Temas: SISTEMAS DE NUMERACIÓN POSICIONAL. SUMAS. RESTAS USANDO COMPLEMENTO A LA BASE Y A LA BASE MENOS UNO. MULTIPLICACIONES. OTROS CÓDIGOS.

Objetivos:

Que el alumno:

a) Aprenda los sistemas de numeración que la computadora utiliza.
b) Entienda las técnicas de cambio de base entre sistemas de numeración.
c) Realice las operaciones básicas con los distintos sistemas de numeración.
d) Sepa restar, como suma de números complementados y justificar la razón teórica que permita esta operación.
e) Se familiarice con el uso de códigos de tétradas y ASCII.

Bibliografía:

- Murdocca y Heuring. "Principios de Arquitectura de Computadoras", Prentice Hall.
- Stalling, William. "Organización y Arquitectura de Computadores", Prentice Hall, 5ta y 7ma edición.
- Ginzburg, M.C. "Algebra de Boole Aplicada a Circuitos de Computación", Biblioteca Técnica Superior.
- Morris Mano. "Arquitectura de Computadoras", Prentice Hall.
- Otros.

SISTEMAS DE NUMERACIÓN

El hombre en su necesidad de representar la realidad numéricamente ha adoptado símbolos para tal fin. Por eso decimos que un sistema numérico: *Es una colección de números ordenados que representan una realidad*.

Ejemplos de sistemas de numeración:

- Decimal
- Binario
- Octal
- Hexadecimal

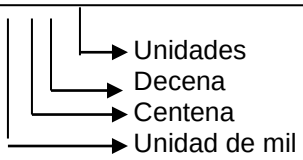
Los sistemas de numeración que vamos a estudiar se denominan posicionales; puesto que el lugar o posición que cada número ocupa dentro de la cifra es importante para determinar la realidad que se quiere representar.
Ejemplo: 101, 110, 011. A pesar de que tienen los mismos números, representan distintas realidades.

Características:

1. Todos los sistemas numéricos posicionales constan de un número finito básico de elementos a partir de los cuales se construye el sistema de numeración, y que se denomina Base o Raíz del Sistema.
   Ejemplo: 0 _ 1 _ 2 _ 3 _ 4 _ 5 _ 6 _ 7 _ 8 _ 9 (elementos del sistema decimal).

2. Cada símbolo aislado representa un número específico de unidades.

   Ejemplo: 1 2 3 4



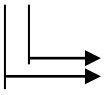
3. Existe un número cero para indicar la ausencia de elementos a representar. Los símbolos del sistema pueden ordenarse de forma monótona creciente.

Ejemplo: 0_ 01_ 02_ 03_ 04_ 05_ 06_ 07_ 08_ 09_ 10_ 11_ 12…

Elementos Básicos — Se forman combinando un Elemento Básico

4. Formando parte de un número compuesto por varios símbolos; un mismo símbolo tiene un significado o "peso", distinto según su posición relativa en el conjunto.

Ejemplo: 1 4 4 5



5. La posición extrema derecha corresponde a unidades (peso=1); a partir de ella, cada posición tiene el peso de la que está su derecha multiplicada por la base resultando siempre que el peso de cada posición es una potencia de la base.

Ejemplo:

3	2	1	0 → posición
1000	100	10	1 → peso
1	4	3	$4_{(10)}$ → número decimal

♦ **Sistema Numérico Posicional Binario (S.N.P. Binario o de base 2)**

Consta de 2 elementos básicos "0" y "1" a partir de los cuales se construye el sistema numérico completo.

Ejemplo: **0**_ **1**_ 10_ 11_ 100_ 101_ 110_ 111_ 1000_ 1001_ 1010_ 1011…∞

Valor de peso:

7	6	5	4	3	2	1	0	→ posición
128	64	32	16	8	4	2	1	→ peso
0	1	1	0	0	1	1	1_2	→ numero binario

♦ **Sistema Numérico Posicional Octal (S.N.P. Octal o de base 8)**

Consta de 8 elementos básicos (del 0 al 7).

Ejemplo: **0_1_2_ 3_ 4_ 5_ 6_ 7_** 10_ 11_ 12_ 13_ 14_ 15_ 16_ 17… 27… 100… 1000…∞

Valor de peso:

4	3	2	1	0	→ posición
4096	512	64	8	1	→ peso
7	6	4	3	1_8	→ numero octal

♦ **Sistema Numérico Posicional Hexadecimal (S.N.P. Hexadecimal o de base 16)**

Consta de 16 elementos básicos (del 0 al 9 y de la A a la F):

Ejemplo: **0_ 1_ 2_ 3_ 4_ 5_ 6_ 7_ 8_ 9_ A_ B_ C_ D_ E_ F_** 10_ 11_ 12_ 13_ 14_ 15_ 16_ 17_ 18_ 19_ 1A_ 1B_ 1C_ … 1F9_ 1FA… 2FDF…∞

Valor de peso:

4	3	2	1	0	→ posición
65536	4096	256	16	1	→ peso
F	A	2	3	4_{16}	→ numero hexadecimal

Nota:
En este práctico haremos hincapié en los sistemas de numeración posicional binarios y hexadecimales.

A continuación, podemos observar en la *TABLA 1* la equivalencia entre los sistemas.

