

ARQUITECTURA DE LAS COMPUTADORAS

PRACTICA

(2008)

SISTEMAS NUMÉRICOS

INTRODUCCIÓN TEÓRICA:

Definimos **Sistema de Numeración** como al conjunto de reglas que permiten, con una cantidad finita de símbolos, representar un número natural cualquiera.

Los números pueden representarse en diversos sistemas de numeración, que se diferencian por su base. La BASE de un sistema numérico es el número de símbolos distintos que tiene el sistema para la representación de las cantidades en el mismo. El mayor de los símbolos es menor en una unidad a la base del sistema.. (Ejs.: base del sistema decimal: 10, símbolo mayor: 9; base del binario: 2, símbolo mayor: 1; base del octal: 8, símbolo mayor: 7; etc.). El menor de los símbolos es siempre el “0”.

Los símbolos utilizados se denominan CARACTERES. Estos presentan dos tipos de valores:

- VALOR INTRINSECO o ABSOLUTO: Es el referido a la unidad.
- VALOR RELATIVO: Es de acuerdo a la posición que ocupe en la configuración de un número.

Es posible representar un número según una expresión genérica como la siguiente:

$$N_r = a_n \cdot r^n + \dots + a_1 \cdot r^1 + a_0 \cdot r^0 + a_{-1} \cdot r^{-1} + \dots + a_{-m} \cdot r^{-m}$$

donde: r es la base del sistema

a_i son los caracteres, que pueden tomar valores entre 0 y $(r - 1)$

Esta expresión es una **suma de Valores Relativos**.

Vemos que:

$$\text{VALOR RELATIVO} = \text{VALOR ABSOLUTO} \times \text{PESO}$$

donde: PESO es la base del sistema elevada a un exponente que indica la posición del caracter dentro del número. Para la parte entera, el exponente es positivo y su valor absoluto es uno menos que la verdadera posición del caracter dentro del número. Para la parte fraccionaria, el exponente es negativo, y su valor absoluto indica la verdadera posición del caracter.

EJEMPLOS:

1) SISTEMA DECIMAL:

- a) SIMBOLOS: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- b) BASE: 10
- c) Cada caracter se denomina DIGITO.
- d) Dado el número: $N_{10} = 1534,27_{10}$, podemos desarrollarlo, según la expresión vista:
$$N_{10} = 1 \cdot 10^3 + 5 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0 + 2 \cdot 10^{-1} + 7 \cdot 10^{-2}$$

2) SISTEMA BINARIO:

- a) SIMBOLOS: 0, 1
- b) BASE: 2
- c) Cada caracter se denomina BIT.
- d) Dado el número: $N_2 = 1001,11_2$, se desarrollará:
$$N_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2}$$

3) SISTEMA OCTAL:

- a) SIMBOLOS: 0, 1, 2, 3, 4, 5, 6, 7.
- b) BASE: 8
- c) Cada símbolo se denomina caracter octal.
- d) Dado: $N_8 = 432,76_8$, será:
$$N_8 = 4 \cdot 8^2 + 3 \cdot 8^1 + 2 \cdot 8^0 + 7 \cdot 8^{-1} + 6 \cdot 8^{-2}$$

4) SISTEMA HEXADECIMAL:

- a) SIMBOLOS: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
- b) BASE: 16
- c) Cada símbolo se llama caracter hexadecimal.
- d) Si tenemos: $N_{16} = 2B41,C5_{16}$, su desarrollo será:
$$N_{16} = 2 \cdot 16^3 + B \cdot 16^2 + 4 \cdot 16^1 + 1 \cdot 16^0 + C \cdot 16^{-1} + 5 \cdot 16^{-2}$$

TABLA :

SIST. DECIMAL	SIST. BINARIO	SIST. OCTAL	SIST. 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
16	10000	20	10
17	10001	21	11

∴ LA BASE DE UN SISTEMA, EN SU PROPIO SISTEMA, SE ESCRIBE “10”.

CONVERSION DE SISTEMAS:

I) Para pasar un número entero del sistema decimal a otro sistema numérico, debe dividirse el número por la base nueva sucesivamente, hasta que el cociente sea menor que la base. Luego se ordenan los restos de izquierda a derecha, comenzando por el último, para poder obtener la nueva expresión del número.

Ejemplo: $87_{10} = N_8$

$$87 / \underline{8}$$

$$7 / \quad 10 / \underline{8}$$

$$2 / \quad 1$$

$$87_{10} = 127_8$$

II) Para pasar un número fraccionario decimal a otro sistema numérico, debe multiplicarse por la base del nuevo sistema, tantas veces, como caracteres queramos tener después de la coma.

Ejemplo: $0,67_{10} = N_{16}$

$$0,67$$

$$\times \underline{16}$$

$$\underline{10}, 72$$

$$0,72$$

$$\times \underline{16}$$

$$\underline{11}, 52$$

$$0,52$$

$$\times \underline{16}$$

$$\underline{8}, 32$$

$$\rightarrow N_{16} = 0, AB8..._{16}$$

III) En el caso que el número tenga parte entera y fraccionaria, debe trabajarse cada parte por separado, y luego juntar los resultados.

Ejemplo: $97,012_{10} = N_8$

$$97 / \underline{8}$$

$$0,012$$

$$1 / \quad 12 / \underline{8}$$

$$\times \underline{8}$$

$$4 / \quad 1$$

$$\underline{0}, 096$$

$$\times \underline{8}$$

$$\underline{0}, 768$$

$$\times \underline{8}$$

$$\underline{6}, 144$$

$$\rightarrow N_8 = 141,006..._8$$

IV) Para pasar un número de un sistema numérico cualquiera a decimal, debe hacerse la suma de los valores relativos de los caracteres del mismo.

