UT2.1: Representación de la información





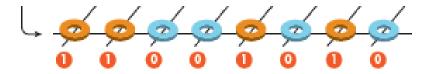


Tipos de datos



Tal y como sabemos componentes internos de un ordenador no almacenan directamente letras ni números o imágenes, si no 1 y 0's, es decir la presencia de corriente eléctrica (1) o su ausencia (0).

Cuando un usuario quiere almacenar una letra en memoria, por ejemplo desde un programa procesador de textos (software), usando el sistema operativo se convertirá en un conjunto de 8 bits que almacenarán esa letra codificada como un conjunto de impulsos eléctricos (1 y 0) en la memoria del ordenador.



En este tema analizaremos los distintos **sistemas de numeración** que se usan en un ordenador (internamente binario) y por parte del sistema operativo (octal y hexadecimal) así como los **sistemas de codificación** o equivalencia utilizados para descifrar esos valores por parte del software.



Se define un **sistema de numeración** como el conjunto de símbolos y reglas que se utilizan para representar cantidades o datos numéricos.

Estos sistemas se caracterizan por la base a la que hacen referencia.

La **base** de un sistema de numeración se refiere al número de símbolos que componen dicho sistema.

Los humanos utilizamos en el día a día un sistema de numeración en base 10 (como nuestros dedos), compuesto por 10 símbolos diferentes (del 0 al 9).

El Teorema Fundamental de la Numeración (TFN) queda determinado por la fórmula siguiente:

NÚM = Σ X_i · Bⁱ

$$283 = 2 \cdot 102 + 8 \cdot 10^{1} + 3 \cdot 100 = 200 + 80 + 3$$

$$\begin{split} N\acute{U}M &= X_{_{\! 1}} \cdot 10^{_{\! 1}} + \ldots + X_{_{\! 2}} \cdot 10^{_{\! 2}} + X_{_{\! 1}} \cdot 10^{_{\! 1}} + X_{_{\! 0}} \cdot 10^{_{\! 0}} + \\ &+ X_{_{\! -1}} \cdot 10^{_{\! -1}} + X_{_{\! -2}} \cdot 10^{_{\! -2}} \cdot \ldots + X_{_{\! -N}} \cdot 10^{_{\! -N}} \end{split}$$



El **Teorema Fundamental de la Numeración (TFN)** queda determinado por la fórmula siguiente:

$$N\dot{U}M = \Sigma X_i \cdot B^i$$

Así para el **sistema decimal en base 10**, aplicando el *Teorema Fundamental de la Numeración* visto anteriormente, las cifras que componen un número son las cantidades que están multiplicando a las distintas potencias de diez (10, 100, 1000, 10000, etc.)

Por ejemplo, $745 = 7 \cdot 100 + 4 \cdot 10 + 5 \cdot 1$ O lo que es lo mismo: $745 = 7 \cdot 10^2 + 4 \cdot 10^1 + 5 \cdot 10^0$



Principales sistemas de codificación numérica usados en informática:

- **Binario** (abreviado como *bin*) Sistema en **base 2** que utiliza dos símbolos diferentes: el cero y el uno (0,1).
- Octal. Es un sistema en base 8 que utiliza los símbolos del 0 al 7 para representar las cantidades, las cuales quedan reproducidas posicionalmente por potencias de 8.
- **Hexadecimal** (abreviado como *hex*). Es un sistema de numeración en **base 16**. Usa 16 símbolos diferentes, del 0 al 9 y los dígitos valores (o letras) A, B, C, D, E y F.

El sistema que maneja internamente un ordenador es el binario, pero, en ocasiones, por comodidad en el manejo de los datos, se suele utilizar el octal y el hexadecimal, ya que mucha de la información que nos muestra el sistema operativo, como direcciones de memoria, se expresa en hexadecimal.



Binario



Decimal	Binario
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000



Hexadecimal



Decimal	Hexadecimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	В
12	С
13	D
14	E
15	F
16	10
17	11



Hexadecimal



0
1
2
3
4
5
6
7
10
11
12
13
14
15
16
17
20





Decimal	Binario	Base 8	Base 16
0	00000	0	0
1	00001	1	1
2	00010	2	2
3	00011	3	3
4	00100	4	4
5	00101	5	5
6	00110	6	6
7	00111	7	7
8	01000	10	8
9	01001	11	9
10	01010	12	Α
11	01011	13	В
12	01100	14	С
13	01101	15	D
14	01110	16	E
15	01111	17	F
16	10000	20	10
17	10001	21	11



Un número binario está por tanto compuesto por bits:

A mayor número de bits (dígitos binarios), mayor número de combinaciones posibles:

Nº de bits	Combinaciones posibles
1	0,1
2	00,01,10,11
3	000,001,010,011,100,101,110,111



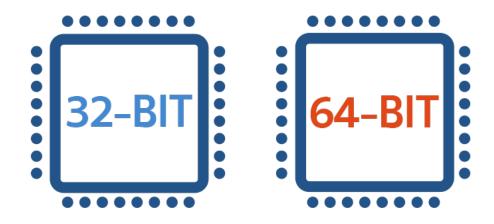
Es necesario saber interpretar el código binario para poder entender las operaciones que en muchas ocasiones se realizan dentro del ordenador. Para ello, debemos aprender a pasar números binarios a decimales y a la inversa, es decir la conversión entre sistemas de numeración.

Por extensión, el ordenador utiliza los sistemas de numeración de base 8 y base 16 (por ser múltiplos del sistema binario) para mostrarnos información relativa a algunos procesos que realiza. A la hora de realizar la comunicación entre dispositivos se utiliza así mismo el código binario (base 2)

Debido a la estructura de 64 bits de un microprocesador las direcciones de memoria se expresan a menudo en hexadecimal. Por ejemplo, para no tener que escribir 111111010100000000000010101100 podemos escribir 3F5000AC en hexadecimal.



- Un procesador y un bus de **32 bits** permite especificar a la CPU 2³²= 4.294.967.296 direcciones de memoria distintas, lo cual a su vez genera un límite de 4GB en el dispositivo.
- Un procesador y un bus de **64 bits** permite especificar a la CPU 2⁶⁴ direcciones (lo que se traduce en un rango de valores desde 0 hasta 18.446.744.073.709.551.615 de direcciones o 18 exabytes)

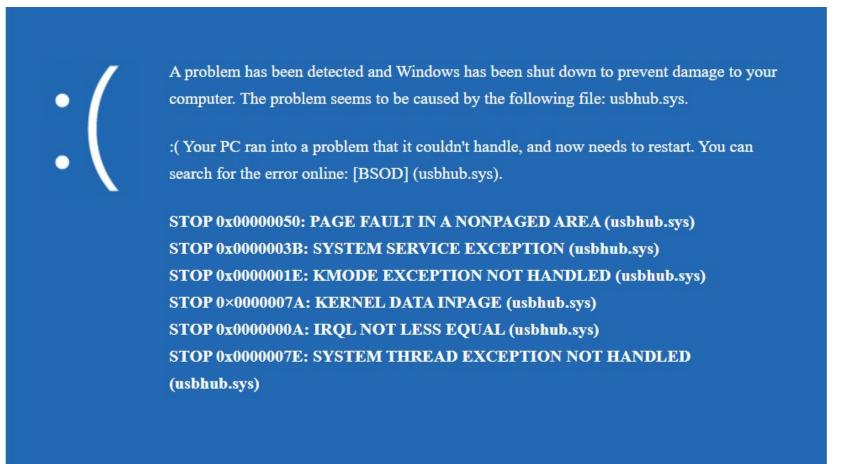




```
A problem has been detected and Windows has been shut down to prevent damage
to your computer.
The problem seems to be caused by the following file: SPCMDCON.SYS
PAGE_FAULT_IN_NONPAGED_AREA
If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:
Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.
If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.
Technical information:
*** STOP: 0x00000050 (0xFD3094C2,0x00000001,0xFBFE7617,0x00000000)
*** SPCMDCON.SYS - Address FBFE7617 base at FBFE5000, DateStamp 3d6dd67c
```

