UT2.1: Representación de la información

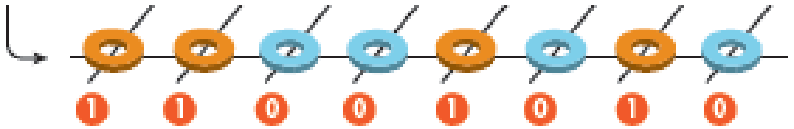
FONDO SOCIAL EUROPEO

Javier Traseira

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.

Sistemas de numeración

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:

$$NÚM = \Sigma X_i \cdot B^i$$

$$283 = 2 \cdot 10^2 + 8 \cdot 10^1 + 3 \cdot 10^0 = 200 + 80 + 3$$

$$NÚM = X_n \cdot 10^n + \ldots + X_2 \cdot 10^2 + X_1 \cdot 10^1 + X_0 \cdot 10^0 + X_{-1} \cdot 10^{-1} + X_{-2} \cdot 10^{-2} \ldots + X_{-N} \cdot 10^{-N}$$

Sistemas de numeración

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

$$NÚM = \Sigma X_i \cdot B^i$$

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** · 100 + **4** · 10 + **5** · 1
O lo que es lo mismo: **745** = **7** · 10^2 + **4** · 10^1 + **5** · 10^0

Sistemas de numeración

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.

Sistemas de numeración

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
…	….

Sistemas de numeración

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	B
12	C
13	D
14	E
15	F
16	10
17	11

Sistemas de numeración

Hexadecimal

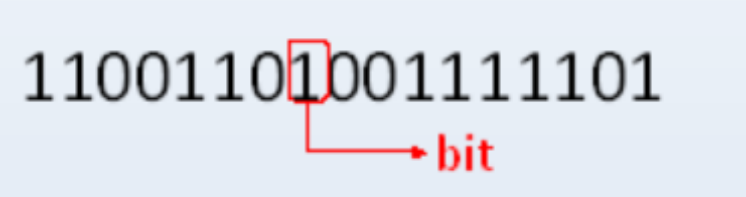
Decimal	Octal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	10
9	11
10	12
11	13
12	14
13	15
14	16
15	17
16	20
…	…

Sistemas de numeración

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	A
11	01011	13	B
12	01100	14	C
13	01101	15	D
14	01110	16	E
15	01111	17	F
16	10000	20	10
17	10001	21	11

Sistemas de numeración

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

Sistemas de numeración

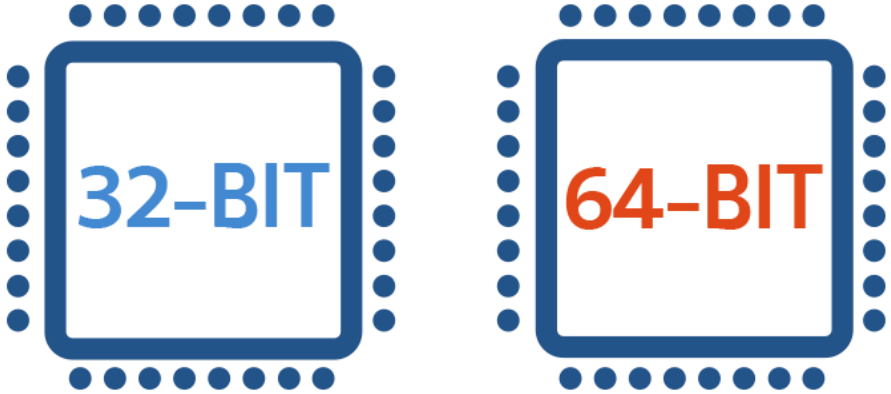
Es necesario saber interpretar el código binario para poder entender las operaciones que en muchas ocasiones se realizan dentro del ordenador. Para ello, debemos aprender a pasar números binarios a decimales y a la inversa, es decir la **conversión entre sistemas de numeración**.

