Sistema binario.

El **sistema binario** es un sistema de numeración en el que los números se representan utilizando solamente las cifras cero y uno (0 y 1). Es por tanto un sistema de numeración de **base 2** (ya que utiliza dos dígitos diferentes), a diferencia del que comúnmente usamos, el decimal, que es de **base 10** ya que utiliza los dígitos del 0 al 9.

Es el que se utiliza en los ordenadores, pues trabajan internamente con dos niveles de voltaje (encendido 1, apagado 0), por lo que su sistema de numeración natural es el sistema binario.

Un bit es un dígito del sistema de numeración binario, y por lo tanto puede representar o bien un 1 o un 0. El bit es la unidad mínima de información empleada en informática.

Con él, podemos representar dos valores cuales quiera, como verdadero o falso, abierto o cerrado, blanco o negro, norte o sur, masculino o femenino, rojo o azul, etc. Basta con asignar uno de esos valores al estado de "apagado" (0), y el otro al estado de "encendido" (1).

BIT	COLOR
0	rojo
1	verde

Si queremos representar mas valores por ejemplo 4 (rojo, verde, azul, amarillo) necesitamos más bits, en este caso 2.

PRIMER BIT	SEGUNDO BIT	COLOR
0	0	rojo
0	1	verde
1	0	azul
1	1	amarillo

Por lo general con **n bits** podemos representar **2^n valores**.

Valor de posición.

En cualquier sistema de **numeración posicional**, por ejemplo el binario o el decimal, el valor de los dígitos depende del lugar en el que se encuentren.

En el sistema decimal, por ejemplo, el dígito 5 puede valer 5 si está en la posición de las unidades, pero vale 50 si está en la posición de las decenas, y 500 si está en la posición de las centenas. Generalizando, cada vez que nos movemos una posición hacia la izquierda el dígito vale 10 veces más.

$$\sum \text{Digito} * \text{Base}^{\text{posición}}$$

Descomponiendo el número decimal 523 según la posición de sus dígitos tendríamos (el dígito más a la derecha es la **posición 0**):

$$523_{10}=5*10^{2}+2*10^{1}+3*10^{0}$$

Con un número binario actuamos de la misma forma teniendo en cuenta que su **base es 2**. Así pues el número 10110 se descompone de la siguiente forma:

$$10110_{2}=1*2^{4}+0*2^{3}+1*2^{2}+1*2^{1}+0*2^{0}=22$$

Por lo tanto el número 10110_2 es la representación binaria correspondiente del número 22_{10} decimal. Notar que los subíndices nos ayudan a indicar que representación estamos utilizando.

Binario a decimal.

Como hemos visto para pasar de un número binario a su correspondiente número decimal debemos sumar el valor posicional de cada uno de sus dígitos.

También podemos representar números reales (con parte fraccionaria). Para ello los dígitos a la izquierda de la coma los numeraremos con la posición habitual, empezando de 0, y los dígitos de la derecha de la coma los numeraremos comenzando en -1..-2..-3 etc.

$$101'11_{2}=1*2^{2}+0*2^{1}+1*2^{0}+1*2^{-1}+1*2^{-2}=5'75$$

Decimal a binario

Dado un número decimal (es decir cuya base es 10) para obtener su correspondiente representación binaria hay que:

Dividir el número del sistema decimal entre **2**, cuyo resultado **entero** se vuelve a dividir entre 2, y así sucesivamente. Ordenados los **restos**, del último al primero, este será el número binario que buscamos.

Para transformar el número decimal 131_{10} en binario el método es muy simple:

131 dividido entre 2 da 65 y el resto es igual a 1
65 dividido entre 2 da 32 y el resto es igual a 1
32 dividido entre 2 da 16 y el resto es igual a 0
16 dividido entre 2 da 8 y el resto es igual a 0
8 dividido entre 2 da 4 y el resto es igual a 0
4 dividido entre 2 da 2 y el resto es igual a 0
2 dividido entre 2 da 1 y el resto es igual a 0
1 dividido entre 2 da 0 y el resto es igual a 1

-> Ordenamos los restos, del último al primero: $\mathbf{10000011}_2$ que es la representación binaria del 131_{10}

Decimal real a binario

Para transformar un número real del sistema decimal al sistema binario como por ejemplo el $3'3125_{10}$ se procede de la siguiente manera:

1. Transformamos la parte entera utilizando el procedimiento anterior de divisiones.
2. Pasamos a transformar la parte fraccionaria:
    a. Multiplicamos la fraccionaria por 2 hasta llegar a obtener como resultado 1.
        i. Si el resultado de la multiplicación es menor a 1 (por ejemplo 0'25) anotamos un 0 y seguimos multiplicando.
        ii. Si el resultado de la multiplicación es mayor a 1 (por ejemplo 1'25) anotamos un 0 y seguimos multiplicando la parte fraccionaria (en este caso 0'25).
        iii. Si el resultado es 1 paramos.

