SISTEMAS DE NUMERACIÓN

Al conjunto de reglas y convenios que permiten la representación de los números mediante varios signos, o varias palabras, se le llama SISTEMA DE NUMERACIÓN. Conocidos son EL ROMANO y EL DECIMAL.

El primero descompone el numero en suma o diferencia de otros varios, cada uno de los cuales está representado por un símbolo especial : I, V, X, L, C, D, M, ...

El segundo, en vez de introducir nuevos símbolos, utiliza el principio del VALOR RELATIVO, es decir una misma cifra representa valores distintos según el lugar que ocupa.

Los sistemas con los mismos principios que el decimal, son los únicos que tienen interés aritméticos, pues en ellos, las leyes y propiedades no presentan modificaciones. Se puede establecer un sistema de numeración, tomando como base un número natural mayor que 1. La computadora no utiliza el sistema decimal, porque necesitaría componentes que tuvieran diez estados. Es por ello que trabaja con el sistema numérico de dos símbolos que son fácilmente representables en las componentes biestables.

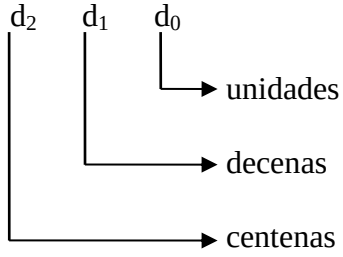
A los n símbolos que componen el sistema en base n > 1 se les suele llamar VALOR ABSOLUTO y los valores posicionales están dados por las potencias de la base del sistema.

Para facilitar el registro y procesamiento de datos, las computadoras utilizan códigos basados en variaciones del SISTEMA BINARIO. Resultan muy prácticos en informática, en donde se utilizan corrientemente expresiones de 8 y 16 elementos binarios, el SISTEMA OCTAL y el SISTEMA HEXADECIMAL.

LOS NÚMEROS NATURALES Y LOS SISTEMA DE NUMERACIÓN

Si se trabaja en el sistema decimal, un número puede ser descompuesto en potencias de 10.

$N = dn \quad ... \quad d_2 \quad d_1 \quad d_0$



$$N = \sum_{i=1}^{n} d_i * 10^i$$

Ejemplo: $845 = 8 * 10^2 + 4 * 10 + 5 * 10^0$

En general, si el numero es de n cifras y llamando B a la base, se tiene:

$$N = \alpha_{n-1} * \beta^{n-1} + \alpha_{n-2} * \beta^{n-2} + \ldots\ldots\ldots\ldots + \alpha_1 * \beta^1 + \alpha_0$$

$$N = \alpha_{n-1}\alpha_{n-2}\ldots\ldots\ldots\alpha_1\ \alpha_0$$

Por ejemplo si B = 5, usando los símbolos 0, 1, 2, 3, 4 resulta:

$2432_{5} = 2 * 5^{3} + 4 * 5^{2} + 3 * 5^{1} + 2 * 5^{0} = 367_{10}$

1440302_{5} = ...= 30702_{10}

Las tablas de la suma y del producto del sistema en base 5 son:

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	10
2	2	3	4	10	11
3	3	4	10	11	12
4	4	10	11	12	13

*	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	11	13
3	0	3	11	14	22
4	0	4	13	22	31

Leibnitz, en el siglo XVII introduce la NUMERACIÓN BINARIA. Presentamos también la numeración OCTAL Y HEXADECIMAL.

DECIMAL	BINARIO	OCTAL	HEXADECIMAL
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

Como sólo se permite un símbolo para cada valor absoluto, en el sistema hexadecimal que requiere 16 valores se utilizan los 10 dígitos del sistema decimal y las 6 primeras letras del alfabeto.

CONVERSIÓN DE UN SISTEMA A OTRO

i)De decimal a……

Se divide el número entre la base del sistema, hasta que el cociente llegue a cero. El resto de cada división se registra y leídos en orden inverso al de aparición, constituyen el número en la nueva base.