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)

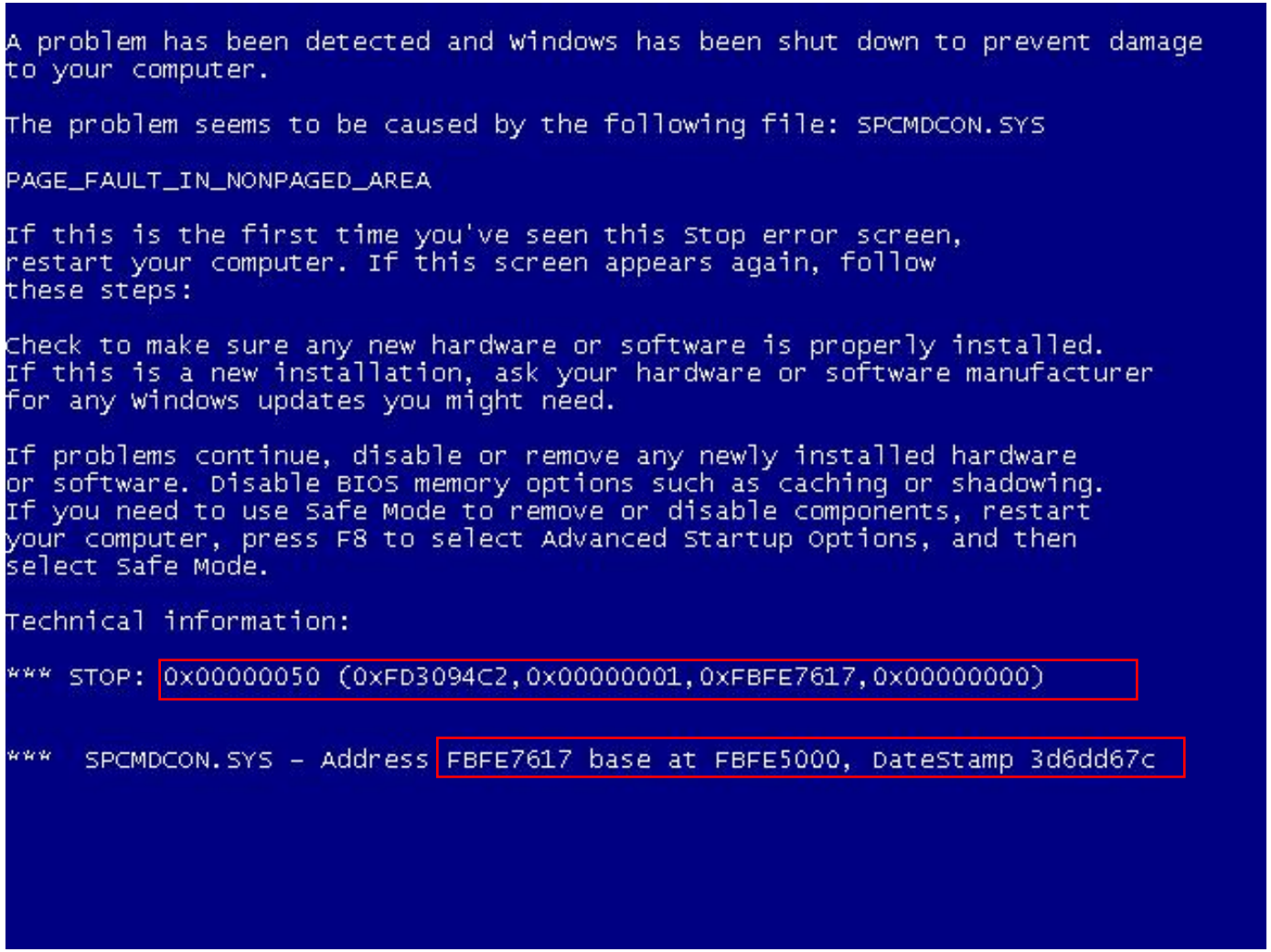
Debido a la estructura de 64 bits de un microprocesador las direcciones de memoria se expresan a menudo en hexadecimal. Por ejemplo, para no tener que escribir 1111110101000000000000010101100 podemos escribir 3F5000AC en hexadecimal.