Pantallazo de la muerte (blue screen of death) en un sistema Windows clásico



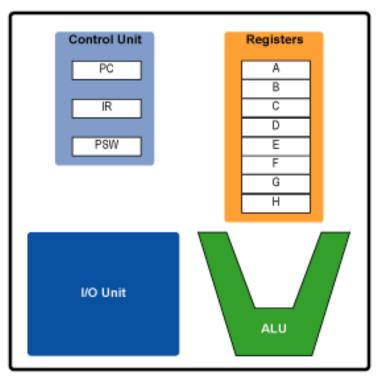


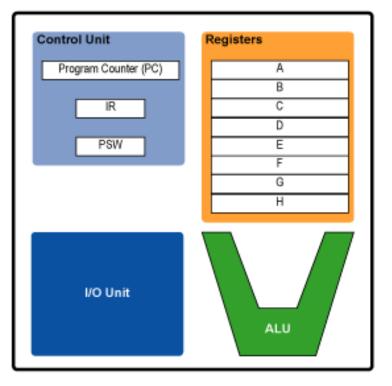
Pantallazo de la muerte (blue screen of death) en otro sistema Windows actual



```
0.5529621 CPU: 3 PID: 1 Comm: swapper/0 Not tainted 4.8.0-44-generic #47~16
.04.1-Ubuntu
     0.5530121 Hardware name: TOSHIBA Satellite C640/Portable PC, BIOS 2.10 11/0
9/2011
     0.5530601
                000000000000000 0000000064eb1541 ffff9575f4473df0 fffffffb8a2e
073
     0.5532001
                ffff9575f3ae5000 ffffffffb92725a0 ffff9575f4473e78 fffffffb879e
6ad
     0.553341} ffff957500000010 ffff9575f4473e88 ffff9575f4473e20 0000000064eb1
541
     0.5534821 Call Trace:
     0.553519] [<ffffffffb8a2e073>1 dump_stack+0x63/0x90
     0.5535621 [<fffffffffb879e6ad>] panic+0xe4/0x226
     0.5536061 [<fffffffb9586540>1 mount_block_root+0x1fb/0x2c2
     0.5536471 [<ffffffffb958663a>1 mount_root+0x33/0x35
     0.5536881 [<ffffffffb9586776>] prepare_namespace+0x13a/0x18f
     0.5537311 [<fffffffb95861eb>] kernel_init_freeable+0x1ee/0x217
    0.553775] [<ffffffffb8e8dZae>] kernel_init+0xe/0x100
    0.553817] [<ffffffffb8e9aa1f>] ret_from_fork+0x1f/0x40
    0.553858] [<fffffffffb8e8d2a0>] ? rest_init+0x80/0x80
     0.5539321 Kernel Offset: 0x37600000 from 0xffffffff81000000 (relocation ran
qe: 0xffffffff80000000-0xfffffffffffffffff)
     0.5539911 --- [ end Kernel panic - not syncing: UFS: Unable to mount root fs
on unknown-block(0.0)
```







32-bit 64-bit

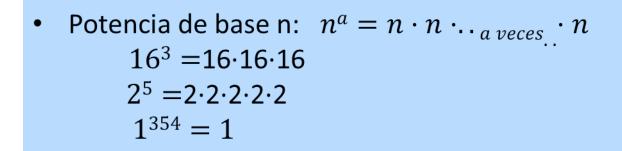


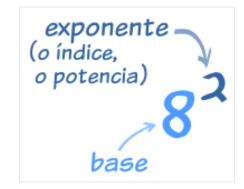
Repaso de operaciones matemáticas con potencias:

• Potencia de exponente 0:
$$a^0 = 1$$

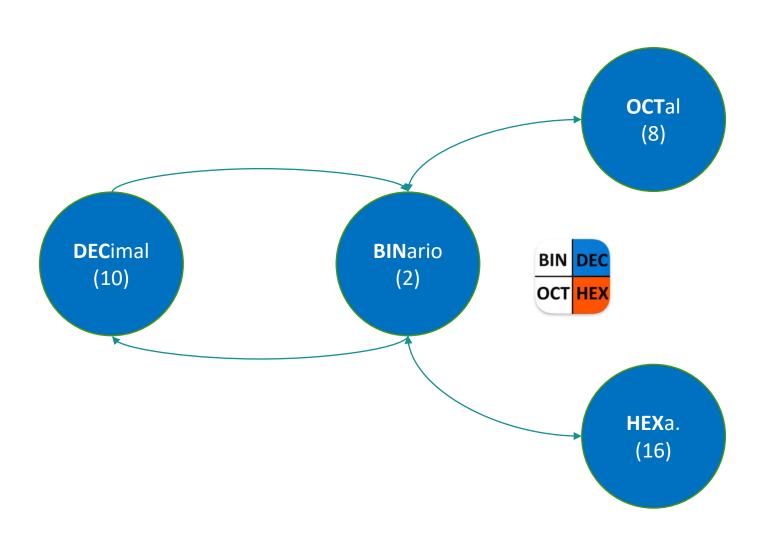
 $16^0 = 1$ $2^0 = 1$

• Potencia de exponente 1: $a^1 = a$ $2^1 = 2$ $6^1 = 6$











Conversión decimal a binario

Para hacer la conversión de decimal a binario, hay que ir dividiendo el número decimal entre 2 y anotar en una columna a la derecha el resto (un 0 si el resultado de la división es par y un 1 si es impar). La lista de ceros y unos leídos de abajo a arriba es el resultado.

✓ Convertir en binario el número decimal 28:



Conversión decimal a binario

Para hacer la conversión de decimal a binario, hay que ir dividiendo el número decimal **entre 2** y anotar en una columna a la derecha el resto (un 0 si el resultado de la división es par y un 1 si es impar). La lista de ceros y unos leídos de abajo a arriba es el resultado.

✓ Convertir en binario el número decimal 52:



Conversión decimal a binario

Para hacer la conversión de decimal a binario, hay que ir dividiendo el número decimal entre 2 y anotar en una columna a la derecha el resto (un 0 si el resultado de la división es par y un 1 si es impar). La lista de ceros y unos leídos de abajo a arriba es el resultado.

✓ Convertir en binario el número decimal 52:



Conversión decimal a octal

Para hacer la conversión de decimal a octal, hay que ir dividiendo el número decimal **entre 8** hasta que el dividendo sea menor que el divisor y anotar en una columna a la derecha el resto. La lista de restos leídos de abajo a arriba es el resultado.