Siguiendo estos pasos el número $3'3125_{10}$ se transformaría de la siguiente manera:

Paso 1

3 dividido entre 2 da 1 y el resto es igual a 1 1 dividido entre 2 da 0 y el resto es igual a 1 Ordenando los restos la parte entera es: $\mathbf{11}_2$

Paso 2

0,3125 x 2 = 0,625 => 0 0,625 x 2 = 1,25 => 1 0,25 x 2 = 0,5 => 0 0,5 x 2 = 1 => 1 En orden: 0101_2

Por lo tanto el número $\mathbf{11'0101}_2$ es la representación binaria de $3'3125_{10}$

Suma de números Binarios

Las posibles combinaciones al sumar dos bits son:

- 0 + 0 = 0
- 0 + 1 = 1
- 1 + 0 = 1
- 1 + 1 = (1)0 al sumar 1+1 siempre nos llevamos 1 a la siguiente operación.

$$\begin{array}{rr} & 10011000_2 \\ + & 00010101_2 \\ \hline & 10101101_2 \end{array}$$

Resta de números Binarios

Las posibles combinaciones al restar dos bits son:

- 0 - 0 = 0
- 1 - 0 = 1
- 1 - 1 = 0
- 0 - 1 = 1 (tomando una unidad prestada de la posición siguiente)

$$\begin{array}{r} 10001_2 \\ -01010_2 \\ \hline 00111_2 \end{array} \qquad \begin{array}{r} 11011001_2 \\ -10101011_2 \\ \hline 00101110_2 \end{array}$$

Producto de números Binarios

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

Por ejemplo, multipliquemos 10110_2 por 1001_2:

$$\begin{array}{r} 10110_2 \\ 1001_2 \\ \hline 10110 \\ 00000 \\ 00000 \\ 10110 \\ \hline 11000110_2 \end{array}$$

División de números binarios

La división en binario es similar a la decimal, la única diferencia es que a la hora de hacer las restas, dentro de la división, estas deben ser realizadas en binario.

Para dividir 100010010_2 entre 1101_2:

```
 100010010₂ |1101₂
            ------
- 0000       010101₂
--------
 10001
- 1101
--------
  01000
 - 0000
 --------
   10000
  - 1101
  --------
    00111
   - 0000
   --------
     01110
    - 1101
    --------
     00001
```

Sistema Octal

El sistema octal tiene base 8, es decir utiliza los dígitos del 0 al 7. Por ejemplo un número en representación octal sería 347_8 (con el subíndice indicamos que es octal para distinguirlo de uno decimal).

Para pasar de base 8 a base 10 se desarrolla el polinomio de potencias de la base:

$$746'12_8=7*8^2+4*8^1+6*8^0+1*8^{-1}+2*8^{-2}$$

El interés de este sistema radica en que la base del octal (8) es potencia de la base del binario 2^3 y por lo tanto la conversión binario octal es sencilla. Agruparemos el número binario en grupos de tres bits empezando por la derecha, rellenado con ceros si es preciso. Después transformaremos a decimal cada grupo de 3 bits:

1101011101_2 -> 001 101 011 101 -> 1535_8

El sistema octal se utiliza normalmente para expresar de manera compacta cantidades binarias.

Sistema Hexadecimal

El sistema hexadecimal tiene base 16, es decir utiliza los dígitos del 0 al 9 y las letras A B C D E F. La letra A equivale a un 10 la B al 11 y así sucesivamente.

Por ejemplo un número en representación hexadecimal sería $45AB_{16}$ (con el subíndice indicamos que es hexadecimal para distinguirlo de uno decimal).

Para pasar de base 16 a base 10 se desarrolla el polinomio de potencias de la base:

$$A46'12_{16}=10*16^2+4*16^1+6*16^0+1*16^{-1}+2*16^{-2}$$

El interés de este sistema radica en que la base del hexadecimal (16) es potencia de la base del binario 2^4 y por lo tanto la conversión binario hexadecimal es sencilla. Agruparemos el número binario en grupos de cuatro bits empezando por la derecha, rellenado con ceros si es preciso. Después transformaremos a decimal cada grupo de 4 bits:

1101011101_2 -> 0011 0101 1101 -> $35D_{16}$

El sistema hexadecimal se utiliza normalmente para expresar de manera compacta cantidades binarias.