BASE 2	BASE 8 → $2^3 = 8$	BASE 10	BASE 16 → $2^4 = 16$
000**0**	**0**	**0**	**0**
000**1**	**1**	**1**	**1**
0010	**2**	**2**	**2**
0011	**3**	**3**	**3**
0100	**4**	**4**	**4**
0101	**5**	**5**	**5**
0110	**6**	**6**	**6**
0111	**7**	**7**	**7**
1000	10	**8**	**8**
1001	11	**9**	**9**
1010	12	10	**A**
1011	13	11	**B**
1100	14	12	**C**
1101	15	13	**D**
1110	16	14	**E**
1111	17	15	**F**

Métodos para Cambios de Base, técnicas No Reducidas.

- **DE DECIMAL A OTRO SISTEMA:**

Divisiones Sucesivas

El método de divisiones sucesivas se utiliza para convertir un número decimal a otra base. Consiste en dividir un número decimal en forma sucesiva en la base del sistema al que deseamos pasarlo, hasta llegar a un valor indivisible (menor a la base de origen). El resultado se leerá a partir del último cociente continuando con todos los restos obtenidos.

Ejemplo:

42 | 2 0 21 | 2 1 10 | 2 0 5 | 2 1 2 | 2 0 1	42 | 16 10 2
Decimal a Binario: 42_{10}= **0**101010_2	Decimal a Hexadecimal: 42_{10}= $2A_{16}$

Nota:
Una vez que obtenemos el número binario hay que considerar el bit de signo, es el primer bit de la izquierda, debe comenzar con un cero, indicando que es un número binario positivo, ya que parte de un número decimal positivo, si no tiene el "0" lo colocamos. En el ejemplo se lo resalta en rojo.

Método de Parte Entera

Cuando tenemos el caso de decimales con parte fraccionaria que debemos transformar a otro sistema numérico, utilizamos para la parte entera el método de "divisiones sucesivas" visto anteriormente, y para la parte fraccionaria el método de "parte entera". Este método consiste en

extraer, del número de origen, la parte fraccionaria, a continuación se la multiplica por la base del sistema al que queremos llegar. Del resultado obtenido, se extraerá la parte entera, luego se toma, nuevamente, la parte fraccionaria de ese resultado y se vuelve a multiplicar por la base del sistema al que queremos llegar. Así sucesivamente hasta llegar a un resultado aceptable. El resultado final estará compuesto por el obtenido de las divisiones sucesivas (parte entera) y luego de la coma, se escribirán los resultados obtenidos por el método de "parte entera", escribiéndolas a partir del primero encontrado hasta el último.

Ejemplo: pasar el número decimal 92,36 a binario

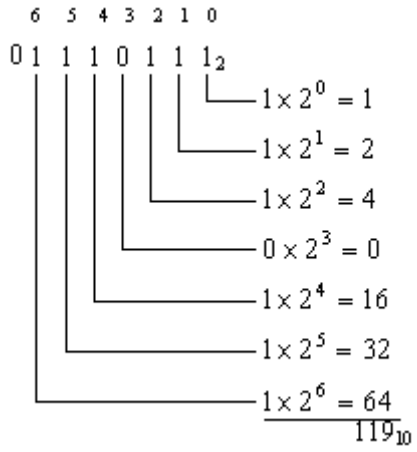
Parte fraccionaria	Parte entera
$0.36 \times 2 = 0.72 \rightarrow 0$ $0.72 \times 2 = 1.44 \rightarrow 1$ $0.44 \times 2 = 0.88 \rightarrow 0$ $0.88 \times 2 = 1.76 \rightarrow 1$	92 | 2 0 46 | 2 0 23 | 2 1 11 | 2 1 5 | 2 1 2 | 2 0 1
Resultado final: **0**1011100,0101$_2$ Por más que el último cociente sea 1, en binario le agregaremos un cero (0) al número final, para representar al número como positivo.Ya veremos más adelante éste tema.	

- **DE OTRO SISTEMA A DECIMAL:**

Multiplicaciones Sucesivas

Este método se utiliza para cambiar la base desde cualquier sistema numérico a decimal. Consiste en multiplicar cada elemento del número por la base del sistema, elevado a la posición que ocupa dentro del número; luego se suman los resultados y se obtiene el número decimal equivalente.

Ejemplo: convertir el numero binario 01110111$_2$ a decimal



$01110111_2 = 119_{10}$

Por medio del uso de la *TABLA 2*, podemos tener una ayuda para evaluar el equivalente decimal de cada posición binaria.

Posición = X	7	6	5	4	3	2	1	0	-1	-2	-3	-4
Valor = 2^X	128	64	32	16	8	4	2	1	0,5	0,25	0,125	0,0625

Como vemos, el numero puede constar de una parte entera *Ne* y de una parte fraccionaria *Nf*. Para convertir la parte fraccionaria también puedo hacer uso de la TABLA de POSICIÓN, aunque en realidad lo que estamos haciendo es asignarles peso negativo a las posiciones a partir de la coma hacia la derecha.