Sistemas de numeración

- Un procesador y un bus de **32 bits** permite especificar a la CPU 2^{32}= 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 2^{64} 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)

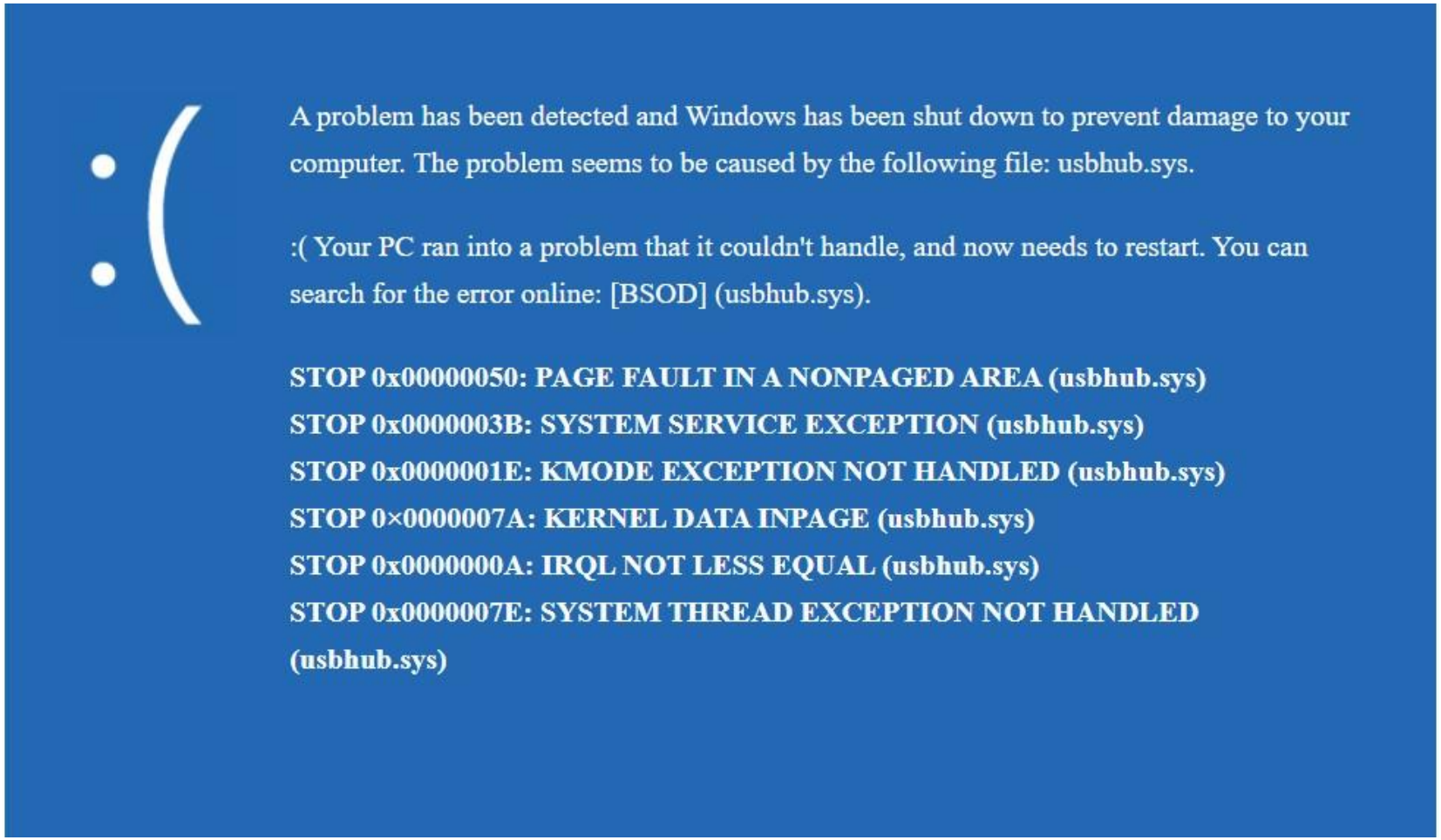


Sistemas de numeración



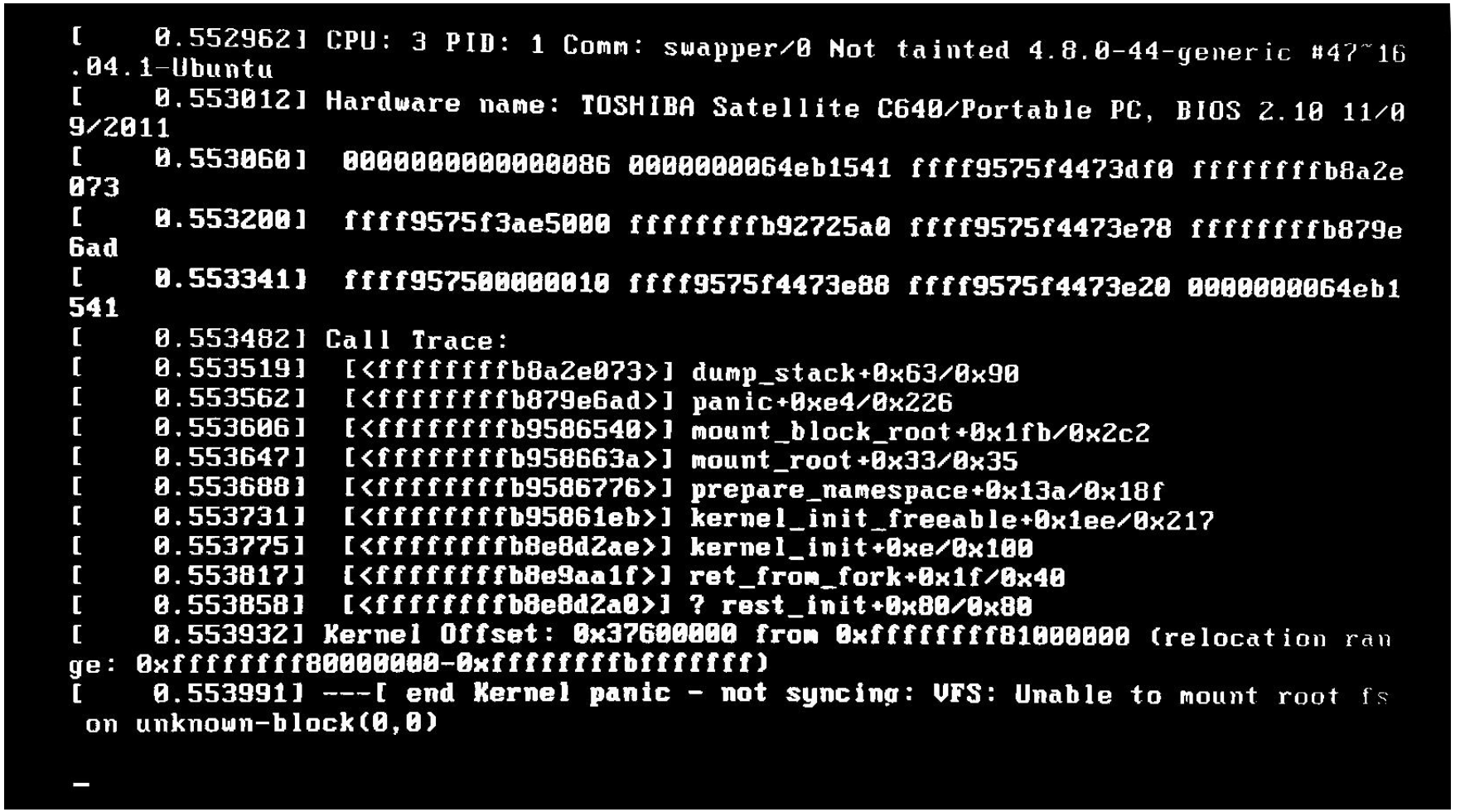
Pantallazo de la muerte (*blue screen of death*) en un sistema Windows clásico