Ejemplos:

$$1) 101_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 4 + 0 + 1 = 5_{10}$$

$$1) 11001,1_2 = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} = 16 + 8 + 1 + 0,5 = 25,5_{10}$$

$$2) 327,1_8 = 3 \cdot 8^2 + 2 \cdot 8^1 + 7 \cdot 8^0 + 1 \cdot 8^{-1} = 215,125_{10}$$

$$3) BA,81_{16} = B \cdot 16^1 + A \cdot 16^0 + 8 \cdot 16^{-1} + 1 \cdot 16^{-2} = 11 \cdot 16 + 10 \cdot 1 + 8 \cdot 16^{-1} + 1 \cdot 16^{-2} = 186,503..._{10}$$

V) Los sistemas numéricos cuya base es de la forma 2^n , ofrecen una simple conversión desde el binario. Un ejemplo sería el sistema numérico octal ($2^3=8$), y otro, el hexadecimal ($2^4=16$).

Veamos primero, una forma de pasar números enteros del sistema numérico binario al decimal, según el método enseñado, pero ordenándolos en una tabla:

----	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
----	256	128	64	32	16	8	4	2	1	
							1	1	0	$= 6_{10}$
					1	0	0	0	0	$= 16_{10}$
				1	1	0	0	1	0	$= 50_{10}$

Ahora, utilicemos esta misma tabla, para pasar números enteros del sistema numérico decimal, al binario, llenando los casilleros desde el que tiene mayor peso al menor, según los necesitemos, comenzando con el que tiene el peso igual o menor al número que estamos convirtiendo (o sea, pasando de un sistema numérico a otro):

----	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
----	256	128	64	32	16	8	4	2	1	
							1	0	1	$= 5_{10}$
				1	0	0	1	1	0	$= 38_{10}$
		1	0	0	0	0	1	0	1	$= 133_{10}$
	1	0	0	0	1	0	0	1	1	$= 275_{10}$

Ahora aprovecharemos esta forma de conversión, de la siguiente manera:

- a) Para pasar del binario al sistema cuya base es potencia de “2”, o sea, “ 2^n ” deben agruparse tantos bits como sea el exponente “n”. La agrupación se hace desde el bit de menor peso hacia el de mayor peso, para la parte entera, y desde el de mayor peso hacia el de menor peso, en la parte fraccionaria. Luego se escribe el caracter equivalente (según el nuevo sistema) de cada grupito.

Ejemplos:

I) Para pasar al sistema octal, como $8 = 2^3$, se agruparán de a tres bits. Como, al trabajar en binario con tres bits, podemos escribir del “0” al “7” decimal, que equivalen del “0” al “7” octal, usaremos las tres columnas de menor peso (las tres de la derecha), de la tabla recién vista.

$$1) 011010_2 = 011 \ 010_2 = 3 \ 2_8$$

Verificación: $011010_2 = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 + 1 \cdot 2^1 + 0 = 16 + 8 + 0 + 2 + 0 = 26_{10}$

$$26 / _8_$$

$$2 / \ 3 \rightarrow 32_8$$

$$2) 1101_2 = 001101_2 = 001 \ 101_2 = 15_8$$

$$3) 10,11_2 = 010,110_2 = 2,6_8$$

En este ejercicio, vemos que el número tiene parte entera y parte fraccionaria, y estamos usando la tabla que era para convertir números enteros! Veamos si el resultado es válido:

Verificación: $10,11_2 = 1 \cdot 2^1 + 0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} = 2 + 0,5 + 0,25 = 2,75_{10}$

$$2 / _8$$

$$0,75$$

$$2 / \ 0$$

$$_x _8$$

$$6,00$$

$$\rightarrow \text{Rta.: } 02,6_8$$

II) Lo mismo sucede cuando queremos pasar del sistema numérico binario al hexadecimal. Como la base **16** es 2^4 , agruparemos de a cuatro bits. Utilizaremos las cuatro columnas de menor peso de la tabla, ya que al trabajar en binario con cuatro bits, podremos escribir del “0” al “15” decimal que equivalen, del “0” al “15” hexadecimal, teniendo en cuenta que a partir del “10” debemos trabajar con las letras.

$$1) 111101,101_2 = \mathbf{0011} \ 1101 \ , \ 1010_2 = 3D,A_{16}$$

$$2) 1011111,111_2 = \mathbf{0101} \ 1111 \ , \ 1110_2 = 5F,E_{16}$$

- b) Para pasar del sistema numérico cuya base es potencia de “2” al binario, debe escribirse cada caracter de ese sistema con tantos bits como sea el exponente “n”.

Ejemplos:

$$1) 64,2_8 = 110\ 100,010_2$$

Verificación: $64,2_8 = 6 \cdot 8^1 + 4 \cdot 8^0 + 2 \cdot 8^{-1} = 48 + 4 + 0,25 = 52,25_{10}$

$$52 \ / \ \underline{2}$$

$$0,25$$

$$0,5$$

$$0 \ / \ 26 \ / \ \underline{2}$$

$$\underline{x \ 2}$$

$$\underline{x \ 2}$$

$$0 \ / \ 13 \ / \ \underline{2}$$

$$0,5$$

$$1,00$$

$$1 \ / \ 6 \ / \ \underline{2}$$

$$0 \ / \ 3 \ / \ \underline{2}$$

$$1 \ / \ 1$$

$$\therefore 52,25_{10} = 110100,01_2$$

$$2) 750,1_8 = 111\ 101\ 000,001_2$$

$$3) 2C4,B1_{16} = 0010\ 1100\ 0100,1011\ 0001_2$$

$$4) 970,F_{16} = 1001\ 0111\ 0000,1111_2$$

Ejercicios: Realizar las siguientes conversiones:

$$1) 249_{16} = N_2, N_8$$

Como:

$$2 = 0010$$

$$4 = 0100$$

$$9 = 1001$$

\Rightarrow

$$N_2 = 001001001001_2$$

$$001/001/001/001_2 = 1111_8 = N_8$$

$$2) 467_{10} = N_{16}$$

$$\text{Rta.: } 1D3_{16}$$

$$3) 57,3_{10} = N_2$$

$$\text{Rta.: } 111001,01001_2$$

$$4) 895,38_{10} = N_8$$

$$\text{Rta.: } 1577,302_8$$

$$5) 01001110,11_2 = N_{10}$$

$$\text{Rta.: } 78,75_{10}$$

$$6) 1234_8 = N_{10}$$

$$\text{Rta.: } 668_{10}$$

$$7) EFD,34_{16} = N_{10}$$

$$\text{Rta.: } 3837,2031_{16}$$

$$8) F78_{16} = N_8$$

$$\text{Rta.: } 7570_8$$

$$9) 111011111,10001_2 = N_{16}$$

$$\text{Rta.: } 1DF,88_{16}$$

$$10) 5260,0216_8 = N_{16}$$

$$\text{Rta.: } AB0,08E_{16}$$

$$11) 11,0B9_{16} = N_2$$

$$\text{Rta.: } 10001,000010111001_2$$

$$12) 1010011,100010_2 = N_8$$

$$\text{Rta.: } 123,42_8$$

$$13) 42,53_8 = N_2$$

$$\text{Rta.: } 100010,101011_2$$

SUMAS EN DIFERENTES BASES:

Se siguen las mismas reglas que en el sistema decimal. Se comienza la suma con la columna de menor peso y se sigue con las siguientes columnas, hasta terminar con la de mayor peso. En el caso de existir acarreo, se pasan ordenadamente a las columnas de mayor peso contiguas, y se sigue sumando.

Ejemplos:

$$1) \quad 110111_2$$

$$2) \quad 6799_{16}$$

$$3) \quad 11002_8$$

$$+ 100101_2$$

$$+ B761_{16}$$

$$+ 7150_8$$

$$101111_2$$

$$\underline{A41_{16}}$$

$$\underline{4750_8}$$

$$\underline{010011_2}$$

$$1293B_{16}$$

$$25122_8$$

$$10011110_2$$

Ejercicios:

$$1) 1011111_2 + 111_2 + 10011_2 = N_2$$

$$\text{Rta.: } 1111001_2$$

$$2) 107D_{16} + F9AB_{16} = N_{16}$$

$$\text{Rta.: } 10A28_{16}$$

$$3) 352_8 + 173_8 = N_8$$

$$\text{Rta.: } 545_8$$

$$4) 1000000_2 + 24_8 + 71_{16} = N_{16}$$

$$\text{Rta.: } C5_{16}$$

$$5) AE6_{16} + CF2_{16} = N_8$$

$$\text{Rta.: } 13730_8$$

$$6) 7765_8 + 110101100_2 = N_{16}$$

$$\text{Rta.: } 11A1_{16}$$

$$7) C18E_{16} + 43D6_{16} + 200F_{16} = N_{16}$$

$$\text{Rta.: } 12573_{16}$$

REPRESENTACION DE NUMEROS BINARIOS:

Un número binario puede representarse con signo o sin él.

A) NÚMEROS SIN SIGNO: Una configuración binaria de “n” bits puede representarse como suma de valores relativos, según lo vimos al comienzo del capítulo. Trabajando con esa expresión, llegamos a que el alcance de la representación sin signo es: $0 \leq N \leq 2^n - 1$

B) NÚMEROS CON SIGNO: Por convención, se reserva el bit más pesado para el signo: “0”, si el número es positivo y “1”, si es negativo. Existen tres modos de representación:

I) SIGNO Y MODULO.

II) COMPLEMENTO A LA BASE MENOS UNO (o Complemento a 1: C_1).

III) COMPLEMENTO A LA BASE (o Complemento a 2: C_2)

I) SIGNO Y MODULO: El bit más significativo representa el signo, y los (n-1) restantes, el valor absoluto del mismo.

Su rango de representación es: $-(2^{n-1} - 1) \leq N_{SM} \leq (2^{n-1} - 1)$

O sea, que tenemos 2^n configuraciones binarias, y dos de las cuales representan al cero.

Ejemplos: $+13_{10} = 00001101_2$ $+0_{10} = 00000000_2$
 $-13_{10} = 10001101_2$ $-0_{10} = 10000000_2$

En estas representaciones, al realizar una suma o una resta, debemos proceder de la siguiente manera: comparar valores absolutos y con ellos determinar el signo luego de la operación, o sumar y luego ver si el resultado es negativo o positivo.

II) COMPLEMENTO A LA BASE MENOS UNO: Es un **convenio** para **representar la inversa de un número** (positivo o negativo). Si la operación la está efectuando un sistema digital, debemos considerar su capacidad, es decir, el número de bits con que trabaja el mismo.

Matemáticamente se expresa con la siguiente fórmula: $-N = b^n - 1 - |N|$

En esta forma de representación, los números positivos son iguales que en el caso anterior. Los negativos son codificados como complementos a la base menos uno.

El alcance de esta representación es igual que en Signo y Módulo:

$$-(2^{n-1}) + 1 \leq N_{C1} \leq (2^{n-1}) - 1$$

Para escribir un número negativo debe escribirse el positivo, y ver **cuánto le falta para la base menos uno** (en binario, cuánto le falta para “1”) a cada uno de los bits que forman el número positivo. Tenemos, nuevamente, una doble representación del cero.