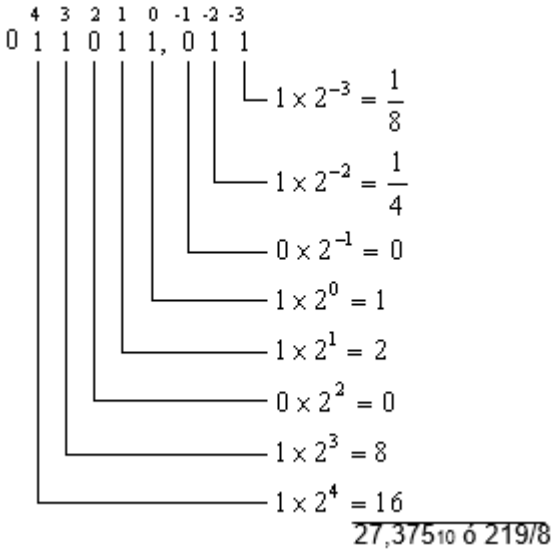
Nota: es muy importante que comprendas y uses la *TABLA 2.*

Veamos cómo sería el procedimiento para convertir un número binario (0101,1101) en decimal:

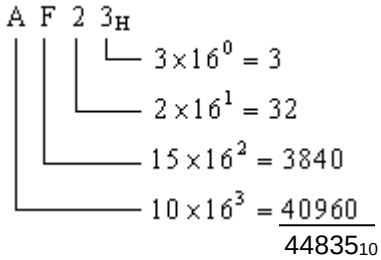
$N_{10} = 0 \times 2^{3} + 1 \times 2^{2} + 0 \times 2^{1} + 1 \times 2^{0} + 1 \times 2^{(-1)} + 1 \times 2^{(-2)} + 0 \times 2^{(-3)} + 1 \times 2^{(-4)}$
$N_{10} = 0 + 4 + 0 + 1 + 0.5 + 0.25 + 0 + 0.0625$
$N_{10} = 5{,}8125$

Otros ejemplos:

Binario a Decimal con parte fraccionaria



Hexadecimal a Decimal:



$AF23_H = 44835_{10}$

Técnicas Reducidas Para Cambio De Base

- **Técnica reducida para conversión de binario-decimal y viceversa.**

La técnica reducida para el pasaje de un número binario a decimal consiste en multiplicar los pesos de los elementos iguales a 1, de acuerdo a la posición del elemento dentro del número, por el dígito y realizar la posterior suma. En esta técnica se utilizan los valores de peso para el cambio de base.
Ejemplo:

128 64 32 16 8 4 2 1 → pesos
0 1 0 0 1 1 0 1 → número binario
128*0 + 64*1 + 8*1 + 4*1 + 1*1 = 64 + 8 + 4 + 1 **= 77_{10}**

Para pasar de decimal a binario se colocan los pesos (valores de las posiciones) hasta pasar el número decimal a convertir, luego se colocan los 1 en las posiciones tales que sumados los valores den el número decimal, al resto de las posiciones se les pone valor cero.

Nota: todo número binario deberá comenzar con un cero, lo que indica que se trata de un número positivo. Luego, usando complemento, se lo puede pasar a su equivalente negativo.

Ejemplo:

24 → número decimal
32 16 8 4 2 1 → pesos
0 1 1 0 0 0 → número binario

- **Técnica reducida para conversión de binario a hexadecimal y viceversa.**

Debido a que un dígito hexadecimal se corresponda con un grupo de cuatro dígitos binarios, es posible la conversión de BINARIO a HEXADECIMAL y viceversa de una forma más sencilla.

Para la conversión de un número binario a uno hexadecimal, se separa, de derecha a izquierda, el número binario en grupos de 4 dígitos. Luego a cada grupo se lo pasa a su valor hexadecimal, teniendo en cuenta el valor de las posiciones.
Ejemplo: Convertir el número binario 010101001011 a su equivalente hexadecimal.

0101 0100 1011
5 4 B
Es decir $010101001011_2 = 54B_{16}$

Para pasar de hexadecimal a binario se reemplaza a cada dígito hexadecimal por su equivalente binario, pero teniendo en cuenta que debemos trabajar en grupos de cuatro bits, según la Tabla 1.
Ejemplo: Convertir el número hexadecimal A7, 95 a su equivalente binario.

A 7 , 9 5
1010 0111 , 1001 0101
Es decir $A7,95_{16} = 0$10100111,10010101$_2$

Nota: Se puso un cero adelante para indicar que es un número positivo.

- **Técnica reducida para conversión de decimal a hexadecimal y viceversa.**

Para pasar de decimal a hexadecimal o de hexadecimal a decimal, como ambos sistemas no se relacionan, primero debemos pasar a binario el número y luego proceder como en las técnicas anteriores.

Nota: se recomienda usar las técnicas reducidas de conversión.

Operaciones de Sumas, Restas y Multiplicaciones.

- **Sumas**

El algoritmo de suma aprendido en la escuela primaria sigue teniendo aplicación para los SNP de otra base, la única salvedad que deberemos prestar atención es la base particular en la que estamos trabajando. Por este motivo, recomendamos hacer uso de la tabla 1.

- **Suma de números binarios**

Usaremos la *TABLA 3* de suma para el SNP binario, que nos servirá para poder realizar esta operación.

+	0	1
0	0	1
1	1	10

Como ejemplo realizaremos la siguiente suma de números binarios:

$$\begin{array}{r} 0100011 \\ +\ 0001101 \\ \hline 0110000 \end{array}$$

Se suman primero los dos dígitos de menor peso, y obtenemos como resultado 10. Como esta cantidad, que es la base, no puede representarse por un solo dígito, (1+1=10), tendré un acarreo (llevo) del que deberé considerar en la suma de los siguientes dígitos. Así sucesivamente, columna por columna, vamos sumando según la tabla 3, teniendo en cuenta los acarreos, en caso de que los hubiera hasta llegar al final de la operación.