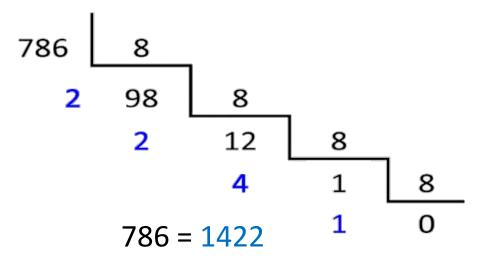
✓ Convertir en octal el número decimal 786:



Conversión decimal a octal

Para hacer la conversión de decimal a octal, hay que ir dividiendo el número decimal **entre 8** hasta que el dividendo sea menor que el divisor y anotar en una columna a la derecha el resto. La lista de restos leídos de abajo a arriba es el resultado.

✓ Convertir en octal el número decimal **786**:





Conversión decimal a hexadecimal

Para hacer la conversión de decimal a hexadecimal, hay que ir dividiendo el número decimal entre 16 hasta que el dividendo sea menor que el divisor y anotar en una columna a la derecha el resto con la cifra *hexadecimal* que se corresponda según su tabla del 1 a la F. La lista de restos leídos abajo a arriba será el resultado.

✓ Convertir en hexadecimal el número decimal 1869:



Conversión decimal a hexadecimal

Para hacer la conversión de decimal a hexadecimal, hay que ir dividiendo el número decimal entre 16 hasta que el dividendo sea menor que el divisor y anotar en una columna a la derecha el resto con la cifra *hexadecimal* que se corresponda según su tabla del 1 a la F. La lista de restos leídos abajo a arriba será el resultado.

✓ Convertir en hexadecimal el número decimal 1869:

$$1869 = 74D$$



Conversión binario a decimal

Para pasar de binario a decimal se toman los dígitos binarios y se van multiplicando por **potencias de 2** de izquierda a derecha comenzando por un exponente n-1 hasta 0, siendo n el número de dígitos del número original

✓ Convertir en decimal el número binario 1000011011:

$$1 \cdot 2^9 + 0 \cdot 2^8 + 0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 =$$

$$512 + 0 + 0 + 0 + 0 + 16 + 8 + 0 + 2 + 1 \rightarrow 539$$



Conversión binario a decimal

Para pasar de binario a decimal se toman los dígitos binarios y se van multiplicando por **potencias de 2** de izquierda a derecha comenzando por un exponente n-1 hasta 0, siendo n el número de dígitos del número original

✓ Convertir en decimal el número binario 1001010:

✓ Convertir en decimal el número binario 1111100:



Conversión binario a decimal

Para pasar de binario a decimal se toman los dígitos binarios y se van multiplicando por **potencias de 2** de izquierda a derecha comenzando por un exponente n-1 hasta 0, siendo n el número de dígitos del número original

✓ Convertir en decimal el número binario 1001010:

$$1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 =$$

64 + 0 + 0 + 8 + 0 + 2 + 0 \rightarrow **74**

✓ Convertir en decimal el número binario 1111100:

$$1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 =$$

64 + 32 + 16 + 8 + 0 + 2 + 0 \rightarrow 124



Conversión binario a octal

Para realizar la conversión de binario a octal se puede agrupar la cantidad binaria en grupos de 3 en 3 empezando por el lado derecho. Si al terminar de agrupar no completa 3 dígitos, entonces habrá que agregar ceros a la izquierda.

Número en binario	000	001	010	011	100	101	110	111
Número en octal	0	1	2	3	4	5	6	7

- ✓ Convertir en octal el número binario 110111
 111 = 7
 110 = 6 → Luego el número será 67
- ✓ Convertir en octal el número binario 11001111



Conversión binario a octal

Para realizar la conversión de binario a octal se puede agrupar la cantidad binaria en grupos de 3 en 3 empezando por el lado derecho. Si al terminar de agrupar no completa 3 dígitos, entonces habrá que agregar ceros a la izquierda.

Número en binario	000	001	010	011	100	101	110	111
Número en octal	0	1	2	3	4	5	6	7

✓ Convertir en octal el número binario 110111

✓ Convertir en octal el número binario 11001111



Conversión binario a hexadecimal

Para realizar la conversión de binario a hexadecimal hay que agrupar la cantidad binaria **en grupos de 4 en 4** empezando por el lado derecho. Si al terminar de agrupar no completa 4 dígitos, habrá que agregar **ceros** a la izquierda.

Binario	Decimal	HEXA	Binario	Decimal	HEXA
0000	0	0	1000	8	8
0001	1	1	1001	9	9
0010	2	2	1010	10	Α
0011	3	3	1011	11	В
0100	4	4	1100	12	С
0101	5	5	1101	13	D
0110	6	6	1110	14	E
0111	7	7	1111	15	F



Conversión binario a hexadecimal

Para realizar la conversión de binario a hexadecimal hay que agrupar la cantidad binaria **en grupos de 4 en 4** empezando por el lado derecho. Si al terminar de agrupar no completa 4 dígitos, habrá que agregar **ceros** a la izquierda.

- ✓ Convertir en hexadecimal el número binario 110111010 1010 = 10 = A 1011 = 11 = B 0001 = 1 → Luego el número será 1BA
- ✓ Convertir en hexadecimal el número binario 11011100101



Conversión binario a hexadecimal

Para realizar la conversión de binario a hexadecimal hay que agrupar la cantidad binaria **en grupos de 4 en 4** empezando por el lado derecho. Si al terminar de agrupar no completa 4 dígitos, habrá que agregar **ceros** a la izquierda.

✓ Convertir en hexadecimal el número binario 110111010

```
1010 = 10 = A
1011 = 11 = B
0001 = 1 → Luego el número será 1BA
```

✓ Convertir en hexadecimal el número binario 11011100101

```
0101 = 5
1111 = 14 = E
0110 = 6 → Luego el número será 6E5
```



Conversión octal a decimal

Para pasar de octal a decimal se toman los dígitos binarios y se van multiplicando por **potencias de 8** de izquierda a derecha y comenzando por un exponente n-1 hasta 0, siendo n el número de dígitos del número original.

✓ Convertir en decimal el número octal 421:

✓ Convertir en decimal el número octal 7014:



Conversión octal a decimal

Para pasar de octal a decimal se toman los dígitos binarios y se van multiplicando por **potencias de 8** de izquierda a derecha y comenzando por un exponente n-1 hasta 0, siendo n el número de dígitos del número original.

✓ Convertir en decimal el número octal 421:

$$4.8^2 + 2.8^1 + 1.8^0 = 4.64 + 2.8 + 1 = 256 + 16 + 1 \rightarrow 273$$

✓ Convertir en decimal el número octal 7014:

$$7.8^3 + 0.8^2 + 1.8^1 + 4.8^0 = 7.512 + 0 + 8 + 4 = 256 + 16 + 1 \rightarrow 3596$$



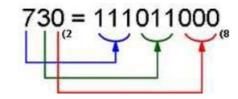
Conversión octal a binario

Para convertir números octales a binarios se sustituye cada dígito octal por su representación binaria **con tres dígitos** de acuerdo a la tabla ya vista.

Número en binario	000	001	010	011	100	101	110	111
Número en octal	0	1	2	3	4	5	6	7

✓ Convertir en binario el número octal 730

7 3 0 = 111 011 000
$$\rightarrow$$
 111 011 000



✓ Convertir en binario el número octal 1274



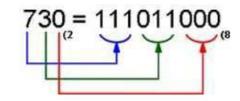
Conversión octal a binario

Para convertir números octales a binarios se sustituye cada dígito octal por su representación binaria **con tres dígitos** de acuerdo a la tabla ya vista.