Ejemplos:

a) N = 57
N_2 = 111001
N_{16} = 39
N_8 = 71

b) N = 678
N_2 = 1010100110
N_{16} = 2 A6
N_8 = 1246

ii) De base B a Decimal.

$$N_B = \alpha_{n-1}\alpha_{n-2}.....\alpha_1\ \alpha_0$$

$$N_{10} = \alpha_{n-1} * B^{n-1} + \alpha_{n-2} * B^{n-2} + ... + \alpha_1 * B^1 + \alpha_0 * B^0$$

Ejemplos:

N_2 = 111001

$N = 1*2^5 + 1*2^4 + 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0 = 32 + 16 + 8 + 1 = 57$

N_{16} = 39

$N = 3 * 16 + 9 = 57$

N_8 = 71

$N = 7 * 8 + 1 = 56 + 1 = 57$

N_{16} = 2 A6

$N = 2 * 16^2 + 10 * 16^1 + 6 = 2 * 256 + 160 + 6 = 512 + 166 = 678$

iii) De binario a hexadecimal y a octal y viceversa:

Entre el sistema binario y el hexadecimal existe la relación directa de 4 a 1 (2^4 a 2^1). Así, cada cuatro dígitos binarios se obtiene un dígito hexadecimal y cada dígito hexadecimal se convierte en cuatro dígitos binarios. Entre el sistema binario y el octal se procede de manera análoga con la vinculación de 3 a 1 (2^3 a 2^1).

Ejemplos:

N = 678

N_2 = 1010100110

N_{16} = 2 A6

N_{16} = 39

N_2 = 00111001

N_2 = 1010100110

N_8 = 1246

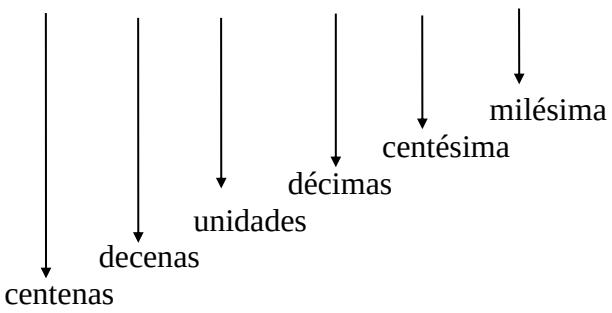
N_8 = 351

N_2 = 011101001

LOS NUMEROS DECIMALES Y LOS SISTEMAS DE NUMERACION

Un número decimal consta de un arreglo de dígitos y de un punto decimal. La forma general y su interpretación en base 10 es la siguiente:

dn ..... d_2 d_1 d_0 . d_{-1} d_{-2} d_{-3} .... d_{-k}



$$N = \sum_{i=-k}^{n} d_i * 10^i$$

Para trabajar en el sistema binario es útil conocer las primeras potencias de "2" pues indican "el peso" de cada cifra de un número binario, o sea el valor en base 10 que tiene un "1" de acuerdo a la posición que ocupa en el número.

2^{-n}	n	2^{n}
1	0	1
0.5	1	2
0.25	2	4
0.125	3	8
0.0625	4	16
0.03125	5	32
0.015625	6	64
0.0078125	7	128
0.00390635	8	256
0.001953125	9	512
0.0009765625	10	1024
0.00048828125	11	2048
0.000244140625	12	4096

La representación binaria resulta muy cómoda para la computadora pero no ocurre lo mismo para el usuario que prefiere una forma mas condensada de escritura. Los números decimales también pueden representarse en sistema octal y hexadecimal.

Decimal	Binario	Octal	Hexadecimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
30	11110	36	1E
40	101000	50	28
50	110010	62	38
60	111100	74	3C
70	1000110	106	46
80	1010000	120	50
90	1011010	132	5ª
100	1100100	144	64
200	11001000	310	C8
300	100101100	454	12C
400	110010000	620	190
500	111110100	764	1F4
600	1001011000	1130	258
700	1010111100	1274	26C
800	1100100000	1440	320
900	1110000100	1604	384
1000	1111101000	1750	3E8
2989	101110101101	5655	BAD

CONVERSIÓN DE UNA BASE EN OTRA

Para convertir un número binario a octal, se lo divide en grupos de tres bits, de modo que los tres bits situados inmediatamente a la izquierda (o derecha) del punto decimal (a menudo denominado punto binario) formen un grupo; y los tres bits situados inmediatamente a la izquierda (o derecha) otro grupo; y así sucesivamente. Cada grupo de tres bits puede formar un único dígito octal, de 0 al 7, de acuerdo a la conversión que se dá a las primeras líneas del cuadro anteriormente ilustrado. Puede ser necesario añadir uno o dos ceros a la izquierda o a la derecha para completar un grupo de tres bits.