- **Suma de números hexadecimales**

Ejemplo: Sumar los números hexadecimales 23 y 2D.

$$\begin{array}{r} 23 \\ +\ 2D \\ \hline 50 \end{array}$$

Puesto que, en hexadecimal 3+D = 3 + 13 = 16 (ver tabla 1), y 16 no se puede representar en hexadecimal, le resto la base (16 – 16 = 0), pongo el resultado de la resta (0) y "llevo" las veces que debí restar la base (en el ejemplo llevo 1), quedando 2 + 2 + 1 = 5. El resultado final es el número hexadecimal 50.

♦ **Restas**

Para restar también valen los algoritmos estudiados y memorizados en la escuela primaria. Ahora, debemos tener en cuenta que, en lugar de acarreo o llevo, tendremos un ”pido” a la columna anterior, en caso de no poder efectuarse la resta directamente.

○ **Resta de números binarios**

Veamos la siguiente resta:

$$\begin{array}{r} 0100011 \\ -\ 0\,011100 \\ \hline 0000111 \end{array}$$

Observemos que el pido que se produce en la tercera columna (dígito de peso dos), vale 2, ya que es la base del sistema y siempre se presta la base, por lo que el uno que presta (que es el dígito de peso cinco), queda reducido a cero (porque queda con uno menos) y los anteriores (dígitos de peso tres y cuatro) quedan primero en dos y luego al prestar quedan en uno.

Nota: tanto para la suma como para la resta de números binarios, estos deben tener la misma cantidad de dígitos. Si no tienen, se completan con ceros a la izquierda hasta llegar a la misma cantidad de dígitos.

○ **Resta de números hexadecimales**

$$\begin{array}{r} F\ 2\ 4\ 8\ 9_H \\ -\ A\ B\ D\ F\ C_H \\ \hline 4\ \ 6\ 6\ 8\ D_H \end{array}$$

Observemos que 9 – C no se puede, entonces se pide prestado al del lado, el 8 puede prestar y presta la base (16) y queda con uno menos (en 7). Ahora tenemos 9 + 16 = 25 y le restamos C. Así seguimos hasta el final.

Recién vimos cómo se resta en binario. El sistema binario es el que entiende la computadora, pero debemos decir que la computadora no tiene la posibilidad física de realizar una resta, solo puede sumar. Para restar la computadora usa complementos. El tema Complemento lo veremos en otra sección de este práctico.

♦ **Multiplicaciones**

○ **Multiplicaciones de números binarios**

Usaremos la *TABLA 4* de multiplicaciones para el SNP binario, que nos servirá para poder realizar esta operación.

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

Como ejemplo realizaremos la siguiente multiplicación de números binarios:

				0	1	0	1	1	0
			x		0	1	0	0	1
				0	1	0	1	1	0
			0	0	0	0	0	0	
		0	0	0	0	0	0		
	0	1	0	1	1	0			
0	0	0	0	0	0				
0	0	1	1	0	0	0	1	1	0

El algoritmo del producto en binario es igual que en números decimales; aunque se lleva a cabo con más sencillez, ya que el 0 multiplicado por cualquier número da 0, y el 1 es el elemento neutro del producto.

SISTEMAS DE REPRESENTACIÓN NUMÉRICA DE ENTEROS

En el sistema de numeración binario cualquier número puede representarse tan solo con los dígitos 1 y 0, el signo menos y la coma.

Sin embargo, para ser almacenados y procesados por un computador, no se tiene la posibilidad de disponer del signo y la coma. Para representar los números solo se pueden usar los dígitos 0 y 1. Si utilizáramos solo enteros no negativos su representación sería inmediata.

Dentro de los sistemas de representación numérica tenemos los siguientes:

1- Representación de magnitud con signo.
2- Representación del complemento a la base con signo.
3- Representación del complemento a la base menos uno con signo.

♦ REPRESENTACIÓN DE SIGNO – MAGNITUD

Los números que se utilizan en cálculos científicos se designan por un signo, la magnitud del número y, algunas veces, el punto decimal.

El signo de un número puede considerarse como un conjunto de dos elementos, MÁS y MENOS, que se acostumbra a colocar en la posición más lejana a la izquierda del número. Se utilizan, en general, los símbolos + (más) y – (menos).

Los ordenadores pueden procesar datos solamente en términos de bits, por lo tanto, deberemos encontrar una notación que nos permita representar números binarios en donde se contemple el signo y la magnitud.

Al conjunto de dos elementos más (+) y menos (-), pueden asignarle un código binario de bits. Adoptaremos el **0** (cero) para representar el signo más (**+**) y el **1** (uno) para el signo menos (**-**). Por lo tanto, para representar un número binario de n bits con signo en un registro, necesitamos n+1 bits; n bits para la magnitud y 1 bit para el signo.

El bit de la izquierda es el bit de signo, donde 0 (cero) significa positivo y 1 (uno) significa negativo. Por ejemplo: +18 = 00010010 y -18 = 10010010.

Ésta forma de representación presenta limitaciones:

- La suma y la resta requieren tener en cuenta tanto los signos de los números como sus magnitudes relativas para llevar a cabo la operación.
- Dos representaciones del cero (+0 = 00000000 y -0 = 10000000).

Debido a que esta representación de signo y magnitud no es adecuada para una computadora, en su lugar se usan otras dos formas de representar un número negativo (complemento a la base y a la base menos uno).

Aunque es posible reservar un bit para denotar el signo, la mayoría de las computadoras almacenan los números negativos en la forma de su complemento aritmético.