Número en binario	000	001	010	011	100	101	110	111
Número en octal	0	1	2	3	4	5	6	7

✓ Convertir en binario el número octal 730

7 3 0 = 111 011 000
$$\rightarrow$$
 111 011 000



✓ Convertir en binario el número octal 1274

1 2 7 4 = 001 010 111
$$100 \rightarrow 001 010 111 100$$



Conversión hexadecimal a decimal

Para pasar de hexadecimal a decimal se toman los dígitos hexadecimales y se van multiplicando por **potencias de 16** de izquierda a derecha y comenzando por un exponente n-1 hasta 0, siendo n el número dígitos del número original.

✓ Convertir en decimal el número hexadecimal 9F2:

9.16²+
$$\mathbf{F}$$
.16¹+ $\mathbf{2}$.16⁰ = 9.16²+15.16¹+2.16⁰= 9.256 + 240 + 2 \rightarrow 2546

✓ Convertir en decimal el número hexadecimal B2F8:

B·
$$16^3$$
 + **2**· 16^2 + **F**· 16^1 + **8**· 16^0 = $11 \cdot 16^3$ + $2 \cdot 16^2$ + $15 \cdot 16^1$ + $8 \cdot 1$ = $11 \cdot 4096$ + $2 \cdot 256$ + $15 \cdot 16$ + $8 \cdot 1$ \rightarrow **45816**



Conversión hexadecimal a binario

Para convertir números hexadecimales a binarios se sustituye cada dígito hexadecimal por su representación binaria **con cuatro dígitos** de acuerdo a la tabla vista anteriormente.

Ejemplo: Convertir 9F2₁₆ a binario

$$9F2_{16} = 9$$

$$\downarrow \qquad \qquad \downarrow \qquad \qquad \downarrow$$

$$1001 \qquad 1111 \qquad 0010$$

$$= 1001111110010_{2}$$

✓ Convertir en binario el número hexadecimal BEE:

✓ Convertir en decimal el número hexadecimal 70C558:



Conversión hexadecimal a binario

Para convertir números hexadecimales a binarios se sustituye cada dígito hexadecimal por su representación binaria **con cuatro dígitos** de acuerdo a la tabla vista anteriormente.

Ejemplo: Convertir 9F2₁₆ a binario

$$9F2_{16} = 9$$

$$\downarrow \qquad \qquad \downarrow \qquad \qquad \downarrow$$

$$1001 \qquad 1111 \qquad 0010$$

$$= 100111110010_{2}$$

✓ Convertir en binario el número hexadecimal BEE:

11 14 14 \rightarrow 1011 1110 1110

✓ Convertir en decimal el número hexadecimal 70C558:

7 0 12 5 5 8 \rightarrow 0111 0000 1100 0101 0101 1000



Conversión hexadecimal a octal

Debido a que ambos sistemas se relacionan directamente con el sistema binario, lo mas conveniente en el desarrollo de esta conversión es:

- 1. Convertir el número del sistema hexadecimal al sistema binario, como se indicó anteriormente en de **hexadecimal a binario**.
- 2. Posteriormente hacer la transformación del sistema **binario a octal** como hemos visto también anteriormente.
- ✓ Convertir en octal el número hexadecimal 1D8:
 - **1 15 8** → 0001 1101 1000 *en binario*

000111011000

```
000 = 0

011 = 3 \rightarrow 0730 \text{ en octal}

111 = 7

000 = 0
```



Es evidente que si se dispone únicamente de los dos símbolos 0 y 1 usando un código binario natural sólo es posible representar números enteros y positivos.

Para representar un **número negativo**, en matemáticas se hace uso de un signo adicional "—" que precede al número negativo. Dado que en un sistema digital sólo se podrán disponer de los dos símbolos ya mencionados, se han ideado múltiples soluciones para representar y operar con números negativos.

Para representar números con signo existen varias posibilidades que veremos:

- 1. Signo y magnitud
- 2. Complemento a uno
- 3. Complemento a dos



1. Signo y magnitud

Partiendo de que la forma de operar en un sistema digital es a través de un conjunto definido de bits, por ejemplo grupos de 8 bits (un byte), el enfoque es reservar 1 bit (normalmente el primero) para indicar el signo. Normalmente se asocia un 0 al signo "+" y un 1 al signo "-". El resto de los bits del grupo indica la magnitud.

La cifra **00000011** equivale a +3 en decimal mientras que **10000011** equivale a – 3.

Se debe tener en cuenta los siguientes aspectos importantes:

- Mientras que en un grupo de 8 bits podemos almacenar desde el valor (en decimal) 0 hasta 255, si es un byte con signo podremos almacenar desde –127 hasta +127.
- A la hora de hacer operaciones debemos tratar de forma separada el signo, es decir, debemos procesar por una parte los signos y por otra las magnitudes.
- El cero está representado dos veces: 00000000 y 10000000 lo cual es poco eficiente.



2. Complemento a uno

Otra forma de representación es utilizar el **complemento a uno** para representar los números negativos. Se reserva igualmente el primer bit para representar el signo y el resto de bits se usan para representar:

- Si es positivo, se pone tal cual.
- Si es negativo, se pone el **complemento a uno**: se cambian todos los 1s por 0s y todos los 0s por 1s.

Así el numero 3 se representa igualmente por 00000011 y el número -3 se representa por 11111100.

Las operaciones son más fáciles que con la representación "signo y magnitud" pero adolece igualmente del problema de la <u>doble representación del 0</u>. En efecto, tenemos que 00000000 y 11111111 lo representan.



3. Complemento a dos

Un enfoque que solventa algunos de los problemas de los anteriores es la representación de número negativo en **complemento a dos**. Al igual que en el caso anterior, el primer bit le reservamos para el signo y el resto de bits se usan para representar:

- Si es positivo, se pone tal cual.
- Si es negativo, se pone el complemento a 2

Para obtener el complemento a dos se halla el complemento a uno como hemos visto anteriormente y se le suma 1.

Así, el número 3 se representa igualmente por 00000011 y el número –3 se representará como por 11111101.

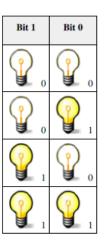
En este caso, el 0 (decimal) sólo tiene una única representación 00000000 y las operaciones aritméticas se pueden realizar mediante <u>sumadores</u>.



Bit es el acrónimo *Binary digit* ('dígito binario'). Un bit es un dígito del sistema de numeración binario. Las unidades de almacenamiento tienen por símbolo bit.

El **bit** es la unidad mínima de información empleada en informática, en cualquier dispositivo digital, o en la teoría de la información. Con él, podemos representar dos valores cuales quiera, como verdadero o falso, abierto o cerrado, blanco o negro, norte o sur.

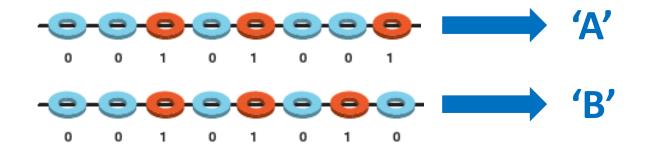
Los bits se pueden combinar para dar distintos resultados y generar distintos sistemas de codificación:





Los **sistemas de codificación** se utilizan para procesar la información que el usuario entiende y el ordenador no.

En dispositivos magnéticos, por ejemplo, cada posición magnetizada se convierte en uno y cada posición no magnetizada en un cero. Se buscará en una **tabla de códigos** y se compara la combinación de esos ocho bits, obteniendo la equivalencia con un carácter concreto en el caso de textos.