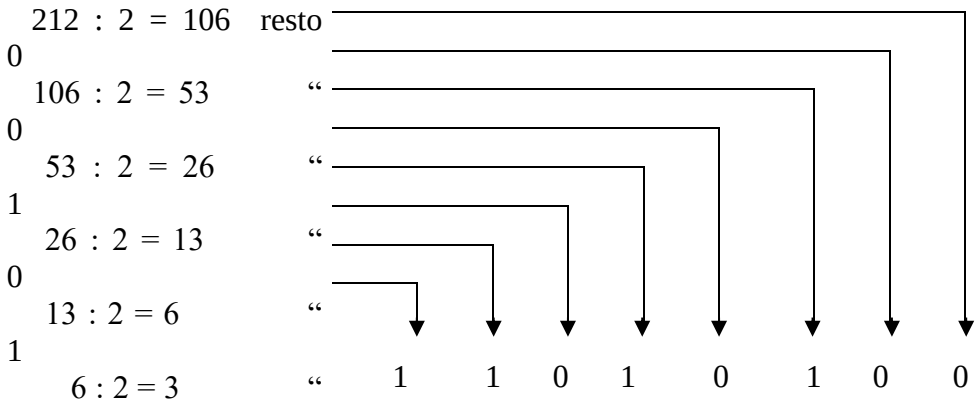
La conversión de octal en binario es igualmente sencilla, cada dígito octal se reemplaza por su binario equivalente de tres bits.

La conversión hexadecimal a binario es esencialmente igual que la de octal a binario, excepto que cada dígito hexadecimal se corresponde con un grupo de cuatro bits en lugar de tres.

Un método para convertir números decimales a binarios consiste en considerar por separado la parte entera y la parte decimal de un número dado. En el siguiente ejemplo se describe el mecanismo del método:

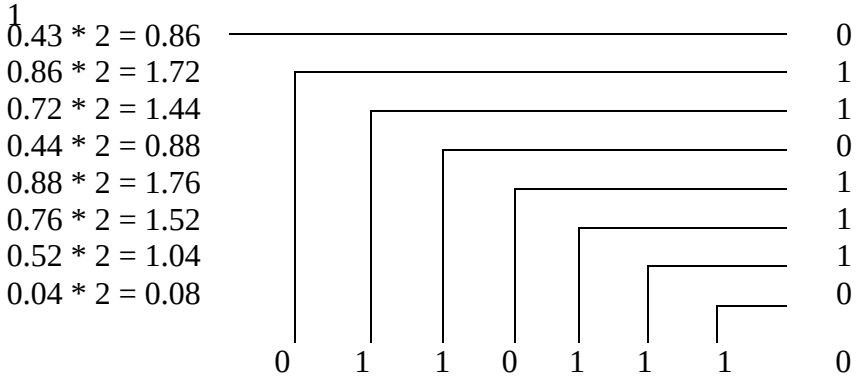
Si N_{10} = 212.43 en binario es:

Parte entera:



Parte decimal:

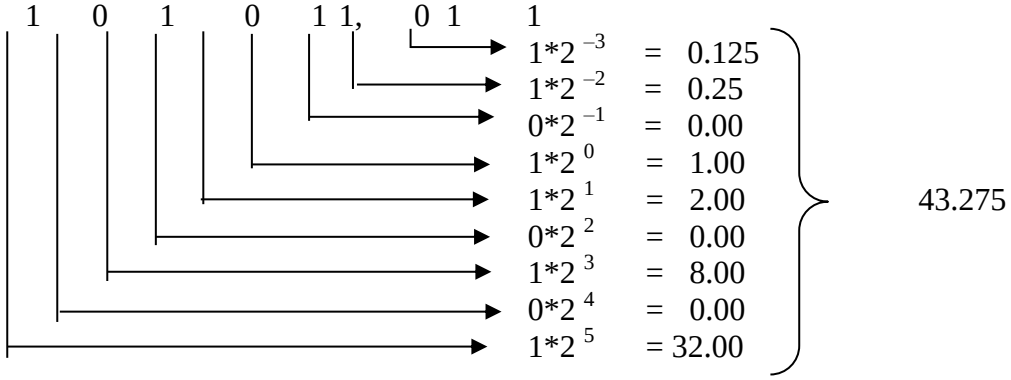
Peso Binario



$0.43 = 0.01101110$

$212.43_{(10)} = 1010100 \,.\, 01101110_{(2)}$

El método para convertir un número binario a su equivalente en base 10 se describe en el siguiente ejemplo:



OPERACIONES

La adición es la operación fundamental efectuada por una computadora. Cuando la computadora resta, multiplica o divide, lo hace sumando y desplazando.

Operaciones en el sistema binario

Adición:

+	0	1
0	0	1
1	1	0

→ Y se lleva 1

Sustracción:

Con 1 prestado de la columna de la izquierda

-	0	1
0	0	1
1	1	0

Ejemplos:

9	1001	13	1101
+	+	-	-
5	0101	6	0110
= 14	= 1110	= 7	= 0111

Observación: aunque la sustracción se realiza en la forma indicada, las computadoras la ejecutan formando un complemento del sustraendo y sumando éste número al minuendo. Los procedimientos para completar los veremos en las representaciones binarias de números negativos.

Multiplicación:

*	0	1
0	0	0
1	0	1

Ejemplo: 6 * 5 = 30

```
   110
 * 101
 -----
   110
+ 000
 ----
 110
 ---
=11110
```

División:

La división binaria se ejecuta en la misma forma que la división decimal aplicando las reglas de la multiplicación y la división binaria.

Ejemplo:

```
 30 | 5
-30   6
 0
```

```
11110 | 101
 101    110
 0101
  101
 00000
```

Observación: En realidad las computadoras efectúan la multiplicación por medio de sumas, y la división por medio de sumas de complementos.

CÁLCULOS HEXADECIMALES Y OCTALES

Las operaciones de la aritmética se efectúan con números hexadecimales u octales, siguiendo las reglas básicas del sistema decimal. La adición hexadecimal no produce un acarreo hasta que se excede del valor decimal 15 y la adición octal al exceder el valor decimal 7. La sustracción hexadecimal pide prestado de la posición de la izquierda el valor 16 y la octal el valor 8.

Ejemplos de adición:

i) S. Hexad.

```
 8A    F    2 A    A     A    9 A
+7   + 3   + F   + B 1  + C 6 + 5 7
 F    D     1E     D B   170    F 1
```

ii) S. Octal

```
   5     36      454
+ 12   + 24    + 620
  17     62      1274
```

Ejemplos de sustracción:

i) S. Decimal

```
  13     19
- 8    - 8
  5      11
```

ii) S. Hexad.

```
  B     13
- 8   - 8      al pedir es 19
  5     B
```

iii) S. Octal

$$\begin{array}{rr} 15 & 23 \\ -\ 10 & -\ 10 \\ \hline 5 & 13 \end{array}$$

Números Binarios Negativos

A continuación se describen algunos de los sistemas más usados para representar números binarios negativos:

a) El primero se llama de **módulo y signo** , en el cual el bit de la izquierda del número binario es el bit del signo (0 es + y 1 es -) y los restantes bits contienen el valor absoluto del número.

b) El segundo sistema se llama **complemento a 1** , también tiene un bit de signo (0 es + y 1 es -). Para obtener el complemento a 1 de un número, se reemplazan los ceros por unos, y los unos por ceros. Esto vale incluso para el bit de signo.

c) El tercer sistema se llama **complemento a 2** , también tiene un bit de signo (0 es + y 1 es -). Para obtener el complemento a 2 de un número, se pueden aplicar alguno de los dos métodos que se exponen a continuación:

1.- Primero cada uno se reemplaza por un cero y cada cero por un uno como en el complemento a 1. En seguida se le suma 1 al resultado.