Ejemplos: $+13_{10} = 00001101_2$ $+0_{10} = 00000000_2$
 $-13_{10} = 11110010_2$ $-0_{10} = 11111111_2$

Como se ve, para obtener el complemento a 1, basta con invertir todos los bits del número dado.

En cualquier sistema numérico, si se suma el número N al negativo (-N) se obtiene “cero”.

$$\begin{array}{rcl}
 \text{Ejemplo:} & 13_{10} & 00001101_2 \\
 & + (-13)_{10} & + 11110010_2 \\
 & 00 & 11111111_2 \text{ (que es el cero “negativo”)}
 \end{array}$$

III) COMPLEMENTO A LA BASE: Al igual que el caso anterior, el **convenio de complemento a a la base** ^{número de dígitos} $(C a b^n)$ es un convenio para representar la **inversa de un número** (positivo o negativo). Si se hablase de “complemento a la base” a “secas”, debería aclararse con cuántos caracteres se está trabajando. Si la operación la está efectuando un sistema digital, debemos considerar su capacidad, es decir, el número de bits con que trabaja el mismo.

Matemáticamente se expresa con la siguiente fórmula: $-N = b^n - |N|$

Los números positivos son iguales que en los dos casos anteriores. Los negativos son codificados como complemento a la base.

El alcance es el siguiente: $-(2^{n-1}) \leq N_{C2} \leq (2^{n-1}) - 1$

Para obtener un número negativo, debe escribirse el número positivo y ver **cuánto le falta para llegar a la base**. En esta representación no existe el cero “negativo”, debido a esto, hay una configuración binaria adicional disponible que aumenta el alcance negativo en una unidad, respecto al caso de Complemento a la base menos 1.

$$\begin{array}{rcl}
 \text{Ejemplos:} & + 13_{10} = 00001101_2 & 0_{10} = 00000000_2 \\
 & - 13_{10} = 11110011_2
 \end{array}$$

Existen dos formas sencillas para obtener el complemento a la base. Una consiste en obtener el complemento a la base menos uno (invirtiendo todos los bits en el caso del binario) y sumarle un uno, en la posición menos significativa. La segunda, es tomar el número positivo y se va buscando desde el bit menos significativo al de mayor peso, un “1”; se copian todos los bits tal cual son, incluyendo a este primer 1, y luego se invierten los bits restantes.

Como es lógico, si sumo al número N, su complemento a la base, el resultado debe ser “0”.

$$\begin{array}{rcl}
 \text{Ejemplo:} & 13_{10} & 00001101_2 \\
 & + (-13)_{10} & + 11110011_2 \\
 & 00_{10} & 1\ 00000000_2 \text{ (el noveno bit se pierde, y queda el 0)}
 \end{array}$$

OPERACIONES EN SISTEMAS NUMÉRICOS:

La ecuación de la resta es:	$\text{Minuendo} - \text{Sustraendo} = \text{Resultado}$
O, lo que es lo mismo:	$\text{Minuendo} + (-\text{Sustraendo}) = \text{Resultado}$
Podríamos escribirla así:	$\text{Minuendo} + \text{Complemento del sustraendo} = \text{Resultado}$

Esta última ecuación será la que usaremos para realizar las restas, o sea, las transformaremos en sumas.

El procedimiento de trabajo es el siguiente:

* **OPERACION MANUAL:** Dados dos números, representados en algún tipo de convenio, debemos:

- 1) Igualar la cantidad de caracteres de ambos números, agregándole ceros al que tiene menor cantidad de caracteres.
- 2) Representar el minuendo (M) y la inversa del sustraendo (-S) en el complemento elegido.
- 3) Sumar al minuendo la inversa del sustraendo, es decir: **M + (-S)**.
- 4) Contar los caracteres del resultado. Si el mismo tiene uno más que los operandos, indica que el resultado es **positivo** y ese carácter se desprecia en Complemento a la base y se suma al resultado parcial en Complemento a la base menos uno, para obtener el resultado final. Si el resultado tiene igual cantidad de caracteres que los operandos, indica que el resultado es **negativo** y el resultado parcial debe complementarse para obtener el resultado final, utilizando el mismo convenio con que se realizó la operación.
- 5) Representar el resultado en el convenio de símbolo y módulo. En este convenio, se representa al número en valor absoluto y se le antepone el signo “+” o “-“, según corresponda.

* **OPERACIÓN AUTOMÁTICA:** La capacidad de la ALU es fija, por lo tanto, los números se representan todos con esa cantidad de caracteres. Cuando los números no son representables con esa cantidad, se produce el **overflow**.

Existe **overflow** o desborde o rebasamiento, cuando al trabajar con un sistema de **n** bits, se suman dos números de igual signo, representables en **n** bits, y el resultado es de signo contrario. Se haría entonces una interpretación errónea del resultado. Esto ocurre cuando la operación excede la capacidad de almacenamiento del sistema, o sea, que el resultado desborda el número de bits del registro destinado a recibirlo.

El overflow se manifiesta, entonces, cuando al sumar dos números positivos, el resultado da negativo, o, al sumar dos negativos, el resultado da positivo. Estos “errores” se corregirían, si se pudiesen realizar las mismas operaciones con un bit más.