Diferentes tipos de información poseen diferentes codificaciones, pero las más habituales son:

Codificaciones numéricas:

Número enteros, números en coma flotante, etc.

Codificaciones alfanuméricas:

ASCII, Unicode (UTF-8, UTF-16)



Audio (wav, mp3, ogg), Gráficos (jpeg, png, tiff), Video (mpeg, avi)

Codificaciones de compresión:

Sin pérdida (GZIP, BZIP2, LHA), con pérdida (mp3, ogg, jpeg, mpeg)

Otros:

Cifrado de clave única, cifrado de clave pública, hash.





Se denomina **alfabeto** al **conjunto de caracteres** para representar una lengua cualquiera.

Código de caracteres:

Código numérico asignado a cada carácter de un alfabeto:

■ **ASCII**: Alfabeto inglés (EEUU) codificado en 7 bits por carácter

@ 80 A 81 B 82 C 83

UNICODE: Código internacionalizado que soporta muchas lenguas

- Codificación del UNICODE:

Representación binaria interna del UNICODE:

- **UTF-8**: Código de longitud variable >= a 1 byte, eficiente para lenguas latinas
- **UTF-16**: Codifica planos BMP en 16 bits
- UTF-32: Código de longitud fija: 4 bytes eficiente con planos que no son BMP.



El código **ASCII** (*American National Standard Code for Information Interchange*) es una codificación alfanumérica usada para el intercambio de información basada en el alfabeto latino del inglés.

El conjunto de caracteres de **ASCII** fue creado como estándar en 1967. Fue diseñado en un principio usando 8 bits (1 byte), dejando un bit libre para el control de errores, por lo que puede representar **2**⁷=**128** caracteres (abecedario, cifras, puntuación y caracteres de control).

El código ASCII define una relación entre caracteres específicos y secuencias de bits pero sin definir la apariencia o formato del texto.

El código ASCII se divide básicamente en:

- Caracteres de control (del 0 al 31): usados como formato interno de editores y ficheros.
- Caracteres imprimibles: letras minúsculas, mayúsculas, signos, etc.



Codificación de la tabla ASCII original (7 bits):

https://ascii.cl/es/

Decimal	Hex C	har	Decimal	Hex C	har	Decima	al Hex Cl	har	Decima	al Hex C	har
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	
1	1	[START OF HEADING]	33	21	1	65	41	Α	97	61	а
2	2	[START OF TEXT]	34	22	II	66	42	В	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	C
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	1	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	Н	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	1	105	69	i
10	Α	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	В	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	С	[FORM FEED]	44	2C		76	4C	L	108	6C	T.
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	Е	[SHIFT OUT]	46	2E		78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	0	111	6F	0
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	р
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	S
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	V
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Υ	121	79	у
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	1	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]



Debido a que 8 bits no son suficientes para representar todos los alfabetos del mundo, continuaron apareciendo variantes ASCII de 8 bits incompatibles entre sí. Estas variantes se llaman a veces ASCII extendido, sin ser parte del estándar ANSI.

Hay varios conjuntos de **ASCII extendido**, cada uno de los cuales contiene codificaciones para muchos lenguajes:

- Windows code pages, usado en aplicaciones gráficas Windows.
- OEM code pages, usando en aplicaciones de consola Windows.
- ISO-8859 es un estándar ISO para codificación en 8 bits. Tiene 16 partes. La primera se llama ISO-8859-1, también conocida como Latin-1, que cubre la mayoría de lenguajes de Europa occidental.

En ASCII solo es posible trabajar con un alfabeto a la vez. ASCII tampoco es válido para representar alfabetos asiáticos, porque contienen miles de caracteres.



Codificación de la tabla ASCII extendida (8 bits):

(teres ASCII			2000		res As			ASCII extendido (Página de código 437)								
		control			- 1		mible					- (na de		0 43)	
00	NULL	(carácter nulo)		32	espacio	64	@	96	•		128	Ç	160	á	192	L	224	Ó
01	SOH	(inicio encabezado)		33	!	65	A	97	a		129	ü	161	ĺ	193	1	225	В
02	STX	(inicio texto)		34		66	В	98	b		130	é	162	Ó	194	Т	226	Ô
03	ETX	(fin de texto)		35	#	67	С	99	C		131	â	163	ú	195	-	227	Ò
04	EOT	(fin transmisión)		36	\$	68	D	100	d		132	ä	164	ñ	196	-	228	ő
05	ENQ	(consulta)		37	%	69	E	101	е		133	à	165	Ñ	197	+	229	Õ
06	ACK	(reconocimiento)		38	8.	70	F	102	f		134	å	166	3	198	ã	230	μ
07	BEL	(timbre)		39	•	71	G	103	g		135	ç	167	0	199	Ã	231	þ
08	BS	(retroceso)		40	(72	Н	104	h		136	ê	168	٤	200	L	232	Þ
09	HT	(tab horizontal)		41)	73	I	105	İ		137	ë	169	®	201	IT JL	233	Ú
10	LF	(nueva línea)		42	*	74	J	106	j		138	è	170	7	202	┸	234	Û
11	VT	(tab vertical)	器	43	+	75	K	107	k		139	Ϊ	171	1/2	203	T	235	Ù
12	FF	(nueva página)		44	,	76	L.	108	- 1		140	î	172	1/4	204	T	236	ý Ý
13	CR	(retorno de carro)		45	-	77	M	109	m		141	ì	173	I	205	==	237	
14	SO	(desplaza afuera)		46		78	N	110	n		142	Ä	174	**	206	#	238	- 10 - 1
15	SI	(desplaza adentro)		47	1	79	0	111	0		143	Å	175	*	207	#	239	•
16	DLE	(esc.vínculo datos)		48	0	80	P	112	р		144	É	176		208	Ŏ	240	5
17	DC1	(control disp. 1)		49	1	81	Q	113	q		145	æ	177	200	209	Ð	241	±
18	DC2	(control disp. 2)		50	2	82	R	114	Г		146	Æ	178	#	210	Ê	242	_
19	DC3	(control disp. 3)		51	3	83	S	115	S		147	Ô	179		211	Ë	243	3/4
20	DC4	(control disp. 4)		52	4	84	T	116	t		148	Ö	180	4	212	È	244	¶
21	NAK	(conf. negativa)		53	5	85	U	117	u		149	Ò	181	Á	213	Ļ	245	§
22	SYN	(inactividad sínc)		54	6	86	V	118	V		150	û	182	Ä	214	ĺ	246	÷
23	ETB	(fin bloque trans)		55	7	87	W	119	W		151	ù	183	Á	215	Ï	247	,
24	CAN	(cancelar)		56	8	88	X	120	×		152	ÿ	184	0	216	Ϊ	248	٥
25	EM	(fin del medio)		57	9	89	Υ	121	У		153	Ö	185	4	217	7	249	15
26	SUB	(sustitución)		58	:	90	Z	122	Z		154	Ü	186		218	Г	250	•
27	ESC	(escape)		59	;	91	[123	{		155	ø	187	٦	219		251	1
28	FS	(sep. archivos)		60	<	92	1	124	1		156	£	188	1	220		252	2
29	GS	(sep. grupos)		61	-	93]	125	}		157	Ø	189	¢	221	į	253	2
30	RS	(sep. registros)		62	>	94	٨	126	~		158	×	190	¥	222	Ì	254	•
31	US	(sep. unidades)		63	?	95	_				159	f	191	٦	223	-	255	nbsp
127	DEL	(suprimir)																



Para traducir al código ASCII cualquier texto, sólo es necesario buscar el código hexadecimal de cada símbolo y escribir su equivalente binario, recordando que cada carácter siempre se codifica a 8 bits.