Sistemas de numeración



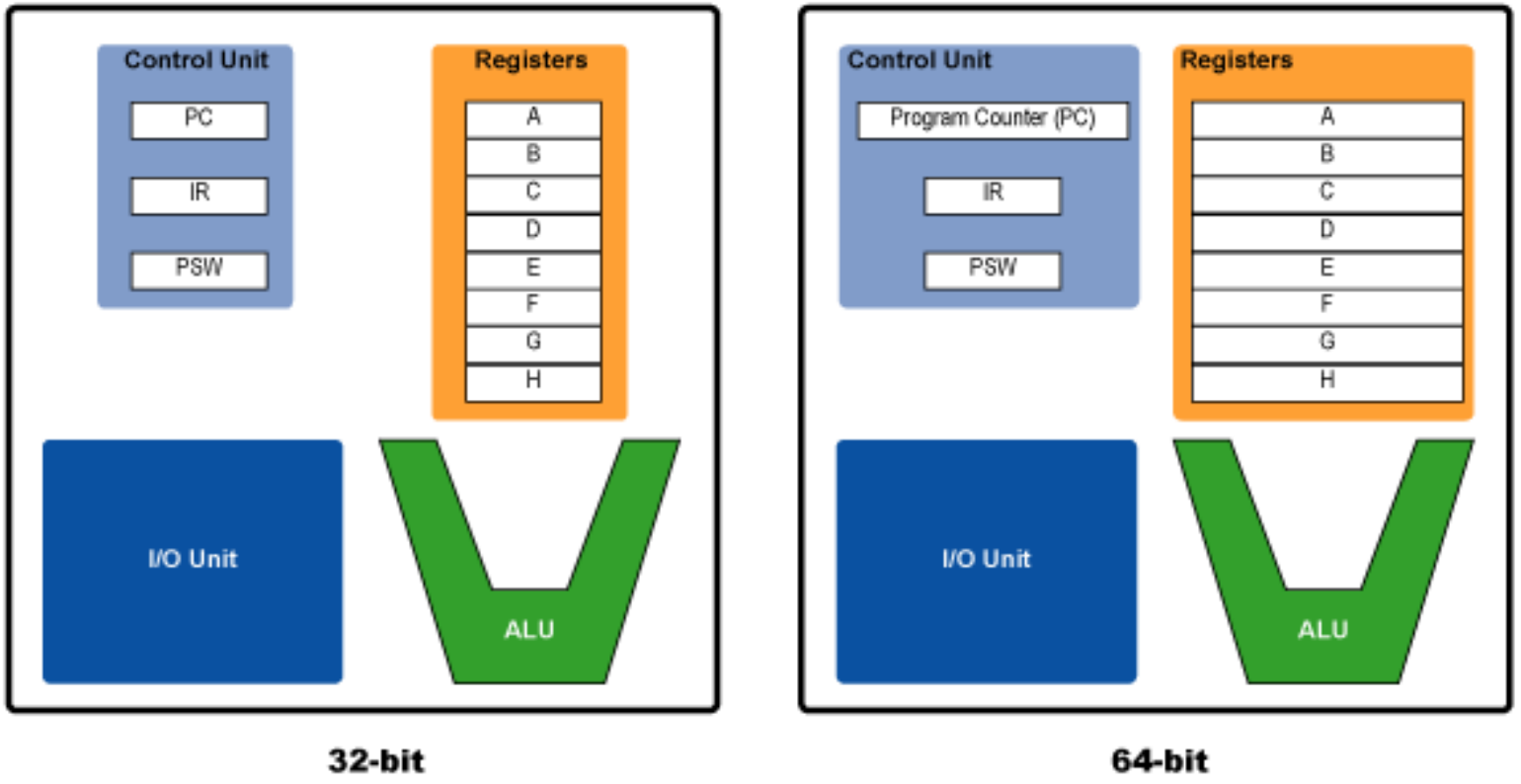
Pantallazo de la muerte (*blue screen of death*) en otro sistema Windows actual