Ejemplos de overflow (suponemos trabajar con 8 bits):

1) $\begin{array}{r} 01101011_2 \\ + 01001001_2 \\ \hline 10110100_2 \end{array}$	2) $\begin{array}{r} 10000001_2 \\ + 11110000_2 \\ \hline 1/01110001_2 \end{array}$
---	---

Ejemplos:

Realizar las siguientes restas como sumas:

- 1) Dados dos números decimales, representados en módulo, resolver su resta mediante una suma y dar el resultado en símbolo y módulo:

I) $57 - 759 = N_{10}$

Rta.: ♦ Igualamos la cantidad de dígitos de cada número: $057 - 759 = N_{10}$

♦ Ahora realizo la operación: $057 + (-759) = N_{10}$

a) representando ambos números en C a 10^3 y realizando la suma:

$$\begin{array}{r} 057 \\ + (-759) \\ \hline \end{array} \xrightarrow{\text{C a } 10^3} \begin{array}{r} 057 \\ + \underline{241} \\ \hline 298 \end{array} \rightarrow \begin{array}{r} 298 \\ \underline{1000} \\ \hline \end{array}$$

$$\Rightarrow N_{10} = -702 \text{ (en símbolo y módulo)}$$

b) representando ambos números en C a $(10^3 - 1)$ y realizando la suma:

$$\begin{array}{r} 057 \\ + (-759) \\ \hline \end{array} \xrightarrow{\text{C a } (10^3 - 1)} \begin{array}{r} 057 \\ + \underline{240} \\ \hline 297 \end{array} \rightarrow \begin{array}{r} 297 \\ \underline{999} \\ \hline \end{array}$$

$$\Rightarrow N_{10} = -702 \text{ (en símbolo y módulo)}$$

II) $21_{10} - 16_{10} = N_{10}$

Rta.:
$$\begin{array}{r} 21 \\ + (-16) \\ \hline \end{array} \xrightarrow{\text{C a } 10^2} \begin{array}{r} 21 \\ + \underline{84} \\ \hline 105 \end{array} \Rightarrow N_{10} = +05 \text{ (en símbolo y módulo)}$$

O bien:
$$\begin{array}{r} 21 \\ + \underline{16} \\ \hline \end{array} \xrightarrow{\text{C a } (10^2 - 1)} \begin{array}{r} 21 \\ + \underline{83} \\ \hline 104 \\ + \underline{1} \\ \hline 105 \end{array} \Rightarrow N_{10} = +05 \text{ (en símbolo y módulo)}$$

- 2) Dados dos números binarios, representados en módulo, realizar su resta como una suma y representar el resultado en símbolo y módulo: $110 - 10111 = x_{(2)}$

Rta.: Según lo visto en el ejercicio anterior, nos queda: $00110 + (-10111) = x_{(2)}$

$$\begin{array}{rcl}
 \text{a) } & 00110 & 00110 \\
 & - \underline{10111} & \xrightarrow{\text{en C a } 2^5} + \underline{01001} \\
 & & 01111 = -10001 \text{ (en símbolo y módulo)} \\
 \\
 \text{b) } & 00110 & 00110 \\
 & - \underline{10111} & \xrightarrow{\text{en C a } (2^5 - 1)} + \underline{01000} \\
 & & 01110 = -10001 \text{ (en símbolo y módulo)}
 \end{array}$$

Ejercicios: Realizar las restas indicadas como sumas:

1) Dados los siguientes números en módulo y símbolo, resolver la resta como suma y expresar el resultado en símbolo y módulo:

- | | |
|---|-----------------------------|
| a) $110110_2 - 11101_2 = N_2$ | Rta.: $011001_2 = +11001_2$ |
| b) $873_{16} - 1532_8 = N_{16}$ | Rta.: $+519_{16}$ |
| c) $31_8 - 50_8 = N_8$ | Rta.: -17_8 |
| d) $00111_2 - 01101_2 = N_2$ | Rta.: -00110_2 |
| e) $5601_8 - 111011101_2 = N_{16}$ | Rta.: $+9A4_{16}$ |
| f) $47_{16} - 6E_{16} = N_{16}$ | Rta.: -27_{16} |
| g) $177_8 - 01111011_2 = N_{16} = 2A_{16}$ | Rta.: -026_{16} |
| h) $-3F_{16} + 157_8 = 00110001_2 + N_8$ | Rta.: -001_8 |
| i) $010011_2 - 10101_2 = N_{8,2}$ | Rta.: $-02_8 = -000010_2$ |
| j) $1011_2 - 00101_2 = N_{16}$ | Rta.: 06_{16} |
| k) $65_8 - 01000_2 = N_2$ | Rta.: 101101_2 |
| l) $A7_{16} - 54_8 = N_{8,16}$ | Rta.: $173_8 = 7B_{16}$ |
| ll) $AF_{16} - 13B_{16} = N_8$ | Rta.: -0214_8 |
| m) $375_8 - 457_8 = N_8$ | Rta.: -062_8 |
| n) $AF_{16} - 3B_{16} = N_{16}$ | Rta.: 74_{16} |
| ñ) $011011_2 - 01110011_2 = N_2$ | Rta.: -01011000_2 |
| o) $317_8 - 52_{10} = N_{16}$ (No hacerlo en decimal) | Rta.: $+09B_{16}$ |

- 2) Realizar las siguientes operaciones “automáticas” encadenadas, mostrando los resultados parciales y el resultado final que queda almacenado. Realizar las restas como sumas. Suponer que estamos trabajando con un sistema digital con una capacidad de $n = 8$ bits y que trabaja con el convenio de :