✓ Por ejemplo, escribir 'Informatica' en código ASCII binario:

Buscar en la tabla ASCII anterior la equivalencia alfanumérica exacta del símbolo con su valor hexadecimal y convertirlo después a binario.

Letra ASCII (dec hex).	ASCII bin.	Letra ASCII hex.	ASCII bin.
I		a	
n		t	
f		i	
0		С	
r		а	
m			



Para traducir al código ASCII cualquier texto, sólo es necesario buscar el código hexadecimal de cada símbolo y escribir su equivalente binario, recordando que cada carácter siempre se codifica a 8 bits.

✓ Por ejemplo, escribir 'Informatica' en código ASCII binario:

Buscar en la tabla ASCII anterior la equivalencia alfanumérica exacta del símbolo con su valor hexadecimal y convertirlo después a binario.

Letra ASCII (dec hex).	ASCII bin.	Letra ASCII hex.	ASCII bin.
l (<mark>73</mark> 49)		a (97 61)	
n (110 6E)		t (116 74)	
f (102 66)		i (105 69)	
o (111 6F)		c (99 63)	
r (114 72)		a (97 61)	
m (109 6D)			



Para traducir al código ASCII cualquier texto, sólo es necesario buscar el código hexadecimal de cada símbolo y escribir su equivalente binario, recordando que cada carácter siempre se codifica a 8 bits.

✓ Por ejemplo, escribir 'Informatica' en código ASCII binario:

Buscar en la tabla ASCII anterior la equivalencia alfanumérica exacta del símbolo con su valor hexadecimal y convertirlo después a binario.

Letra ASCII (dec hex).	ASCII bin.	Letra ASCII hex.	ASCII bin.
l (<mark>73</mark> 49)	0101001	a (97 61)	1100001
n (110 6E)	1101110	t (116 74)	1110100
f (102 66)	1100110	i (105 69)	1101001
o (111 6F)	1101111	c (99 63)	1100011
r (114 72)	1110010	a (97 61)	1100001
m (109 6D)	1101101		



UNICODE es un consorcio internacional nacido en **1991** que define normas de internacionalización (I18N): Códigos de caracteres (Unicode), símbolos, librerías software, formatos. http://www.unicode.org



El **código UNICODE** es un código de caracteres internacionalizado. Es el resultado más conocido del consorcio.

Puede representar la gran mayoría de lenguas presentes, pasadas e incluso inventadas.





En Unicode existen distintas formas de codificar un mismo carácter según el formato. Existen los siguientes formatos *UTF-8, UTF-16 y UTF-32.*

- **UTF-8** utiliza <u>1 byte</u> para representar caracteres en el set ASCII, dos bytes para caracteres en otros bloques alfabéticos y tres bytes para el resto del BMP. Para los caracteres complementarios se utilizan 4 bytes.
- **UTF-16** utiliza <u>2 bytes</u> para cualquier carácter en el BMP y 4 bytes para los caracteres complementarios.
- UTF-32 emplea 4 bytes para todos los caracteres.

	A	א	好	不
Punto de código	U+0041	U+05D0	U+597D	U+233B4
UTF-8	41	D7 90	E5 A5 BD	F0 A3 8E B4
UTF-16	00 41	05 D0	59 7D	D8 4C DF B4
UTF-32	00 00 00 41	00 00 05 D0	00 00 59 7D	00 02 33 B4

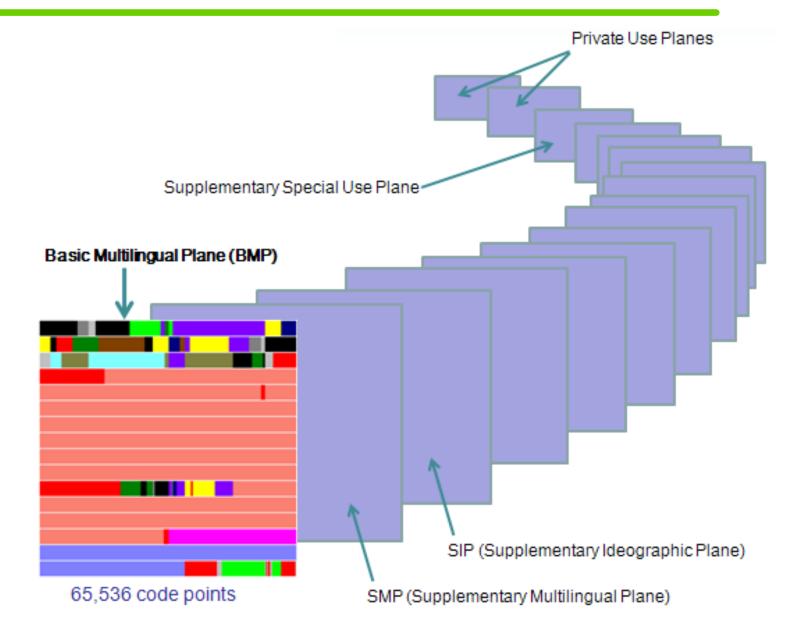


- Los caracteres se agrupan a su vez en **planos**
 - ► Cada plano por tanto se codifica en 2 bytes 2 bytes (16 bits): 2¹⁶ = **64536** caracteres

	000	001	002	003	004	005	006	007
0	NUL	DLE	SP	0	@	P	,	p
	0000	0010	0020	0030	0040	0050	0060	0070
1	SOH	DC1	0021	1	A	Q	a	q
2	STX	DC2	"	2	В	R	b	r

- Planos más importantes:
 - BMP (Basic Multilingual Plane) Agrupa los símbolos más habituales de la mayoría de lenguas actuales
 - SMP (Suplementary Multilingual Plane): Lenguas antiguas y más
 - o SIP (Supl. Ideographic Plane): Ext. CJK (China, Japón, Korea)
 - o **TIP** (Tiertary Ideographic Plane): Lenguas antiguas asiáticas
 - SSP (Suplementary Special-purpose Plane): usos especiales







- Los caracteres en Unicode se escriben usando el formato *U+xxxx* donde las xxxx son de cuatro a seis dígitos en sistema de numeración hexadecimal.
- A partir de Unicode 7.0 el rango válido de puntos de código va de 0 a 10FFFF16. El hexadecimal se usa por conveniencia en lugar del binario, porque es más fácil recordar. La versión más actual es la 12.0 (marzo 2019) con soporte para 136690 caracteres.
- Por ejemplo, Unicode asigna el número 65 a la letra a latina mayúscula. El punto de código correspondiente es U+0041 porque 65decimal = 0x41hexadecimal.
- Unicode es compatible con codificaciones anteriores ya que los 256 primeros caracteres de Unicode coinciden con los caracteres de ISO-8859-1. Esto hace que la mayoría de texto en uso requiera solo un byte por carácter.
- UTF-8 y UTF-16 son codificaciones de amplitud variable. Esto significa que si un carácter se puede representar con un sólo byte, UTF-8 empleará sólo un byte. Si requiere dos bytes, usará dos, y así sucesivamente.





