 1	**a** 129	**j** 145			**A** 193	**J** 209		**1** 241
0010 2	**b** 130	**k** 146	**s** 162		**B** 194	**K** 210	**S** 226	**2** 242
0011 3	**c** 131	**l** 147	**t** 163		**C** 195	**L** 211	**T** 227	**3** 243
0100 4	**d** 132	**m** 148	**u** 164		**D** 196	**M** 212	**U** 228	**4** 244
0101 5	**e** 133	**n** 149	**v** 165		**E** 197	**N** 213	**V** 229	**5** 245
0110 6	**f** 134	**o** 150	**w** 166		**F** 198	**O** 214	**W** 230	**6** 246
0111 7	**g** 135	**p** 151	**x** 167		**G** 199	**P** 215	**X** 231	**7** 247
1000 8	**h** 136	**q** 152	**y** 168		**H** 200	**Q** 216	**Y** 232	**8** 248
1001 9	**i** 137	**r** 153	**z** 169		**I** 201	**R** 217	**Z** 233	**9** 249

Tabla 2.8: tabla de codificación EBCDIC

Ejemplo 2.20: Escriba en código EBCDIC la palabra ADIOS. En efecto, el código de cada letra es: A $\Rightarrow$11000001, D $\Rightarrow$11000100, I $\Rightarrow$11001001, O $\Rightarrow$11010110 y S $\Rightarrow$11100010. Luego, palabra ADIOS codificada en EBCDIC es:

11000001110001001100100111010110111100010

2.9.2 Sistema de codificación ASCII

Es una sigla que proviene del inglés American Standard Code for Information Interchange (Código Estándar Americano para el Intercambio de Información). Se pronuncia Ask-i). Este sistema fue desarrollado para sistemas de computación que no son de la IBM. Corresponde al código alfanumérico más utilizado en los diferentes sistemas de los microprocesadores.

Al igual que código EBCDIC, también tiene los bits de zona: 0101, para lo dígitos; 1010, para las letras A-O; 1011 para las demás letras y, 0010, para algunos caracteres especiales.

	000 0	001 1	010 2	011 3	100 4	101 5	110 6	111 7
0000 0	**NUL** 0	**DLE** 16	**SP** 32	**0** 48	**@** 64	**P** 80	**`** 96	**p** 112
0001 1	**SOH** 1	**DC1** 17	**!** 33	**1** 49	**A** 65	**Q** 81	**a** 97	**q** 113
0010 2	**STX** 2	**DC2** 18	**“** 34	**2** 50	**B** 66	**R** 82	**b** 98	**r** 114
0011 3	**ETX** 3	**DC3** 19	**#** 35	**3** 51	**C** 67	**S** 83	**c** 99	**s** 115
0100 4	**EOT** 4	**DC4** 20	**$** 36	**4** 52	**D** 68	**T** 84	**d** 100	**t** 116
0101 5	**ENQ** 5	**NAK** 21	**%** 37	**5** 53	**E** 69	**U** 85	**e** 101	**u** 117
0110 6	**ACK** 6	**SYN** 22	**&** 38	**6** 54	**F** 70	**V** 86	**f** 102	**v** 118
0111 7	**BEL** 7	**ETB** 23	**’** 39	**7** 55	**G** 71	**W** 87	**g** 103	**w** 119
1000 8	**BS** 8	**CAN** 24	**(** 40	**8** 56	**H** 72	**X** 88	**h** 104	**x** 120
1001 9	**HT** 9	**EM** 25	**)** 41	**9** 57	**I** 73	**Y** 89	**i** 105	**y** 121
1010 A	**LF** 10	**SUB** 26	***** 42	**:** 58	**J** 74	**Z** 90	**j** 106	**z** 122
1011 B	**VT** 11	**ESC** 27	**+** 43	**;** 59	**K** 75	**[** 91	**k** 107	**{** 123
1100 C	**FF** 12	**FS** 28	**´** 44	**<** 60	**L** 76	**** 92	**l** 108	**|** 124
1101 D	**CR** 13	**GS** 29	**-** 45	**=** 61	**M** 77	**]** 93	**m** 109	**}** 125
1110 E	**SO** 14	**RS** 30	**.** 46	**>** 62	**N** 78	**^** 94	**n** 110	**~** 126
1111 F	**SI** 15	**US** 31	**/** 47	**?** 63	**O** 79	**_** 95	**o** 111	**DEL** 127

Tabla 2.9: tabla de código ASCII

Ejemplo 2.19: Escriba en código ASCII la frase, 15 LOBAS.