♦ Complemento a dos (C a 2):

a) $21 - 9 + 2 - 12 = 2$

Rta.: $21 = 00010101$

$2 = 00000010$

I) 00010101

$+ \underline{11110111}$

$1/00001100$

$9 = 00001001$

$12 = 00001100$

II) 00001100

$+ \underline{00000010}$

00001110

$- 9 = 11110111$

$- 12 = 11110100$

III) 00001110

$+ \underline{11110100}$

$1/00000010$

b) $17 - 44 + 11 = - 16$

Rta.: $17 = 00010001$

$11 = 00001011$

I) 00010001

$+ \underline{11010100}$

11100101

$44 = 00101100$

$- 44 = 11010100$

II) 11100101

$+ \underline{00001011}$

$11110000 (= - 00010000_2 = -16_{10})$

c) $- 96 + 38 - 3 + 21 = - 40$

Rta.: $96 = 01100000$

$3 = 00000011$

I) 10100000

$+ \underline{00100110}$

11000110

$- 96 = 10100000$

$- 3 = 11111101$

II) 11000110

$+ \underline{11111101}$

$1/11000011$

$38 = 00100110$

$21 = 00010101$

III) 11000011

$+ \underline{00010101}$

11011000

$(= - 00101000_2 = - 40_{10})$

♦ Complemento a uno (C a 1):

a) $34 + 23 - 56 - 8 = - 7$

Rta.: $34 = 00100010$

$- 56 = 11000111$

I) 00100010

$+ \underline{00010111}$

00111001

$23 = 00010111$

$8 = 00001000$

II) 00111001

$+ \underline{11000111}$

$1/00000000$

$+ \underline{\quad\quad\quad 1}$

00000001

$56 = 00111000$

$- 8 = 11110111$

III) 00000001

$+ \underline{11110111}$

11111000

$(= - 0001111_2 = - 7_{10})$

b) $64 - 71 + 25 = 18$

Rta.: $64 = 01000000$

$25 = 00011001$

I) 01000000

$+ \underline{10111000}$

11111000

$71 = 01000111$

$- 71 = 10111000$

II) 11111000

$+ \underline{00011001}$

$1/00010001$

$+ \underline{\quad\quad\quad 1}$

$00010010 (= 18_{10})$

c) $- 87 + 29 - 2 = - 60$

Rta.: $87 = 01010111$

$2 = 00000010$

I) 10101000

$+ \underline{00011101}$

11000101

$(= - 00111010_2 = - 58_{10})$

$- 87 = 10101000$

$- 2 = 11111101$

II) 11000101

$+ \underline{11111101}$

$1/11000010$

$+ \underline{\quad\quad\quad 1}$

$11000011 (= - 00111100_2 = - 60_{10})$

ARITMÉTICA DE PUNTO FIJO Y PUNTO FLOTANTE:

La representación de un número en **punto fijo** consiste en escribir el mismo colocando la coma en el lugar que le corresponde, separando la parte entera de la fraccionaria. Esto hace que esté muy limitada la capacidad de almacenamiento, ya que se deben prever cifras enteras y varias fraccionarias. Normalmente, los datos de punto (coma) fijo son números enteros con signo.

La mayoría de las calculadoras y computadoras escriben los números en **punto flotante** o **notación científica**. Esta es una representación que permite localizar automáticamente la coma en el curso del cálculo.

Un número N , en punto o coma flotante, se representa de la siguiente manera: $N = M \cdot B^E$

donde: M es la mantisa, B es la base de la representación y E es el exponente.

Como las computadoras disponen de un número fijo “ n ” para escribir el número binario, el mismo se almacena de la siguiente manera:

$$\boxed{n(M)} \boxed{n(E)} = n$$

donde: “ $n(M)$ ” son los bits que conforman la mantisa y “ $n(E)$ ” son los del exponente.

Ejemplos:

1) Encontrar la representación en punto flotante de los siguientes números en punto fijo:

- en el sistema binario:

$$\begin{aligned} \text{a) } 27,5_{10} &= 0,110111_2 \times 2^{(5)_{10}} = .110111_2 \times 2^{(5)_{10}} \\ &= .110111_2 \times 2^{101} = 0,110111_2 \times 2^{101} \end{aligned}$$

$$\text{b) } .00001101_2 = .1101_2 \times 2^{-(4)_{10}} = .1101_2 \times 2^{-100} = 0,1101_2 \times 2^{-100}$$

- sin cambiar de base:

$$\text{a) } 342,16_{10} = 0,34216 \times 10^3$$

$$\text{b) } 0,056_7 = 0,56 \times 7^{-1}$$

2) Encontrar la representación en punto fijo de los siguientes números en punto flotante:

- en el sistema binario:

$$\text{a) } .110101_2 \times 2^{1011} = .110101_2 \times 2^{(11)_{10}} = 11010100000_2$$

$$\text{b) } -0,5 \times 10^0 = -0,5_{10} = -0,1_2$$

- sin cambiar de base:

$$\text{a) } -0,458 \times 9^{-3} = -0,000458_9$$

$$\text{b) } 0,22 \times 7^4 = 2200_7$$

Ejercicios:

1) Encontrar la representación en punto flotante, sin cambiar de base:

- | | |
|----------------------|---|
| a) $10110,1101_2$ | R: $.10110110 \times 2^{101} = 0,10110110 \times 2^{101}$ |
| b) $.00000010110_2$ | R: $.10110 \times 2^{(-110)} = 0,10110 \times 2^{(-110)}$ |
| c) 1000000000000_2 | R: $.1 \times 2^{1101} = 0,1 \times 2^{1101}$ |
| d) $36,7_8$ | R: $0,367 \times 8^2$ |
| e) $0,000A8C_{16}$ | R: $0,A8C \times 16^{-3}$ |

2) Encontrar la representación en punto fijo de los siguientes números en punto flotante, sin cambiar de base:

- | | |
|---------------------------------|---------------------------|
| a) $.101101 \times 2^{(-1011)}$ | R: $.00000000000101101_2$ |
| b) $.101101 \times 2^{110}$ | R: 101101_2 |
| c) $.11111 \times 2^{1010}$ | R: 1111100000_2 |
| d) $0,321 \times 4^{-2}$ | R: $0,00321_4$ |
| e) $0,43 \times 5^3$ | R: 430_5 |

PRECISIÓN SIMPLE Y DOBLE:

En el caso de usar palabras de 16 bits, podemos representar, en punto fijo, números entre (-32768) y (+32767). Si se necesitase representar números más grandes, se usan palabras de 32 bits, o sea dos palabras de 16 bits. El primer caso se llama **precisión simple** y el segundo, **precisión doble**.