Mapa de caracteres		_	\Box X							
Fuente: Arial		×	Ayuda							
! " # \$ %	& ' () * + , -	. / 0 1	2 3 4 ^							
5 6 7 8 9	: ; < = > ? @ A	B C D E	FGH							
I J K L M	NOPQRSTU	V W X Y	Z [\							
] ^ _ ` a	b c d e f g h i	j k l m	n o p							
q r s t u	v w x y z { }	~ i ¢	£ ¤ ¥							
	« ¬ - ® - ° ± 2	³ ′ µ ¶	. ا							
0 » 1/4 1/2 3/4	•	ÇÈÉÉ	ËÌÍ							
ÎÜĐÑÒ	ÓÔÕÖרÙÚ	ÛÜÝÞ	ß à á							
â ã ä å æ	ç è é ê ë ì í î	ï ð ñ ò	ó ô õ							
ö ÷ ø ù ú	û ü ý þ ÿ Ā ā Ă	ă Ą ą Ć	ć Ĉ ĉ							
Caracteres para copiar: Seleccionar Copiar Vista avanzada										
Juego de caracteres: Unicode Ir a Unicode:										
Agrupar por:										
Buscar: U+0021: Exclamación de cierre										



Unicode BMP: subconjunto ASCII (basic Latin)

www.unicode.org/charts/PDF/U0000.pdf

	000	001	002	003	004	005	006	007
0	NUL 0000	DLE 0010	SP 0020	0030	<u>@</u>	P	0060	p
1	SOH 0001	DC1	0021	1	A 0041	Q 0061	a 0061	q
2	STX 0002	DC2	0022	2	B	R	b	r 0072
3	ETX	DC3	# 0023	3	C 0043	S 0053	C 0063	S 0073
4	EOT 0004	DC4	\$	4	D 0044	T	d	t
5	ENQ 0005	NAK 0015	% 0025	5	E 0045	U 0065	e 0065	u 0075
6	ACK 0006	SYN 0016	& 0026	6	F 0046	V 0066	f	V 0076
7	BEL 0007	(ETB)	0027	7	G 0047	W 0067	g 0067	W 0077

8	BS 0008	(CAN)	0028	8	H 0048	X 0058	h	X 0078
9	HT 0009	EM 3	0029	9	I 0049	Y	i 0069	y 0079
Α	LF 000A	SUB 001A	* 002A	003A	J	Z 005A	j	Z
В	(V T	ESC 001B	+ 002B	• • • •	K 0048	0058	k	{ 007B
С	(FF)	[FS]	9 002C	< 003C	L 004C	005C	1	007C
D	CR 0000	GS OO1D	 002D		M 004D] 005D	m	}
Е	SO	RS 001E	• 002E	> 003E	N 004E	∧	n 006E	~ 007E
F	SI	[US]	/ 002F	?	O 004F	006F	O 006F	DEL 007F



Unicode BMP: subconjunto ASCII (basic Latin)

www.unicode.org/charts/PDF/U0000.pdf

	000	001	002	003	004	005	006	007	U+ 0 ()41	001	002	003	004	005	006	007 [
0	NUL 0000	DLE 0010	SP 0020	0	<u>@</u>	P	0060	p	8	BS 0008	(CAN)	0028	8	H 0048	X 0058	h	X 0078
1	SOH 0001	DC1	0021	1	A 0041	Q 0061	a 0061	q	9	HT 0009	EM 0019	0029	9	I 0049	Y 0059	i	y 0079
2	STX	DC2	0022	2	B 0042	R	b	r 0072	А	LF 000A	SUB 001A	* 002A	• • 003A	J	Z 005A	j	Z 007A
3	ETX	DC3	# 0023	3	C 0043	S 0053	C 0063	S 0073	В	(V T	ESC 001B	+ 002B	• • • •	K	0058	k	{ 007B
4	EOT 0004	DC4	\$	4	D 0044	T	d	t	C	(FF)	FS 001C	9 002C	< 009C	L	005C	1	007C
5	ENQ 0005	NAK 0015	% 0025	5	E 0045	U 0065	e 0065	0+0 0075	D	CR 0000	GS 001D	 002D	 003D	M 004D] 005D	m	} 007D
6	0006	SYN 0016	& 0026	6	F 0046	V 0056	f	V	E	SO	RS OO1E	• 002E	> 003E	N 004E	∧	n	~ 007E
7	BEL 0007	(ETB)	0027	7	G	W 0067	g 0067	W	F	SI	US 001F	/ 002F	?	O 004F	006F	O 006F	DEL 007F



✓ Buscar en el mapa de caracteres de Windows los valores hexadecimales de los siguientes caracteres de texto Unicode:

Carácter Unicode	Valor hexadecimal
=	
μ	
Т	
æ	
&	
@	
\odot	
Ф	
•	



✓ Buscar en el mapa de caracteres de Windows los valores hexadecimales de los siguientes caracteres de texto Unicode:

Carácter Unicode	Valor hexadecimal
=	U+003D
μ	U+00B5
Т	U+ 0054
æ	U+ 00E6
&	U+ 0026
@	U+ 0040
\odot	U+263A
Ф	U+ 0424
•	U+2660

Código binario BCD



Todo lo que hemos visto hasta ahora se refiere al llamado código binario natural. Sin embargo existen distintos tipos de códigos binarios, que son empleados según favorezcan el proceso que se va a realizar con ellos, ya que son más eficientes en la ejecución de determinadas operaciones.

Códigos BCD (Decimal codificado en binario).

Con estos códigos, para representar un número decimal en binario, se transforman cada una de las cifras que constituyen el número decimal separadamente, en el caso anterior el número 35, transformado en binario sería el resultado de transformar primero 3, y después 5.

Código BCD natural (8421)

Es un código con peso o ponderado, es decir el número decimal equivalente es el resultado de sumar el valor de la posición de los dígitos binarios que constituyen el código. En este código los pesos de los dígitos son las potencias sucesivas de dos, es decir 2^3 (8), 2^2 (4), 2^1 (2) y 2^0 (1). se transcriben las cifras decimales a binario y viceversa, según la posición que ocupan. Ejemplo $35 = 0011 \ 0111$.

Código binario BCD



En sistemas de computación el código **BCD** (Binary-Coded Decimal) o Decimal codificado en binario es un estándar para representar números decimales en el sistema binario, en donde cada dígito decimal es codificado **con una secuencia de 4 bits**.

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



Hay que recordar que cuando hablamos de cantidades de información y sus unidades de medida, éstas serán múltiplos de **potencias de 2**, aumentando en **2**¹⁰ cada vez.

Nombre	Abreviatura	Factor
Kilo	K	2 ¹⁰ = 1024
Mega	M	2 ²⁰ = 1 048 576
Giga	G	2 ³⁰ = 1 073 741 824
Tera	Т	2 ⁴⁰ = 1 099 511 627 776
Peta	P	2 ⁵⁰ = 1 125 899 906 842 624
Exa	E	2 ⁶⁰ = 1 152 921 504 606 846 976
Zetta	Z	2 ⁷⁰ = 1 180 591 620 717 411 303 424
Yotta	Y	2 ⁸⁰ = 1 208 925 819 614 629 174 706 176

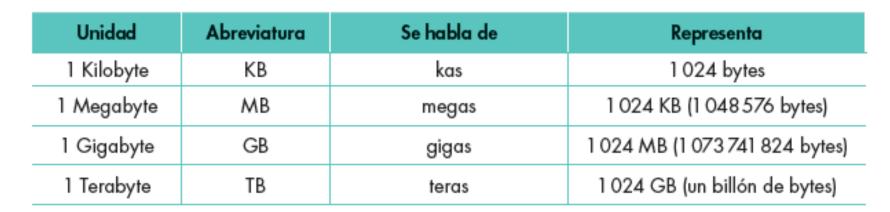
• 1 Bit: es la unidad básica de medida de almacenamiento de información utilizada en una computadora.