Hay dos tipos de complemento: el COMPLEMENTO A LA BASE MENOS UNO y el COMPLEMENTO A LA BASE o simplemente COMPLEMENTO. En ambos casos el signo del número viene dado por el dígito más significativo.

- **COMPLEMENTO A LA BASE MENOS UNO O C1**

Dado un número positivo, se puede encontrar su negativo equivalente en complemento a la base menos uno restando a cada uno de los dígitos, incluido el de signo, la base numérica menos uno.

- *En decimal:*
  Ejemplo: el complemento a la base menos 1 del número decimal -22 es 977, (999 - 022 = 977).
- *En binario*: el complemento a la base menos uno en el sistema binario consiste en cambiar los ceros por unos y los unos por ceros del número a complementar.
  Ejemplo: el complemento a la base menos 1 de 0101101 (es el numero decimal + 32) es 1010010 (-32).

- **COMPLEMENTO A LA BASE O C2**

Dado un número positivo en complemento a la base, su negativo equivalente se encuentra restando a ese número 10^x, donde 10 es la base del sistema y x el número de dígitos del número a transformar (incluido el de signo).

- *En decimal:*
  Ejemplo: el complemento a la base del número decimal 035 es 965, (10^3= 1000, 1000 - 035 = 965, donde el 9 es el signo menos).
- *En binario:* para obtener el complemento a la base en el sistema binario (complemento a 2), la regla práctica dice: copiar el número a complementar desde el bit menos significativo hasta que se encuentre el primer uno inclusive y luego cambiar los 1 por 0 y los 0 por 1.
  Ejemplo: el complemento a la base del número binario 0101101 es 1010011.

La Resta Como Suma Del Complemento

Para poner en evidencia la utilidad de los complementos, consideremos el siguiente razonamiento en el sistema decimal:

Sean M y S dos números enteros positivos con igual cantidad de dígitos, digamos 4. Consideremos la siguiente diferencia:

$$D = M - S$$

Sumemos a ambos miembros de la igualdad $10^4 = 10000$

$$D + 10000 = M - S + 10000$$

Esta igualdad se puede escribir también como:

$$D + 10000 = M + (10000 - S) = M + \text{Complemento a la base de S},$$

o también como:

$$D + 10000 = M + (9999 - S) = M + \text{Complemento a la base menos uno de S} + 1.$$

Por lo tanto, podemos calcular la diferencia D ya sea sumando a M el complemento a la base de S, ya sea sumando a M el complemento a la base menos uno de S y luego sumarle 1. En ambos casos debemos restar 10000 al resultado, para obtener el valor de la diferencia D.

- **REGLA PARA RESTAR COMO SUMA DE COMPLEMENTOS, EN BINARIO**

Para el caso del sistema binario podemos enumerar las siguientes reglas para restar como suma de complementos:

1. Evaluar la cantidad de bits necesarios para realizar la operación, teniendo en cuanta el mayor valor a representar.

Para sumar, ambos números deben tener igual cantidad de dígitos. Esto se hace para evitar el **overflow** o **sobreflujo**. Cuando dos números de *n* dígitos cada uno se suman y el resultado ocupa *n + 1* dígitos, decimos que ha ocurrido un sobreflujo. Esto es un problema porque el ancho de los registros en una computadora es finito. Un resultado que contenga *n + 1* bits no pueden acomodarse en un registro con una longitud estándar de *n* bits. La detección del sobreflujo se nota cuando se suman dos números positivos y el resultado da negativo o cuando se suman dos números negativos y el resultado da positivo.

Si ocurre hay que corregirlo agregando a ambos números un nuevo bit de signo.

2. *Los números binarios positivos comienzan en cero, a los negativos se los obtiene complementándolos y comienzan en 1.* Complementar solo aquellos números a ser restados.
3. Efectuar la suma de estos números. El procedimiento es diferente, según se trate de complemento a 1 o a 2.

⇨ **Regla de suma para el complemento a 1**: sumar los dos números, incluyendo el bit de signo, y si hubiera acarreo de la última columna, sume éste a la columna menos significativa.

⇨ **Regla de suma para el complemento a 2:** sumar los dos números, incluyendo el bit de signo, y si hubiera acarreo de la última columna descartarlo.

Nota: en la sección “PROBLEMAS RESUELTOS” de este práctico se muestran ejemplos del tema.

OTROS CÓDIGOS

Código ASCII

Muchas aplicaciones de computadoras digitales requieren el manejo de datos que no solo están formados por números sino también por letras del alfabeto y por ciertos caracteres especiales.

El código binario estándar para los caracteres alfanuméricos es el código ASCII (American Standard Code for Information Interchange).

Èste emplea siete bits para codificar 128 caracteres, como se muestra en la siguiente tabla:

B_6 B_5 B_4								
000	**001**	**010**	**011**	**100**	**101**	**110**	**111**	**$B_3B_2B_1B_0$**
NUL	DLE	SP	0	@	P	,	p	**0000**
SOH	DC1	!	1	A	Q	a	q	**0001**
STX	DC2	“	2	B	R	b	r	**0010**
ETX	DC3	#	3	C	S	c	s	**0011**
EOT	DC4	$	4	D	T	d	t	**0100**
ENQ	NAK	%	5	E	U	e	u	**0101**
ACK	SYN	&	6	F	V	f	v	**0110**
BEL	ETB	‘	7	G	W	g	w	**0111**
BS	CAN	(	8	H	X	h	x	**1000**
HT	EM	)	9	I	Y	i	y	**1001**
LF	SUB	*	:	J	Z	j	z	**1010**
VT	ESC	+	;	K	[	k	{	**1011**
FF	FS	´	<	L	\	l	¦	**1100**
CR	GS	-	=	M	]	m	}	**1101**
SO	RS	.	>	N	^	n	~	**1110**
ST	US	/	?	O	_	o	DEL	**1111**

El código ASCII contiene 94 caracteres gráficos imprimibles y 34 caracteres para funciones de control no imprimibles. Los caracteres gráficos constan de las 26 letras mayúsculas, las 26 letras minúsculas, los 10 números y 32 caracteres especiales como #, * y $.