En el caso de trabajar con punto flotante, en precisión simple, se usa una palabra para escribir la mantisa y el exponente ($n_{(M)}$ y $n_{(E)}$). En precisión doble, una palabra es para escribir la mantisa y la otra, para el exponente (pueden existir modificaciones según el micro).

SISTEMAS DE CODIFICACIÓN:

“Codificar” significa poner en correspondencia biunívoca dos conjuntos de símbolos.

Podemos considerar al sistema binario como un código, ya que con dos símbolos: 0 y 1, representamos al sistema decimal. Existen códigos numéricos y códigos alfanuméricos.

I) CÓDIGOS NUMÉRICOS:

a) Código BCD: BCD significa: Decimal Codificado en Binario. Es el código más simple: describe en binario, los diez caracteres decimales. Se codifica cada dígito con cuatro bits.

La correspondencia es la siguiente:

0 - 0000	5 - 0101
1 - 0001	6 - 0110
2 - 0010	7 - 0111
3 - 0011	8 - 1000
4 - 0100	9 - 1001

Ejemplo: $12_{10} = 00010010_{\text{BCD}}$

b) Código de GRAY: Este código se denomina también “Código Binario Reflejo” o “Código Espejo”.

La correspondencia con el sistema decimal es la siguiente (supongamos trabajar con 8 bits):

0 - 00000000	5 - 00000111	10 - 00001111
1 - 00000001	6 - 00000101	11 - 00001110
2 - 00000011	7 - 00000100	12 - 00001010
3 - 00000010	8 - 00001100	etc...
4 - 00000110	9 - 00001101	

Como se puede observar, entre un número en Gray y el siguiente en la tabla, existe **adyacencia** (un solo cambio de bit).

◆ CONVERSIÓN DE BINARIO A GRAY: Explicaremos dos métodos:

1er. método) Se suma al número en binario, el mismo número corrido un lugar a la derecha, perdiendo el menos significativo. La suma se hace sin acarreo.

$$\begin{array}{r} \text{Ejemplo: } 12_{10} = 1100_2 \\ \phantom{12_{10} = } 1100 \\ \phantom{12_{10} = } + \underline{110/0} \\ \phantom{12_{10} = } 1010 \text{ GRAY} = 12_{10} \end{array}$$

2do. método) Comenzando con el MSB del número binario, se compara cada par de bits sucesivos:

- Si son iguales, se coloca un “0” en el número en Gray.
- Si son distintos, se coloca un “1” en la palabra en código Gray.

Para comenzar, se compara el primer bit con “0”.

Nota: Esta acción es igual a sumar en “módulo 2” a cada par de bits. Esto significa, sumar como una OR-EXCLUSIVA.

$$\text{Ejemplo: } 12_{10} = 1100_2 = 1010_{\text{GRAY}}$$

♦ CONVERSION DE GRAY A BINARIO:

Veremos dos métodos, que son los inversos a los recién explicados:

1er método) Se toma el resultado de la “cuenta”, que es el número en Gray, y se va averiguando desde el bit más significativo al de menor peso, cuáles fueron los bits de los sumandos, teniendo en cuenta la ausencia de acarreo y el desplazamiento de los bits de los mismos.

Por ejemplo:

$$\begin{array}{r} \text{A B C D} \\ + \quad \text{A B C / D} \\ \hline 1 \ 0 \ 1 \ 0_{\text{GRAY}} \end{array}$$

Luego:

$$\begin{aligned} \text{A} + 0 &= 1 \rightarrow \text{A} = 1 \\ \text{B} + \text{A} &= 0 \rightarrow \text{B} + 1 = 0 \rightarrow \text{B} = 1 \\ \text{C} + \text{B} &= 1 \rightarrow \text{C} + 1 = 1 \rightarrow \text{C} = 0 \\ \text{D} + \text{C} &= 0 \rightarrow \text{D} + 0 = 0 \rightarrow \text{D} = 0 \end{aligned}$$

O sea: $\text{A B C D} = 1 \ 1 \ 0 \ 0_2 = 12_{10}$

2do. método) Reglas:

2₁) El MSB de Gray es igual al MSB del binario, y todos los bits hacia la derecha del binario siguen siendo iguales al número en Gray, hasta el primer 1, inclusive.

2₂) Ahora se usa el bit del código Gray, como control:

- Si es “1”, cambiar el carácter binario que precede, para obtener el bit actual.
- Si es “0”, se repite el carácter binario.

Repetir este último punto, para convertir todos los bits del número en código Gray.

Nota: El bit que se cambia o se repite, es sólo en la palabra binaria. El código Gray, actúa como control.

Ejemplo: $1010_{\text{GRAY}} = 1100_2 = 12_{10}$

c) Códigos detectores y autocorrectores de errores: Existen códigos que detectan errores en la transmisión de información. Hay otros, que no sólo detectan los errores, sino que también los corrigen. En este último grupo, uno de los más conocidos, es el denominado Código Autocorrector de Hamming.