Sistemas de numeración



Kernel Panic en un sistema Linux

Sistemas de numeración



Conversiones sistemas numeración

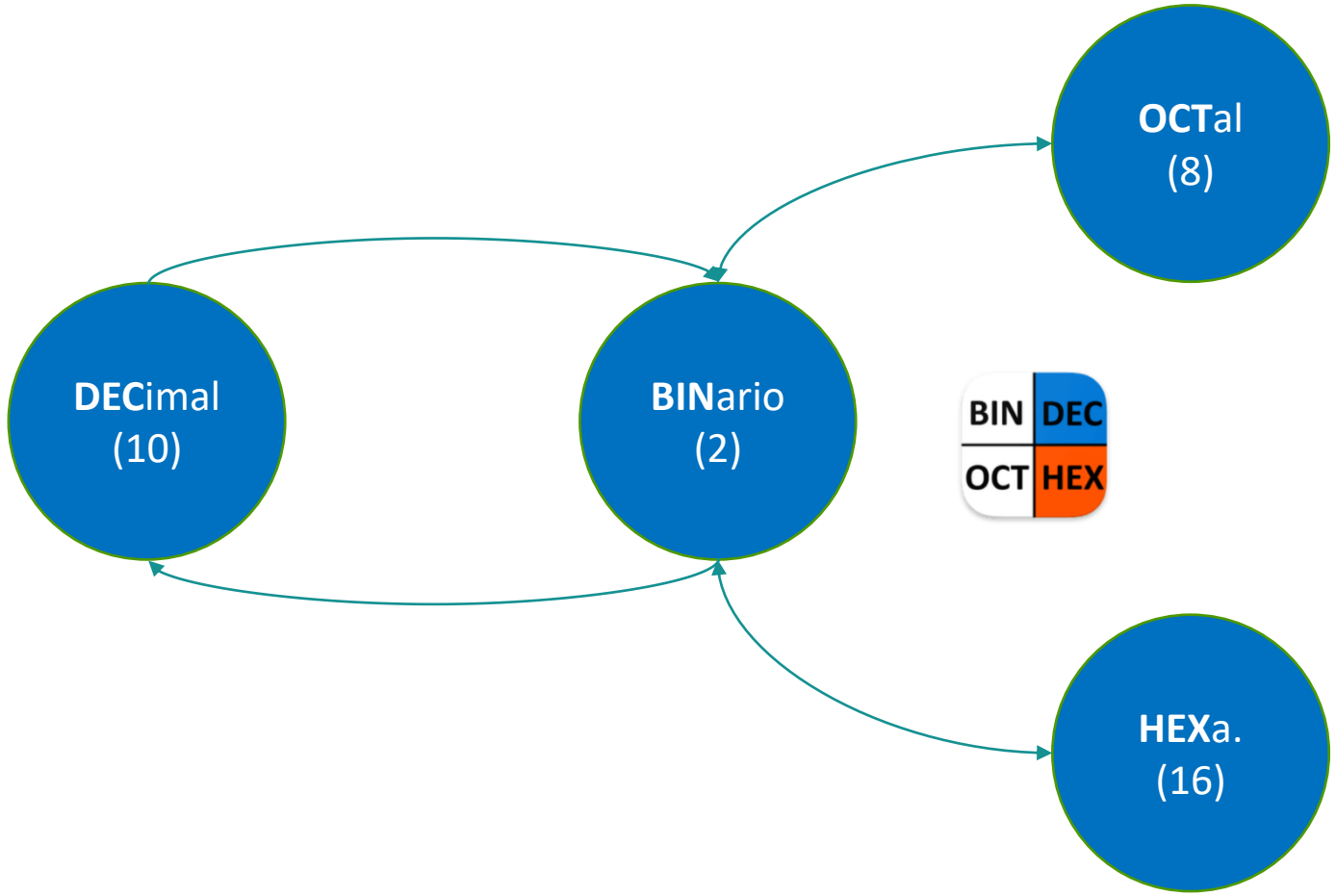
- Repaso de **operaciones matemáticas con potencias**:

- Potencia de exponente 0: $a^0 = 1$
  $16^0 = 1 \quad 2^0 = 1$

- Potencia de exponente 1: $a^1 = a$
  $2^1 = 2 \quad 6^1 = 6$

- Potencia de base n: $n^a = n \cdot n \cdot \ldots_{a\ veces} \cdot n$
  $16^3 = 16 \cdot 16 \cdot 16$
  $2^5 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2$
  $1^{354} = 1$

