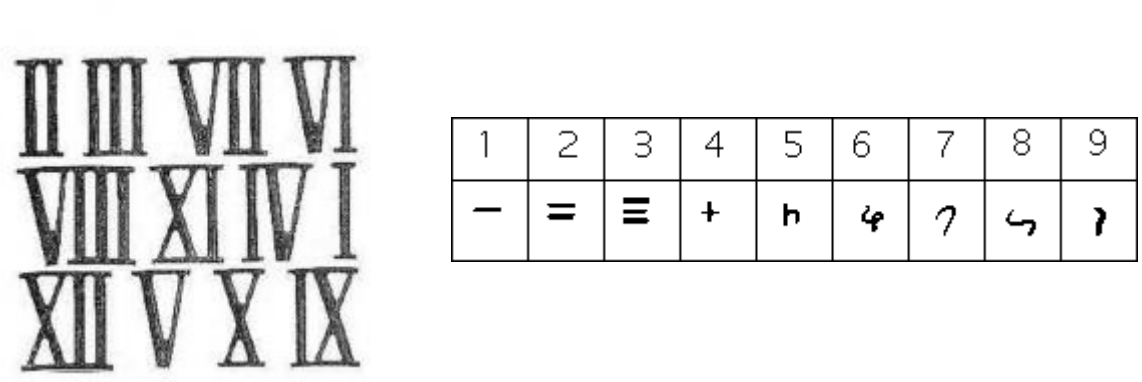
Sistemas de numeración: decimal, binario, hexadecimal e octal



Sistemas de numeración romano e hindú (100 AC)

Os ordenadores, como é sabido, almacenan a información de xeito binario, é dicir, como ceros e uns (0's e 1's). Para poder entender como se garda a información, e como os humanos podemos interactuar con ese xeito de gardar información temos que achegarnos ao que son os sistemas de numeración.

Un **sistema de numeración** é un conxunto de **símbolos** e unhas **regras** para representar cantidades numéricas.

Nos sistemas modernos de numeración cada cantidade vén representada por un conxunto único de símbolos, e viceversa, un conxunto de símbolos representa unha única cantidade.

Os sistemas de numeración máis antigos non (exipcios, gregos, romanos) non empregaba a notación posicional, empregando diferentes métodos para representar unha cantidade numérica. Os sistemas modernos son os chamados **posicionais**, cos que se facilita a realización de operacións matemáticas.

As súas principais características son:

- *Emprégase un **número finito de símbolos, díxitos ou cifras**, o que determina a base do sistema.*
- *Cada cantidade ven representada por unha **secuencia finita** de símbolos do sistema.*
- *A cantidade total obtense **sumando** o valor de cada un dos símbolos.*
- *O valor de cada símbolo depende **do propio símbolo e da posición** que ocupa dentro da secuencia de símbolos. Normalmente terá máis valor canto máis á dereita se sitúe.*

Cantos máis símbolos empregamos, menos cifras precisaremos para representar un número, pois cada cifra pode representar un valor maior.

Veremos estas características co sistema empregado pola maioría das civilizacións actuais, o sistema decimal, para ver a continuación o sistema **binario**, o **hexadecimal** e o **octal**.

Sistema decimal

O sistema decimal é o máis empregado na actualidade, tomado dos hindús polos árabes no século VIII. É o chamado sistema decimal hindú-arábigo.

O hindús tiñan dez símbolos: 0 ,1, 2, 3, 4, 5, 6, 7, 8 e 9, co que se di que a **base** do sistema é **b = 10** (o

número de símbolos empregados). Ademais, o sistema segue o principio da posición, co que un 4 na posición 3 será 4 x 10^3. En xeral, cada número de **n** cifras será un polinomio do tipo:

$$número_{10} = a_{n-1}\ 10^{n-1} + a_{n-2}\ 10^{n-2} + ... + a_2\ 10^2 + a_1\ 10^1 + a_0$$

onde a_i representa o símbolo (unha cifra) que está na posición i. A notación **$número_{10}$** indica que estamos representando o número en base 10.
Por exemplo, como xa sabemos:

$234_{10} = 2 \cdot 100 + 3 \cdot 10 + 4 = 200 + 30 + 4$
$12345_{10} = 1 \cdot 10^4 + 2 \cdot 10^3 + 3 \cdot 10^2 + 4 \cdot 10 + 5 = 10000 + 2000 + 300 + 40 + 5$

Sistema binario

O sistema binario foi introducido por Leibniz no século XVII. É o que empregan as máquinas electrónicas dixitais, xa que representan 2 estados diferentes (pasa corrente ou non, 1 ou 0). Ao contar con 2 símbolos unicamente, simplifícanse moito todas as operacións aritméticas.
Mais, o emprego de só 2 símbolos (o 0 e o 1) obriga a que unha cantidade calquera precise de moitas máis cifras que nun sistema de numeración con base maior.
No sistema binario o alfabeto está formado polos **símbolos {0,1}** e a base **b=2**.
Para representar un número en binario haberá que calcular o polinomio do mesmo xeito que no sistema decimal, como se aprecia nos seguintes exemplos de conversión de base 10 a binario:

$5_{10} = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 = (101)_2$
$121_{10} = 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 = 64+32+16+8+0+0+1 = (1111001)_2$

Estamos descompoñendo un número en base 10 na suma de potencias de 2: $2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, $2^4 = 16$, $2^5 = 32$, $2^6 = 64$, $2^7 = 128$, $2^8 = 256$, etc.
Mostramos na seguinte táboa os primeiros 16 números en binario (temos que calcular as sumas de 8,4,2 e 1).

Decimal	Polinomio	Binario
0	0	0
1	1	1
2	$1 \cdot 2^1 + 0 = 2 + 0$	10
3	$1 \cdot 2^1 + 1 = 2 + 1$	11
4	$1 \cdot 2^2 + 0 \cdot 2^1 + 0 = 4 + 0 + 0$	100
5	$1 \cdot 2^2 + 0 \cdot 2^1 + 1 = 4 + 0 + 1$	101
6	$1 \cdot 2^2 + 1 \cdot 2^1 + 0 = 4 + 2 + 0$	110

7	$1\cdot 2^2 + 1\cdot 2^1 + 1 = 4 + 2 + 1$	111
8	$1\cdot 2^3 + 0\cdot 2^2 + 0\cdot 2^1 + 0 = 8 + 0 + 0 + 0$	1000
9	$1\cdot 2^3 + 0\cdot 2^2 + 0\cdot 2^1 + 1 = 8 + 0 + 0 + 1$	1001
10	$1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 0 = 8 + 0 + 2 + 0$	1010
11	$1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1 = 8 + 0 + 2 + 1$	1011
12	$1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 0 = 8 + 4 + 0 + 0$	1100
13	$1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1 = 8 + 4 + 0 + 1$	1101
14	$1\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 0 = 8 + 4 + 2 + 0$	1110
15	$1\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 1 = 8 + 4 + 2 + 1$	1111

Fixémonos que imos construíndo a táboa dos números binarios do mesmo xeito que se fai cos números do sistema decimal. Comezamos co 0, seguimos co 1, como xa non hai máis cifras temos que comezar a empregar números de 2 cifras: 10 e 11 (o 2 e o 3). Logo pasamos aos de 3 cifras: 100, 101, 110 e 111 (4, 5, 6 e 7). Logo pasamos aos de 4 cifras, e así seguiríamos sucesivamente ata chegar a calquera valor que quixeramos representar.

Cada unha das cifras dun número binario (1 ou 0) coñécese como **díxito binario**.

Con **n** díxitos binarios podemos representar todos os números binarios no rango $[0,2^n-1]$. Así, con 2 díxitos podemos representar $[0, 2^2-1]$, e dicir, do 0 ao 3. Con 4 díxitos, como vemos na táboa, $[0,2^4-1]$, do 0 ao 15. Con 8 díxitos, do 0 ao 255, con 16 díxitos, do 0 ao 65535, etc.