Ejercicios:

- | | |
|---|------------------------------|
| 1) $011010111101_2 = N_{GRAY}$ | R: 010111100011_{GRAY} |
| 2) $111010101110_2 = N_{GRAY}$ | R: 100111111001_{GRAY} |
| 3) $011010111101_2 = N_{GRAY}$ | R: 010111100011_{GRAY} |
| 4) $1001110101101_{GRAY} = N_2$ | R: 1110100110110_2 |
| 5) $10101010101_{GRAY} = N_2$ | R: 11001100110_2 |
| 6) $01110111011110_{GRAY} = N_2$ | R: 01011010010100_2 |
| 7) $689_{10} = N_{NBCD} = N_{BCD}$ | R: 011010001001_{BCD} |
| 8) $164_8 = N_{NBCD} = N_{BCD}$ | R: 000100010110_{BCD} |
| 9) $01001001_{BCD} = N_8$ | R: 61_8 |
| 10) $10010_2 = N_{BCD}$ | R: 00011000_{BCD} |
| 11) $37_8 + 10001101000_{BCD} = N_{16}$ | R: $1F3_{16}$ |
| 12) $FA_{16} = N_{GRAY}$ | R: 10000111_{GRAY} |
| 13) $11011_{GRAY} = N_{NBCD}$ | R: 00011000_{NBCD} |
| 14) $1010111011_2 - 7A2_{16} = N_{BCD}$ | R: -0001001001010101_{BCD} |
| 15) $111001_{BCD} = N_{GRAY}$ | R: 0110100_{GRAY} |
| 16) $11001_{BCD} + 1011_2 = N_2$ | R: 011110_2 |
| 17) $1A_{16} + 1011_2 = N_{BCD}$ | R: 00110111_{BCD} |
| 18) $10111_2 - 101001_{BCD} = N_{16}$ | R: -6_{16} |
| 19) $1010010_{BCD} + 1011_{GRAY} = N_{BCD}$ | R: 01100101_{BCD} |
| 20) $70_8 - 10011_{GRAY} = N_{GRAY}$ | R: 010110_{GRAY} |

II) CÓDIGOS ALFANUMÉRICOS:

Representan caracteres alfabéticos, numéricos y caracteres especiales (>, <, {, }, +, %, =, \$, -, ?, etc.). Es usual que los códigos alfanuméricos sufran pequeñas alteraciones, al ser utilizados en países con distintos alfabetos.

- a) Código ASCII: (American Standard Code for Information Interchange o Código estandar americano para el intercambio de información). Este código se ha adoptado a nivel internacional. Inicialmente, se desarrolló con 6 bits, permitiendo 2^6 (64) configuraciones distintas: 10 caracteres numéricos decimales, 26 caracteres alfabéticos y 28 símbolos especiales. Posteriormente, se aumentó a 7 bits, obteniéndose 2^7 (128) configuraciones diferentes, que incluían los mismos caracteres alfanuméricos de la primera versión, más algunas expresiones sintácticas y símbolos que permitían realizar instrucciones operativas de unidades periféricas. Actualmente utilizan 8 bits, o sea, $2^8 = 256$ configuraciones distintas.
- b) Código EBCDIC: (Extended Binary Coded Decimal Information Code). Este código fue creado por IBM, y es, después del ASCII, el más utilizado. Utiliza 8 bits, y deja varias de las 256 configuraciones, sin ningún significado.

CÓDIGOS CON PARIDAD:

El código puede ser de **paridad par** o de **paridad impar**. Se usa para detectar errores en transmisión de datos.

Al número que se desea transmitir, se le adiciona un bit, llamado “bit de paridad”. Este número binario que se agrega hará que el total de “1” en un número sea par o impar, según la paridad elegida.

Cuando se recibe un número, se chequea la paridad (par o impar, según se haya elegido), y se toma como correcto si pasa el test.

Al código BCD, se le puede agregar un bit de paridad (par o impar), a la derecha.

Ejercicios:

1) Decir “FALSO” o “VERDADERO” en las siguientes palabras en código BCD con paridad:

- | | | |
|-----------|---------------|----------------------------|
| a) 1001 0 | paridad impar | (R: FALSO) |
| b) 1010 0 | paridad par | (R: FALSO, pues no es BCD) |
| c) 1000 0 | paridad par | (R: FALSO) |
| d) 0111 0 | paridad impar | (R: VERDADERO) |
| e) 1011 0 | paridad impar | (R: FALSO, pues no es BCD) |

2) Escribir en BCD con la paridad indicada:

- | | |
|---|---------------------------|
| a) 961 ₁₀ con bit de paridad par, por dígito. | (R: 100100110000011) |
| b) 5798 ₁₀ con bit de paridad impar, por dígito. | (R: 01011011101001110000) |
| c) 620 ₁₀ con bit de paridad impar por dígito. | (R: 011010010000001) |
| d) 6058 ₁₀ con bit de paridad par por dígito. | (R: 01100000000101010001) |

3) Decir FALSO o VERDADERO en las siguientes palabras con paridad:

- | | | |
|---------------|---------------|----------------|
| a) 11110001 1 | paridad par | (R: VERDADERO) |
| b) 00000111 1 | paridad impar | (R: FALSO) |
| c) 110110010 | paridad par | (R: FALSO) |
| d) 111000111 | paridad impar | (R: FALSO) |
| e) 011111101 | paridad impar | (R: VERDADERO) |

4) Colocar el bit necesario para obtener paridad par en las siguientes palabras:

- | | |
|-------------|--------|
| a) 01110100 | (R: 0) |
| b) 11100110 | (R: 1) |
| c) 11111111 | (R: 0) |

5) Colocar el bit necesario para obtener paridad impar en las siguientes palabras:

- | | |
|-------------|--------|
| a) 01101100 | (R: 1) |
| b) 10001001 | (R: 0) |
| c) 00001001 | (R: 1) |