En efecto, el código quedaría:
00110001 00110101 00100000 01001100 01001111 01000010 01000001 01010011

AUTOEVALUACION 2

1. SELECCIÓN MÚLTIPLE DE MÚLTIPLE RESPUESTA

Resuelva los problemas 1.1 a 1.4 y seleccione la respuesta correcta según las siguientes situaciones:

A) Si 1 y 2 son correctas
B) Si 2 y 3 son correctas
C) Si 3 y 4 son correctas
D) Si 2 y 4 son correctas
E) Si 1 y 3 son correctas

1.1 El límite inferior y el superior del intervalo de números enteros decimales con signo que se puede utilizar con 10 bits, incluyendo el signo, es:

1. -512
2. +512
3. +511
4. -511 Respuesta: ____

1.2 Al efectuar la sustracción con signo de $-1110001_2-1111111_2$ se obtiene:

1. 11110000_2
2. 100010000_2
3. -11110000_2
4. -00010000_2 Respuesta: ____

1.3 El resultado de efectuar con signo la operación indicada $-2A_H + 51_8$ es:

1. -1_2
2. 1111111111_2
3. -000101010_2
4. 000101001_2 Respuesta: ____

1.4 La realización con signo de las operaciones indicadas $3AF_{(H)} - 3172_{(8)} - 1001011010_{(2)}$ expresadas la respuesta en binario y en decimal es:

1) -1317
2) $1010100100101_{(2)}$
3) $1101011011011_{(2)}$
4) -6875 Respuesta: ____

2. Un registro es almacenado en la dirección $1CBA_{(H)}$. Determine el número de su localidad. R/ 7354

3. ¿Cuántos bits se necesitan para representar los números enteros decimales entre –8 y +7 es
   A) 3 B) 4 C) 5 D) 6

4. ¿Cuántos bits se necesitan para representar los números enteros decimales entre –65 y +111? R/ desde –64, se requieren 7 bits para números negativos (incluyendo el signo) y hasta el +111, se requieren

A) 6 bits para los números positivos (incluyendo el signo).
B) 7 bits para los números positivos (incluyendo el signo).
C) 8 bits para los números positivos (incluyendo el signo).
D) 9 bits para los números positivos (incluyendo el signo).

5. ¿Cuántos bits se necesitan para representar los números enteros decimales entre 0 y 999?
   A) 7 bits B) 8 bits C) 9 bits D) 10 bits

6. Un dispositivo tiene espacio libre a partir de $DFEC_{(H)}$, Se sabe que un archivo requiere de 12315 localidades de memoria. La posición del dispositivo a partir de la cual queda espacio disponible es
   A) 11007 B) 69639 C) $69639_{(H)}$ D) $11007_{(H)}$

TALLER

1. Convierta a octal:
   a. $11111.011_{(2)}$ b. $1111.11_{(2)}$ c. $F0CA_{(H)}$
   d. $A.FA_{(H)}$ e. $B0CA_{(H)}$ f. $10001.0001_{(2)}$

2. Convierta a hexadecimal
   a. $77.77_{(8)}$ b. $1110001.001_{(2)}$
   c. $1110000101.1_{(2)}$ d. $1234.567_{(8)}$

2. Efectúe las siguientes operaciones con signo como lo haría un computador:
   a. $3_{(8)}-4.7_{(8)}+10101.01_{(8)}-365.3_{(8)}+4326.05_{(8)}$
   b. $4A.37_{(H)}+5ABD_{(H)}-49.7AD_{(H)}+0.4006A_{(H)}$
   c. $111.01_{(8)}+1111.11_{(8)}-1001.001_{(8)}+11.01_{(8)}-1111.0111_{(8)}+1.1_{(8)}+011111_{(8)}$
   d. $111.01_{(2)}+1111.11_{(2)}-1001.001_{(2)}+11.01_{(2)}-1111.0111_{(2)}+1.1_{(2)}+011111_{(2)}$
   e. $-504.31_{(8)}+36.2_{(8)}+3.4_{(8)}-5.43_{(8)}+471.3_{(8)}+4.4_{(8)}-7.77_{(8)}+40.3_{(8)}$

3. Realice, dando la respuesta en decimal:
   $F1.06_{(H)}+A.FA_{(H)}-11.011_{(8)}-100010.011_{(2)}$
   $777.77_{(8)}-77.77_{(H)}-111111.0101_{(2)}$

4. Determine el valor de:
   a. $A.3AA_{(H)}*10001.2_{(H)}$
   b. $77.7_{(8)}*123.45_{(8)}$
   c. $77.7_{(8)}*123.45_{(8)}$
   d. $6.51_{(8)}*3.F_{(H)}-4.1_{(H)}*11111.011_{(2)}+3.47_{(8)}$

5. Calcule el cociente (con 4 cifras en la parte fraccionaria) y el residuo de las siguientes divisiones:
   a. $11010110.1_{(2)} \div 11.11_{(2)}$
   b. $176.42_{(8)} \div 3.64_{(8)}$
   c. $A5B8.D_{(H)} \div B9E_{(H)}$
   d. $(7 \oplus 3) \otimes 72$ en $\mathbb{Z}_{72}$

e. $(1001_2 \otimes 11_2) \oslash (100011_2)$ (si existe en $\mathbb{Z}_{72}$)

f. $13_8 \otimes (111_2 \oslash F_H)$ (si existe en $\mathbb{Z}_{72}$)

g. $(1000_2 \oslash 12_H) \oplus (33_8)$ (si existe en $\mathbb{Z}_{72}$)