Los caracteres de control se utilizan para enviar datos y disponer el texto impreso en un formato preestablecido. Entre estos se cuentan con:

NUL	Nulo	DEL	Escape de enlace de datos
SOH	Comienzo de encabezado	DC1	Control de dispositivo 1
STX	Comienzo de texto	DC2	Control de dispositivo 2
ETX	Fin de texto	DC3	Control de dispositivo 3
EOT	Fin de transmisión	DC4	Control de dispositivo 4
ENQ	Consulta	NAK	Reconocimiento negativo
ACK	Reconocimiento	SYN	Inactivo sincrónico
BEL	Campana	ETB	Fin de bloque de transmisión
BS	Retroceso	CAN	Cancelar
HT	Tabulador horizontal	EM	Fin de medio
LF	Alimentación de línea	SUB	Sustituto
VT	Tabulador vertical	ESC	Escape
FF	Alimentación de forma	FS	Separador de archivo
CR	Retorno de carro	GS	Separador de grupo
SO	Tecla de mayúscula oprimida	RS	Separador de registro
SI	Tecla de mayúscula sin oprimir	US	Separador de unidad
SP	Espacio	DEL	Borrar

Los siete bits del código se representan mediante B_0 a B_6 donde B_6 es el bit más significativo.

$$B_6\ B_5\ B_4\ B_3\ B_2\ B_1\ B_0$$

Ejemplo: la letra A se representa como 1000001 (columna 100, fila 0001).

El código ASCII es un código de 7 bits, pero la mayoría de las computadoras manipula una cantidad de 8 bits como una unidad única llamada byte. Por lo tanto, con mucha

frecuencia, los caracteres ASCII se almacenan uno por byte. El bit extra en ocasiones se utiliza para otros propósitos, dependiendo de la aplicación.

Código De Tétradas Para Uso Aritmético

En la siguiente tabla se muestran los más importantes códigos de tétradas. En esta materia solo estudiaremos el BCD.

	CÓDIGOS			
TÉTRADAS	**BCD (8421)**	**AIKEN (2421)**	**EXCESO 3 (S T I B I T Z)**	**G R A Y**
0000	0	0	Pseudotétradas	0
0001	1	1	Pseudotétradas	1
0010	2	2	Pseudotétradas	3
0011	3	3	0	2
0100	4	4	1	7
0101	5	Pseudotétradas	2	6
0110	6	Pseudotétradas	3	4
0111	7	Pseudotétradas	4	5
1000	8	Pseudotétradas	5	Pseudotétradas
1001	9	Pseudotétradas	6	Pseudotétradas
1010	Pseudotétradas	Pseudotétradas	7	Pseudotétradas
1011	Pseudotétradas	5	8	Pseudotétradas
1100	Pseudotétradas	6	9	8
1101	Pseudotétradas	7	Pseudotétradas	9
1110	Pseudotétradas	8	Pseudotétradas	Pseudotétradas
1111	Pseudotétradas	9	Pseudotétradas	Pseudotétradas

En la primera columna se han listado todas las 2^4=16 posibles combinaciones de ceros y unos que permite una tétrada. En las restantes columnas puede apreciarse las cifras decimales correspondientes a estas combinaciones en los diferentes códigos de tétradas. La tétrada 0110 equivale, por ejemplo, en el código BCD a la cifra decimal 6.

Debido a que de las 16 tétradas que se forman, solamente necesitamos 10 para las diez cifras decimales, en cada código por tétradas, 6 de ellas permanecen sin ser usadas. Estas reciben el nombre de PSEUDOTETRADAS (PT).

De acuerdo con las correspondencias de la tabla anterior, el número decimal 8125, que en el sistema binario natural es el 1111110111101, tendrá la siguiente representación en código BCD:

1000	0001	0010	0101	(4 tétradas)
8	1	2	5	(Nº decimal)

Es decir 8125_{10}= $1000000100100101_{(BCD)}$

Se observa que, en este código, la representación de un número requiere igual o mayor número de bits que en el sistema binario natural, aunque una vez conocida la correspondencia binaria de cada dígito decimal en BCD, es más rápida la lectura y escritura de un número.