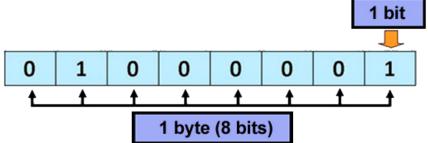
• 1 Byte: combinación de 8 bits.

• 1 Kbyte (KB): 1024 bytes

• 1 Megabyte (MB): 1024 Kbytes

• 1 Gigabyte (GB): 1024 Mbytes











Unidades en comunicaciones

Para representar la velocidad de transmisión de datos en telecomunicaciones vamos a utilizar el bit por segundo o (b/s) o (bps) y sus múltiplos. Debido a que es una medida en función del tiempo, se introduce esta magnitud elemental.

Nombre de magnitud	Símbolo	Factor en sistema decimal	Valor en sistema binario (en bits)
bit por segundo	bps	10 ⁰	1
Kilobit por segundo	Kbps	10 ³	1.000
Megabit por segundo	Mbps	10 ⁶	1.000.000
Gigabit por segundo	Gbps	10 ⁹	1.000.000.000
Terabit por segundo	Tbps	10 ¹²	1.000.000.000



Unidades en comunicaciones

Velocidad de Internet	Conversión a MB	Tiempo para descargar un fichero de 100 MB
10 Mbps	1,25 MB/s	80 segundos
100 Mbps	12,5 MB/s	8 segundos
200 Mbps	25 MB/s	4 segundos
300 Mbps	37,5 MB/s	2,67 segundos
600 Mbps	75 MB/s	1,34 segundos
1.000 Mbps	125 MB/s	0,8 segundos

Ejercicios de conversión



✓ Continua las siguientes secuencias de números en la base especificada:

Base 2: 1100, 1101 →

Base 2: 1111 →

Base 8: 65, 66, 67 \rightarrow

Base 16: FFC, FFD, FFE →

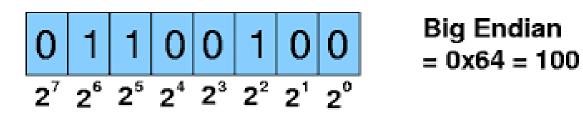
Base 16: FFEF →

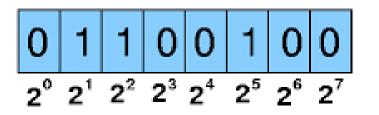
Little endian y Big endian



Little endian y Big endian se refieren al <u>orden</u> que las máquinas asignan a los bytes que representan números o valores numéricos internos.

- **Big endian** asigna los bytes menos significativos en el extremo más alto. Este formato que puede parecer una forma más "natural" de escritura es utilizado por procesadores usados en máquinas Apple entre otras.
- Little endian asigna los bytes menos significativos en el extremo más bajo de la memoria. Este formato es adoptado por la mayoría de procesadores Intel y AMD.



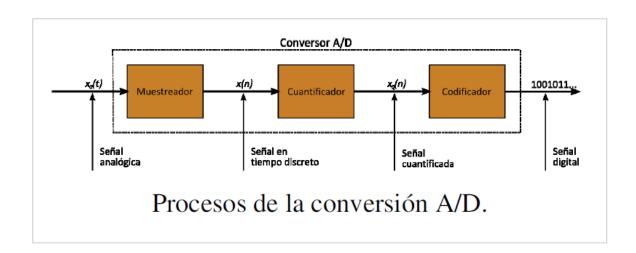


Little Endian = 0x26 = 38

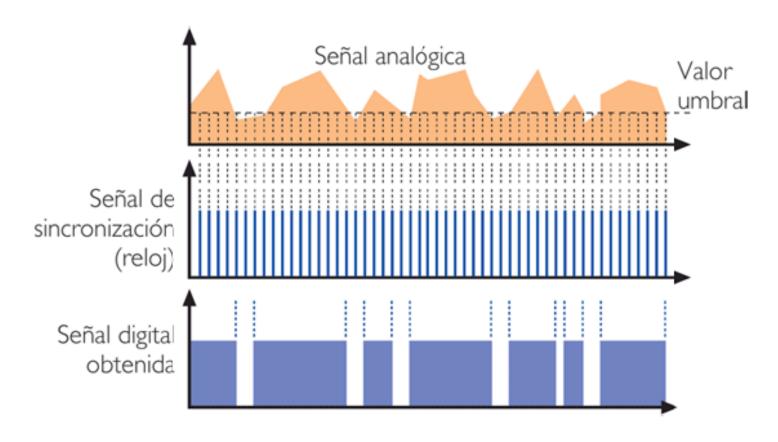


Un ordenador o cualquier sistema de control basado en un microprocesador no puede interpretar señales analógicas, ya que sólo utiliza señales digitales como bien sabemos. Es necesario traducir, o transformar en señales binarias, lo que se denomina proceso de digitalización o conversión de señales analógicas a digitales.

La conversión **analógica-digital** (CAD) o <u>digitalización</u> consiste en la transcripción de señales analógicas en señales digitales, con el propósito de facilitar su procesamiento (codificación, compresión, etc.) y hacer la señal resultante la digital más inmune al ruido y otras interferencias.





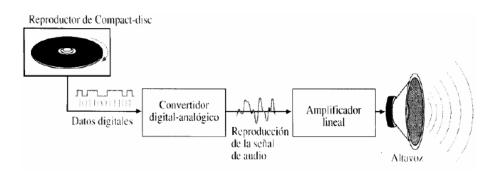


Digitalización por muestreado de una señal analógica



Ventajas de la digitalización:

1. Cuando una señal digital es atenuada o experimenta perturbaciones leves, puede ser reconstruida y amplificada mediante sistemas de regeneración de señales.



Sistema Digital - Analógico.

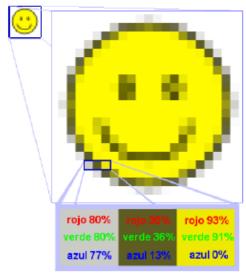
- 2. Cuenta con sistemas de detección y corrección de errores, que se utilizan cuando la señal llega al receptor; entonces comprueban (uso de redundancia) la señal, primero para detectar algún error, y, algunos sistemas, pueden luego corregir alguno o todos los errores detectados previamente.
- 3. Facilidad para el procesamiento de la señal. Cualquier operación es fácilmente realizable a través de cualquier software de edición o procesamiento de señal.
- 4. La señal digital permite la multigeneración infinita sin pérdidas de calidad.
- 5. Es posible aplicar técnicas de compresión de datos sin pérdidas o técnicas de compresión con pérdidas basados en la codificación perceptual mucho más eficientes que con señales analógicas.



Una imagen en **mapa de bits**, es una estructura o fichero de datos digital que representa una matriz de píxeles o puntos de color, que se puede visualizar en un monitor o cualquier otro dispositivo de representación

A las imágenes en mapa de bits se las suele definir por su altura y anchura (en píxeles) y por su **profundidad de color** (en bits por píxel), que determina el número de colores distintos que se pueden almacenar en cada punto individual, y por lo tanto la calidad del color de la imagen.





Detalle de una imagen en mapa de bits. Si hacemos zoom sobre esta imagen, podemos ver los puntos (píxeles) que la conforman, representados como cuadrados.