Ejemplo:

$$\begin{array}{ll} 00000101 & (= +5) \\ 11111010 & (-5 \text{ en el compl. a } 1) \\ +\qquad 1 & \\ \hline 11111011 & (-5 \text{ en el compl. a } 2) \end{array}$$

2.- Analizar el número dado de derecha a izquierda. Hasta la aparición de primer 1, inclusive se copia el número tal cual. Los bits que siguen al primer 1, se cambian 0 por 1 y 1 `por 0.

Ejemplo:

$C^{(2)}$ 1001100100 = 0110011100

El siguiente cuadro muestra los números negativos representado en los tres sistemas:

N DECIMAL	N BINARIO	- N MOD. Y SIG	- N COMP. A 1	- N COMP. A 2
0	00000000	10000000	11111111	00000000
1	00000001	10000001	11111110	11111111
2	00000010	10000010	11111101	11111110
3	00000011	10000011	11111100	11111101
4	00000100	10000100	11111011	11111100
5	00000101	10000101	11111010	11111011
6	00000110	10000110	11111001	11111010
7	00000111	10000111	11111000	11111001
8	00001000	10001000	11110111	11111000
9	00001001	10001001	11110110	11110111
10	00001010	10001010	11110101	11110110
20	00010100	10010100	11101011	11101100
30	00011110	10011110	11100001	11100010
40	00101000	10101000	11010111	11011000
50	00110010	10110010	11001101	11001110
60	00111100	10111100	11000011	11000100
70	01000110	11000110	10111001	10111010
80	01010000	11010000	10101111	10110000
90	01011010	11011010	10100101	10100110
100	01100100	11100100	10011011	10011100
127	01111111	11111111	10000000	10000001
128	----------	----------	----------	10000000

Tanto los números en **módulo y signo** como en **complemento a 1** tienen dos representaciones del 0: un +0 y un –0.

Ejemplo: Para binarios de 8 bits:
+0 = 00000000 y -0 = 11111111

Usando números en **complemento a 1**, se pueden representar todos los enteros comprendidos entre:

(decimal)	-127..... –0	+0.....127
(binario)	10000000.....-11111111	00000000.....01111111

En cambio los números en **complemento a 2**, tienen una única representación del 0:
-0 = +0 = 00000000
esto se debe a que el complemento a 2 de 00000000 es 00000000
Usando números en complemento a 2, se pueden representar todos los enteros comprendidos entre:

(decimal)	-128.....0	127
(binario)	10000000.....00000000.....	01111111

Observe que el rango de los números es asimétrico, es decir, que hay un número negativo que carece de su correspondiente número positivo.

Lo ideal sería un sistema de codificación con dos propiedades:
1.- Que haya una única representación del cero.
2.- que haya exactamente tantos números positivos como negativos.

El problema es que todo conjunto con tantos números positivos como negativos y solo un cero, tiene un número impar de miembros, mientras que m bits permiten 2^m combinaciones que siempre es un número par. Por lo tanto siempre habrá una redundancia o una falta: redundancia cuando se representa un mismo número con dos secuencias de bits distintas, o falta cuando uno de los dos subconjuntos (positivos o negativos) tienen un miembro mas que el otro.

La aplicación de éstos sistemas de codificación de números negativos permite resolver las operaciones de resta usando el algoritmo de suma.

Para ello, se debe obtener previamente el complemento a 2 del sustraendo.

Ejemplos: siguiendo con los binarios de 8 bits, resolver -3 – (-2) = -1

Mediante el algoritmo de la resta:

$$\begin{array}{r} 11111101\ (=-3) \\ -\ 11111110\ (=-2) \\ \hline = 11111111\ (=-1) \end{array}$$

Como suma complementando a 2 el sustraendo:

$$\begin{array}{r} 11111101\ (=-3) \\ +\ 00000010\ \ (=2 = C^{(2)}\ ) \\ \hline = 11111111 \end{array}$$

La naturaleza finita de las computadoras hace que los resultados de algunos cálculos sean incorrectos por problemas de **desbordamiento**. Esto ocurre cuando el resultado de una operación de un número cae fuera del rango de los valores que puede manejar la máquina. Casi todos los equipos comunican ésta situación al operador mediante un mensaje de **overflow**.-