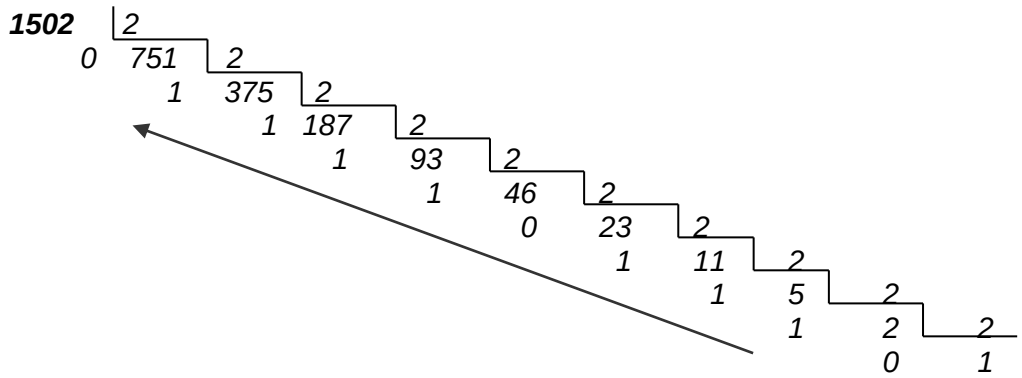
PROBLEMAS RESUELTOS

1. *Cambios de una base a otra:*

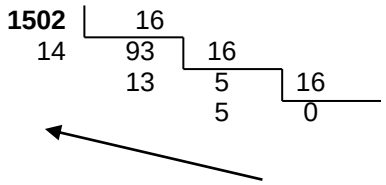
$\mathbf{010110101_2}$ = $0x\ 2^8 + 1x\ 2^7 + 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
= 0 + 128 + 0 + 32 + 16 + 0 + 4 + 0 + 1
= $\mathbf{181_{10}}$

$\mathbf{B5_{16}}$ = $(11 \times 16^1) + (5 \times 16^0)$
= 176 + 5
= $\mathbf{181_{10}}$

2. *Ahora veamos cómo realizar el proceso inverso, es decir pasar un número en base 10 a bases 2 y 16. Primero efectuamos la división sucesiva del número en la base a la que lo queremos llevar:*



$\mathbf{1502_{10} = 010111011110_2}$

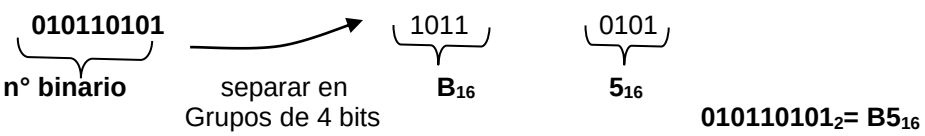


$\mathbf{1502_{10} = 5DE_{16}}$

Como vemos, puede llegar a ser bastante tedioso la utilización de esta técnica para el caso de números largos, lo que se presta a equivocaciones.
Por ello existen ***técnicas reducidas*** *que permiten pasar de un sistema a otro.*

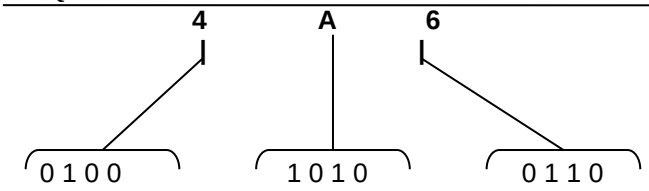
3. *Cambios de base usando técnicas reducidas:*

Si tenemos el n° 010110101_2, podemos pasarlo a la base 16.
Lo que se hace es dividir los dígitos en grupos de 4 de derecha a izquierda. Si al llegar al extremo de la izquierda no se llega a completar el grupo, sencillamente se añaden tantos ceros como sea necesario.



$\mathbf{010110101_2 = B5_{16}}$

El paso contrario, ir de hexadecimal a binario es igualmente simple.



$4A6_{16} = 010010100110_2$

Recuerde:
Una técnica reducida para el pasaje de un número binario a decimal es la de multiplicar los pesos de acuerdo a la posición del número por el dígito y realizar la posterior suma.

Cambio de base de binario a decimal:

128 64 32 16 8 4 2 1
0 1 0 0 1 1 0 1 64x1 + 8x1 + 4x1 + 1x1 = **77_{10}**

4. Ejemplos de sumas y restas:

Dejamos como inquietud para el estudiante la comprobación de los resultados, pasando los valores a decimal.

$0\,1\,0\,0\,1\,1\,1_2$ + $0\,0\,1\,0\,1\,1\,1_2$ = $0\,1\,1\,1\,1\,1\,0_2$

$0\,0\,1\,1\,1\,1\,0\,0\,0\,0_2$ + $0\,0\,1\,1\,0\,0\,0\,0\,1\,1_2$ = $0\,1\,1\,0\,1\,1\,0\,0\,1\,1_2$

$FEA23_H$ + $ADE27_H$ = $1AC844_H$

$ABDF4_H$ + $EEFD1_H$ = $19ADC5_H$

$1\,1\,1\,0\,1\,0\,1_2$ - $1\,0\,1\,1\,1\,1\,1_2$ = $0\,0\,1\,0\,1\,1\,0_2$

$1\,1\,0\,0\,1\,1\,0\,0_2$ - $1\,0\,1\,1\,1\,0\,1\,1_2$ = $0\,0\,0\,1\,0\,0\,0\,1_2$

$EBA94_H$ - $CFED9_H$ = $1BBBB_H$

5. Dado el n° con bit de signo en C1, representarlo en decimal:

N° negativo →1 1 0 1 → se sabe que es un n° negativo, pero no su equivalente en decimal.
N° positivo → 0 0 1 0 → se obtiene complementando a 1 el número anterior.
$1101_2 = -2_{10}$

6. Sumar dos números positivos:

Supongamos que tenemos que sumar 0010 (2_{10}) y 0100 (4_{10}). Efectuamos la suma:

$0\,0\,1\,0$ + $0\,1\,0\,0$ = $0\,1\,1\,0$ → Obtenemos

2 + 4 = 6_{10}

Digamos que queremos restar 3 al número 5. Para tal efecto, la operación de resta será tratada como una suma en donde uno de los sumandos es negativo.