Conversión entre decimal e binario

Para pasar un número binario a decimal non hai máis que realizar o desenrolo polinomial:

$$(a_8\,a_7\,a_6\,a_5\,a_4\,a_3\,a_2\,a_1\,a_0)_2 = a_8\cdot 2^8 + a_7\cdot 2^7 + a_6\cdot 2^6 + a_5\cdot 2^5 + a_4\cdot 2^4 + a_3\cdot 2^3 + a_2\cdot 2^2 + a_1\cdot 2^1 + a_0$$

como fixemos no apartado anterior.

Por exemplo:

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

Se queremos facer o proceso inverso, converter un número decimal a binario, trátase a parte enteira e a parte decimal por separado. A idea sería buscar primeiro o maior múltiplo de 2 menor que o número (chamémoslle n), asignarlle un 1, dividir o número entre ese múltiplo, coller o resto, dividilo entre n/2 para ter a seguinte cifra, coller o resto, dividilo entre n/4, coller o resto, etc. ata chegar a dividir o resto entre 2. O valor en binario estaría formado polos cocientes ordenados.

Por exemplo, se partimos de 121 fariamos o seguinte. A maior potencia de 2 menor que 121 sería 64, así que (os resultado enteiro da división, os cocientes, están na liña inferior) imos calculando todos os cocientes e os restos (o operador % é o operador resto enteiro):

121 % 64 = 57 % 32 = 25 % 16 = 9 % 8 = 1 % 4 = 1 % 2 = 1 % 1 = 0
1 1 1 1 0 0 1

Formado o resultado 121_{10} = $(1111001)_2$

O procedemento anterior parece o lóxico se sabemos a maior potencia de 2 menor que o número. Pero moitas veces non sabemos cal é ese valor. Nestes casos podemos realizar a operación anterior doutro xeito: iremos calculando os restos de ir dividindo o número inicial entre 2 sucesivamente, ata chegar a 0. Cada un dos restos formará o número en binario en orde inversa, mentres que os cocientes serán os seguintes dividendos. Vexamos co mesmo exemplo como facer (neste caso os restos están na liña inferior):

121 / 2 = 60 / 2 = 30 / 2 = 15 / 2 = 7 / 2 = 3 / 2 = 1 / 2 = 0
1 0 0 1 1 1 1

Logo ordenamos os restos en orde inversa, co que 121_{10} = $(1111001)_2$, o mesmo resultado que co método anterior.

Para converter a parte fraccionaria dun número a binario, realízanse multiplicacións sucesivas por 2, e imos eliminando a parte enteira (que será 0 ou 1), ata que se obteña co resultado da multiplicación o 1 ou se consigan suficientes díxitos binarios.

Nalgún caso, un número cun número finito de díxitos decimais pode ter infinitos díxitos cando se pase a binario.

Por exemplo, converter $(0,6875)_{10}$ a base 2:

0,6875 · 2 = 1,375. Parte enteira: 1
0,375 · 2 = 0,75. Parte enteira: 0
0,75 · 2 = 1,5. Parte enteira: 1
0, 5 · 2 = 1 Parte enteira: 1

Xuntamos agora as partes enteiras: 1011

Polo tanto $(0,6875)_{10}=(0,1011)_2$.

Para facer a conversión inversa, sumamos todos os compoñentes do polinomio de base 2. É dicir, facemos:

$0,1011_2 = 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} = 1 \cdot 0,5 + 0 + 1 \cdot 0,125 + 1 \cdot 0,0625 = 0,5+0,125+0,0625= 0,6875_{10}$

Sistema hexadecimal

Un dos problemas do sistema binario (o que emprega o ordenador) é que resulta complicado para as persoas. Por exemplo, supoñamos que queremos realizar unha operación suma entre 2 números. O ordenador codificará os dous números en binario, e terá tamén unha simboloxía en binario para o operador suma.

Toda a codificación en binario (como se empregaba no principio na programación en código máquina) resulta moi complexa para realizar por un humano. Para evitar ter que traballar en código binario, pasouse a utilizar un sistema de numeración que tivese unhas características intermedias:

- A base do sistema sexa próxima ao decimal para que resulte fácil o seu traballo para as persoas.
- A conversión ao sistema binario sexa simple e rápida.