Resuelva los siguientes problemas, teniendo en cuenta que la dirección de las localidades de memoria de una microcomputadora se especifica en hexadecimal. Dichas direcciones son números secuenciales que identifican cada circuito de la memoria:

7. Una microcomputadora en particular puede almacenar un número de 8 bits en cada localidad de la memoria. Si las direcciones de la memoria van de $0000_{(H)}$ a $FFFF_{(H)}$, ¿cuántas localidades de memoria tiene? R/ En total 65535 localidades
8. Se especifica que otra computadora tiene 4096 localidades de memoria, ¿qué intervalo de direcciones hexadecimales utiliza dicha computadora? R/ Va desde $000_{(H)}$ hasta $FFF_{(H)}$
9. ¿Cuál es el intervalo de números decimales con signo que se puede utilizar con 10 bits, incluyendo el signo? R/ Entre -512 y +511

10. Para convertir números binarios a decimal, los números se representan en complemento a dos. Convierta a decimal utilizando complemento 2 los números:

a. $10001_{(2)}$ b. $10101010_{(2)}$ c. $10010011_{(2)}$ d. $1010010001_{(2)}$

11. Represente cada uno de los siguientes valores como un número de 6 bits con signo en el sistema complemento a dos:

a. +10 b. –13 c. +5 d. –15 e. +20 f. –30 g. +17

12. Realice las siguientes operaciones, dando la respuesta con signo, tal como la hace computadora y determine el resultado que se mostraría en la pantalla:

a. 15 +23 b. 30 –14 c. 18 – 34 d. –34 – 28 e. 35 - 67

ANALISIS DE RELACIÓN

En los numerales 8.1 hasta 8.3, seleccione la opción correcta, así:

A) Si la afirmación y la razón son VERDADERAS y la razón es una explicación CORRECTA de la afirmación
B) Si la afirmación y la razón son VERDADERAS, pero la razón NO es una explicación CORRECTA de la afirmación
C) Si la afirmación es VERDADERA, pero la razón es una proposición FALSA
D) Si la afirmación es FALSA, pero la razón es una proposición VERDADERA

Si tanto la afirmación como la razón son proposiciones FALSAS

12.1 Una computadora que tenga 4096 localidades de memoria utiliza un intervalo de direcciones hexadecimales que va desde $000_{(H)}$ hasta $FFF_{(H)}$ PORQUE Un intervalo que va desde $000_{(H)}$ hasta $FFF_{(H)}$ tiene 4095 posiciones más una posición del signo.

Respuesta: ____

12.2 Una variable declarada en un lenguaje de programación como un número entero define a números entre -32768 y +32767 representados en la computadora por 15 bits incluyendo el signo PORQUE el intervalo de números

enteros decimales sin signo se pueden representar como -2^n y $+(2^n-1)$ donde n es el número de bits.

Respuesta: ____

12.3 Para representar números enteros decimales entre -74 y +73 se necesitan 8 bits PORQUE el intervalo -74 hasta +73 se encuentra entre -128 y +127 cuyos números con signo se representan por 7 bits.

Respuesta: ____

Los problemas 10, 11, 12 debe completarlos para dar la solución:

10. Un usuario quiso calcular la cantidad de localidades libres de memoria que podía tener su computadora; para tal fin realizó la siguiente operación:

6754_8 - AFA_H - 729

a) ¿el usuario puede asegurar que tiene suficientes localidades para almacenar números de 5 bits más el del signo? _____ ¿Por qué? Justifique su respuesta en la hoja. **R/. No, porque solamente puede almacenar números de hasta 5 bits sin signo.**

b) El número máximo sin signo que puede almacenarse en esas localidades es: _____ **R/. 25**

11. Una memoria puede almacenar números de hasta 16 bits sin signo en sus localidades de memoria disponibles. Si un archivo ocupa desde la dirección $AFEA_H$, hasta $B0CA_H$ dicho archivo puede ocupar ___ localidades de memoria.

R/. 224 localidades

12. Una PC, muy particular, tiene capacidad para almacenar en 46118 localidades. Si a partir de la dirección $35BA_H$ queda disponible y tiene que almacenar un archivo que ocuparía 16254 localidades de memoria; el intervalo de direcciones (después de almacenar ese archivo) que puede utilizarse en esa computadora va desde _____ hasta _____ y máximo le quedan _____ localidades libres.

R/. El intervalo va desde 7538_H hasta $B426_H$ y le quedan 16110 localidades libres.

Aplicación en computación

Haga un programa que realice cambio de una base cualquiera a otra; es decir, convierta números dados en una base ingresada por consola a otra base definida por el usuario. Debe mostrar el pantallazo del resultado.