En Complemento a 1:

5_{10} → va el nº positivo → 0101
-3_{10} → se pone el nº en C1 de +3 (0011) → + 1100
2_{10} (1) 0001
+ 1
se suma el ultimo acarreo *0010* → *resultado*

En complemento a 2:

5_{10} 0101
-3_{10} 1101
2_{10} (1) 0010 *resultado*
se descarta

7. Realice las operaciones 70 + 80 y – 70 – 80.

	128 64 32 16 8 4 2 1		***En C1***	***En C2***
+ 70	0 1 0 0 0 1 1 0	- 70	1 0 1 1 1 0 0 1	1 0 1 1 1 0 1 0
+ 80	0 1 0 1 0 0 0 0	- 80	1 0 1 0 1 1 1 1	1 0 1 1 0 0 0 0

70 + 80 = 150		Corrección, con nuevo bit de signo
01000110 + 01010000 10010110	Ver bits de signo. O*curre overflow (positivo + positivo da negativo)*	001000110 + 001010000 010010110 = + 150

En C1		
(-70)+(-80)= - 150		Corrección, con nuevo bit de signo
10111001 + 10101111 01101000 ~~1~~ + 1 01101001	Ver bits de signo. O*curre overflow (negativo + negativo da positivo)*	110111001 + 110101111 101101000 ~~1~~ + 1 101101001 = - 150

En C2		
(-70)+(-80)= - 150		Corrección, con nuevo bit de signo
10111010 + 10110000 ~~1~~ 01101010	Ver bits de signo. O*curre overflow (negativo + negativo da positivo)*	110111010 + 110110000 ~~1~~ 101101010 = - 150

PROBLEMAS PROPUESTOS

1) a) Usando los símbolos @, # y * construya un sistema de numeración posicional de base tres.

b) Dado el número ***@#@*** del SNP antes definido, determinar su valor en decimal, usando el concepto de digito por base de origen elevado a la posición.

2) Dados los siguientes números en base 2, pasarlos a base 10 y 16:

a) 01001101_2 b) 01110010_2 c) 011011_2 d) 0111_2

3) Dados los siguientes números en base 16, pasarlos a base 2 y 10:

a) $A01_{16}$ b) $507F_{16}$ c) $39D_{16}$ d) 801_{16}

4) Dados los siguientes números en base 10 pasarlos a base 2 y 16:

a) 359_{10} b) 101_{10} c) 39_{10} d) 86_{10}

5) Realice las siguientes sumas:

Base 2: a) 011011 + 010101 b) 0110 + 0101 + 01111

Base 16: c) FF0 + 1457 d) 5C7 + B2

6) Convertir los siguientes sumandos de decimal a binario y resolver las sumas indicadas, operando en binario, en complemento a 1 y a 2.

25 + 12 25 + (-12) (-25) + 12 (-25)+(-12)

7) Resolver las siguientes operaciones, trabajando en binario. Los números negativos expresarlos en complemento a 2. Al resultado pasarlo a la base indicada.

a) $9AF_{(16)} - 543_{(10)} + 011001101_{(2)} =$ -------- (16)
b) $F22_{(16)} - 702_{(10)} + 010010011_{(2)} =$ -------- (10)
c) $4E_{(16)} + 001011000_{(2)} + 122_{(5)} - 010010_{(2)} =$ -------(10)

8) Realice las siguientes multiplicaciones binarias. Compruebe los resultados pasando al sistema decimal.
a) 01000101 x 01011
b) 01111011 x 010011

9) Represente el número decimal 8620 en: a) binario, b) ASCII c) BCD.
10) Pase el numero 100101110010 , que está en BCD, a decimal, binario y hexadecimal.

Nota: se recomienda usar las técnicas reducidas de conversión.

PROBLEMAS COMPLEMENTARIOS

1) Dados los siguientes números en base 2 y 16 pasarlos a base 10:

Base 2:	01000100	0110001	011010	01010
Base 16:	1011	C72EA	A19	DF

2) Dados los siguientes números en base 10, pasarlos a base 2 y 16.

a) 651 b) 52

3) Realice las siguientes restas:

Base 2:	011011 - 010101
Base 16:	81A - 10F

4) Convertir los siguientes sumandos de decimal a binario y resolver las sumas indicadas operando en binario, en complemento a 1 y a 2.

34 + 17 34 + (-17) (-34) + 17 (-34) + (-17)

5) Realice las operaciones 35 + 40 y (-35) + (-40) con números binarios en complemento a 1. Utilice 7 bits para la representación del número y su signo. Muestre que el sobreflujo (overflow) ocurre en ambos casos. Interprete los resultados.

6) Los números binarios que se enumeran abajo tienen un bit de signo. Los números negativos están en la forma de su complemento a 1 con signo. Realice las operaciones indicadas y verifique los resultados.

001110+111010	010101-000111
101011+111000	101011-100110
010101 + 000011	001010 - 111001
111001 + 001010	111001 - 00101

7) Repita el problema anterior suponiendo que los números negativos están en complemento a 2 con signo.

8) Muestra la configuración de bits que representa el número decimal 295 en: a) binario, b) BCD y c) ASCII.