Un dos sistemas que compre estas dúas características é o que se coñece como **sistema hexadecimal**. Este sistema usa como base b=16, co que como sabemos ten que empregar 16 símbolos. Estes son as 10 cifras e as 6 primeiras letras maiúsculas: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.

Así, estes números representaran do 0 ao 15, e cando queremos representar o 16 temos que pasar a 2 cifras: 10_{16},11_{16},12_{16},...,$1A_{16}$,$1B_{16}$,$1C_{16}$,$1D_{16}$,$1E_{16}$,$1F_{16}$ (que serían o 16, 17, 18,..., 30 e 31). Logo seguiríamos co 20_{16} (32_{10}), 21_{16} (33_{10}), etc.

Mostramos na táboa seguinte os primeiros 32 números do **sistema hexadecimal** coa súa correspondencia co sistema decimal e binario.

Decimal	Hexadecimal	Binario	Decimal	Hexadecimal	Binario
0	0	0000	16	10	00010000
1	1	0001	17	11	00010001
2	2	0010	18	12	00010010
3	3	0011	19	13	00010011
4	4	0100	20	14	00010100
5	5	0101	21	15	00010101
6	6	0110	22	16	00010110
7	7	0111	23	17	00010111
8	8	1000	24	18	00011000
9	9	1001	25	19	00011001
10	A	1010	26	1A	00011010
11	B	1011	27	1B	00011011
12	C	1100	28	1C	00011100
13	D	1101	29	1D	00011101
14	E	1110	30	1E	00011110
15	F	1111	31	1F	00011111

Hai que darse conta que a base do sistema hexadecimal é $16=2^4$, isto é, potencia de 2. De modo que a conversión entre o sistema hexadecimal e o sistema binario é inmediata, pois cada cifra en hexadecimal está representada en binario **sempre polos mesmos 4 díxitos**, independentemente da posición que ocupen. Isto pódese apreciar na táboa anterior. Por exemplo, o F en hexadecimal sempre se traduce a

binario como 1111_2.

Supoñamos que queremos representar o $11111111_2=255_{10}$, para pasalo a hexadecimal só temos que agrupar 4 cifras do binario: $FF_{16}= 11111111_2=255_{10}$. Do mesmo xeito, o número binario 11110000_2 que é 240_{10} representarémolo en hexadecimal como $F0_{16}$, pois as primeiras 4 cifras en binario (1111_2) son F_{16} e as últimas catro cifras 0000_2 correspóndense co 0_{16}.

Vexamos algún exemplo máis:

- converter o número $EA12_{16}$ a binario. Iremos traducindo cada símbolo hexadecimal a binario directamente. O E_{16} é o 1110_2, o A_{16} 1010_2, o 1_{16} é 0001_2, e o 2_{16} é o 0010_2. Así pois, todo xunto:

  $EA12_{16}=1110101000010010_2$
- converter o número 1010000100011110_2 a hexadecimal. Do mesmo xeito que no exemplo anterior, iremos descompoñendo o número en grupos de 4 díxitos, co que 1010_2 é A_{16}, 0001 é o 1_{16}, e o 1110_2 é o E_{16}. Así, pois o 1010000100011110_2 é o $A11E_{16}$.

Para representar os números fraccionarios no sistema hexadecimal só hai que engadir ceros (á esquerda na parte enteira, á dereita na parte fraccionaria) ata completar grupos de 4 cifras. Por exemplo:

- converter $111100,111011_2$ ao sistema hexadecimal. Completamos primeiro con ceros: $00111100,11101100_2$ e agora asignamos o símbolo correspondente a cada grupo de 4 cifras: $3C,EC_{16}$.

Sistema Octal

O sistema octal é un sistema de numeración, que ten como base 8. Emprega por tanto 8 símbolos, que son os 8 primeiros números: 0,1,2,3,4,5,6,7. É menos empregado que o hexadecimal, se ben tamén é moi usado no eido da informática.

A táboa en potencias de 8 sería:

8^3	512
8^2	64
8^1	8
8^0	1
8^{-1}	0,125
8^{-2}	0,015625

A conversión entre decimal e octal faise do mesmo xeito que o xa estudado para binario e hexadecimal:

Para converter un número octal a decimal descompoñemos nas potencias de 8:

$764_8 = 7 * 8^2 + 6 * 8^1 + 4 * 8^0 = 448 + 48 + 4 = 500_{10}$

Para facer o proceso inverso, converter un número decimal a octal faremos de xeito similar ao xa estudado. Iremos dividindo entre 8 e gardando os restos, ata chegar ao número menor que 8. Logo construiremos o número octal tomando o resultado final e os restos en orde inverso:

500 / 8 = 62 / 8 = 7

4 6

Co que $500_{10} = 764_8$.

Como $8 = 2^3$, cada número en octal pódese transformar en binario de xeito directo, substituíndo cada

número octal por 3 cifras binarias (do mesmo que xeito que traducíamos cada número hexadecimal por 4 cifras binarias). Vexamos as equivalencias para os primeiros números (fíxate no agrupamento de 3 en 3 bits):

Decimal	Octal	Binario	Decimal	Octal	Binario
0	0	000	8	10	001 000
1	1	001	9	11	001 001
2	2	010	10	12	001 010
3	3	011	11	13	001 011
4	4	100	12	14	001 100
5	5	101	13	15	001 101
6	6	110	14	16	001 110
7	7	111	15	17	001 111

Do mesmo xeito que fixemos cando falamos de numeración hexadecimal, para transformar calquera número de octal a binario, non temos máis que ir traducindo cada díxito octal por 3 díxitos binarios (mostramos o número binario con grupos de 3 bits para mellor comprensión):

354223_8 = 011 101 100 010 010 011_2
657113_8 = 110 101 111 001 001 011_2

Suma de números binarios

Do mesmo xeito que podemos sumar números no sistema decimal podemos facer a suma en binario, tendo en conta que 1 + 1 = 0, e levamos 1 (o que se coñece como acarreo).

0 + 0	0
0 + 1	1
1 + 0	1
1 + 1	10 (0 con acarreo 1)

O resultado da suma debe ser igual ao resultado se estivésemos traballando en decimal. Exemplos:

a) Sumar os números binarios 100100_2 e 10010_2

1 0 0 1 0 0 .....36_{10}
+ 1 0 0 1 0 .....+ 18_{10}
---
1 1 0 1 1 0.....54_{10}

b) Sumar os números 110011_2 e 10011_2 (fíxate que cando se suma 1+1 da 0, e hai un acarreo (un 1) para a seguinte cifra a sumar.)

1 1 0 0 1..... 25_{10}
+ 1 0 0 1 1.....+ 19_{10}
---
1 0 1 1 0 0.....44_{10}

Operadores lóxicos

Os operadores lóxicos son utilizados na informática para realizar operacións. Veremos os operadores AND (empregado en redes en máscaras IP) e o operador OR:

Operador AND

Produce un resultado con valor de 1 cando ambos operandos son 1, e 0 nos outros casos.

Valor	Valor	Resultado
0	0	0
0	1	0
1	0	0
1	1	1

En programación enténdese 1 como *true* – verdadeiro, e 0 como *false* – falso, co que para dar un resultado verdadeiro (1) teñen que ser os dous valores 1. Só será verdadeiro se o primeiro valor é verdadeiro AND o segundo valor é verdadeiro. Se non o resultado será falso.

Exemplos:

```
    110011          0011011
AND 011001      AND 1010111
    ------          -------
    010001          0010011
```

O operador AND en binario correspóndese coa **multiplicación binaria**: a táboa de multiplicación é a mesma que a do operador AND. Exemplo, 1 x 1 = 1 AND 1 = 1

Operador OR

Será verdadeiro (1) se algún dos dous valores é verdadeiro. O OR só será falso (0) cando ámbolos dous valores sexan falsos (0).

Valor	Valor	Resultado
0	0	0
0	1	1
1	0	1
1	1	1

Exemplos:

```
    110011          0011011
OR  011001      OR  1010111
    ------          -------
    111011          1011111
```