Conversiones sistemas numeración

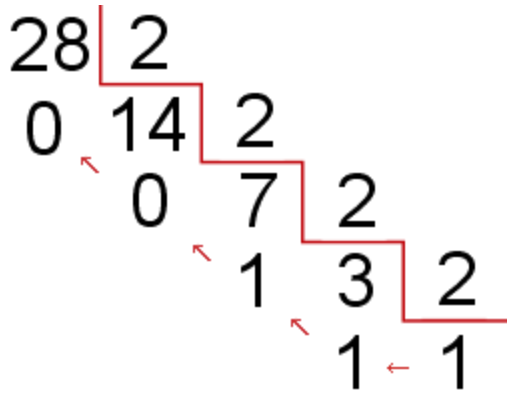


Conversiones sistemas numeración

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**:



28 = 11100

Conversiones sistemas numeración

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**:

Conversiones sistemas numeración

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**:

52	2	
0	26	2
	0	13	2
		1	6	2
			0	3	2
				1	**1**

52 = 110100

Conversiones sistemas numeración

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.

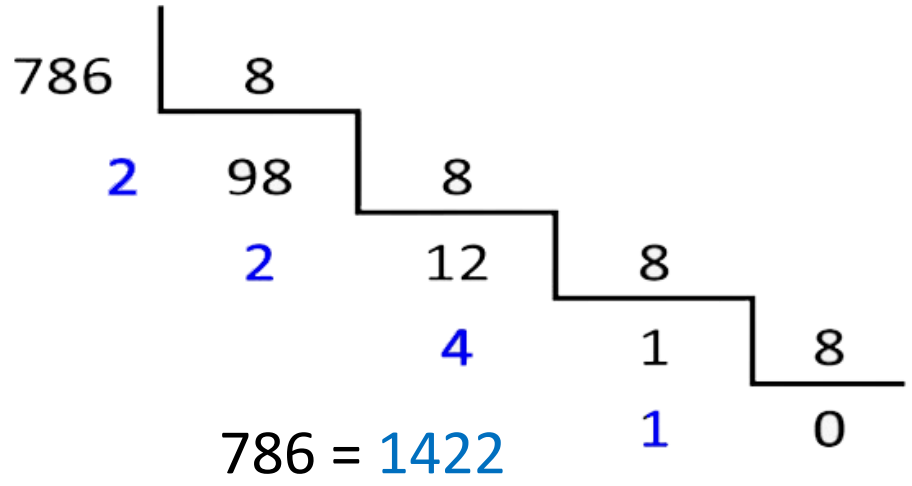
- ✓ Convertir en octal el número decimal **786**:

Conversiones sistemas numeración

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**:



Conversiones sistemas numeración

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.

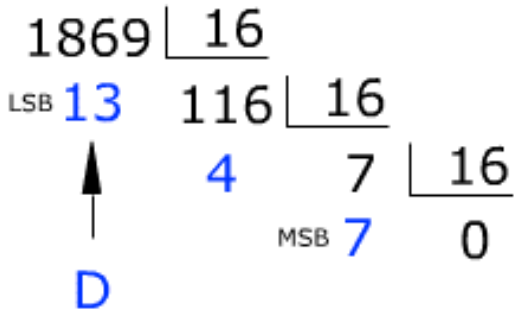
- ✓ Convertir en hexadecimal el número decimal **1869**:

Conversiones sistemas numeración

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

Conversiones sistemas numeración

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**:

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

$$512 + 0 + 0 + 0 + 0 + 16 + 8 + 0 + 2 + 1 \rightarrow \mathbf{539}$$

2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
512	256	128	64	32	16	8	4	2	1
1	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**1**	**1**

512
16
8
2
\+ 1

= 539

Conversiones sistemas numeración

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**:

Conversiones sistemas numeración

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**:

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

$$64 + 0 + 0 + 8 + 0 + 2 + 0 \rightarrow \mathbf{74}$$

- ✓ Convertir en decimal el número binario **1111100**:

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

$$64 + 32 + 16 + 8 + 0 + 2 + 0 \rightarrow \mathbf{124}$$

Conversiones sistemas numeración

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

Conversiones sistemas numeración

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

111 = 7

001 = 1

011 = 3 → Luego el número será 317

Conversiones sistemas numeración

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	A
0011	3	3	**1011**	11	B
0100	4	4	**1100**	12	C
0101	5	5	**1101**	13	D
0110	6	6	**1110**	14	E
0111	7	7	**1111**	15	F

Conversiones sistemas numeración

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

Conversiones sistemas numeración

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

Conversiones sistemas numeración

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**:

Conversiones sistemas numeración

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**:

$$\mathbf{4}\cdot 8^2+\mathbf{2}\cdot 8^1+\mathbf{1}\cdot 8^0 = 4\cdot 64 + 2\cdot 8 + 1 = 256 + 16 + 1 \rightarrow 273$$

✓ Convertir en decimal el número octal **7014**:

$$\mathbf{7}\cdot 8^3+\mathbf{0}\cdot 8^2+\mathbf{1}\cdot 8^1+\mathbf{4}\cdot 8^0 = 7\cdot 512 + 0 + 8 + 4 = 256 + 16 + 1 \rightarrow 3596$$

Conversiones sistemas numeración

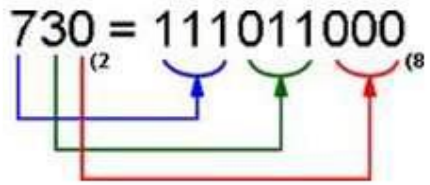
Conversión octal a binario

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 → 111 011 000



✓ Convertir en binario el número octal 1274

Conversiones sistemas numeración

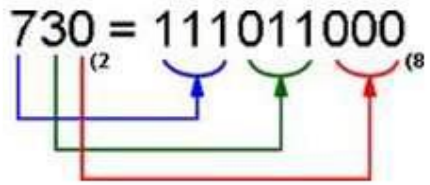
Conversión octal a binario

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 → 111 011 000



✓ Convertir en binario el número octal 1274

1 2 7 4 = 001 010 111 100 → 001 010 111 100

Conversiones sistemas numeración

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**:

$$\mathbf{9}\cdot 16^2+\mathbf{F}\cdot 16^1+\mathbf{2}\cdot 16^0 = 9\cdot 16^2+15\cdot 16^1+2\cdot 16^0= 9\cdot 256 + 240 + 2 \rightarrow 2546$$

✓ Convertir en decimal el número hexadecimal **B2F8**:

$$\mathbf{B}\cdot 16^3 + \mathbf{2}\cdot 16^2+ \mathbf{F}\cdot 16^1 + \mathbf{8}\cdot 16^0 = 11\cdot 16^3 + 2\cdot 16^2 + 15\cdot 16^1 + 8\cdot 1 =$$

$$11\cdot 4096 + 2\cdot 256 + 15\cdot 16+ 8\cdot 1 \rightarrow 45816$$

Conversiones sistemas numeración

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 $9F2_{16}$ a binario

$9F2_{16}$ = 9 → 1001 F → 1111 2 → 0010

$= 100111110010_2$

- ✓ Convertir en binario el número hexadecimal **BEE**:

- ✓ Convertir en decimal el número hexadecimal **70C558**:

Conversiones sistemas numeración

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 $9F2_{16}$ a binario

$9F2_{16}$ = 9 F 2

↓ ↓ ↓

1001 1111 0010

$= 100111110010_2$

- ✓ Convertir en binario el número hexadecimal **BEE**:

  11 14 14 → 1011 1110 1110

- ✓ Convertir en decimal el número hexadecimal **70C558**:

  7 0 12 5 5 8 → 0111 0000 1100 0101 0101 1000

Conversiones sistemas numeración

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 → 0730 *en octal*
111 = 7
000 = 0

Representación números negativos

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**

Representación números negativos

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.

Representación números negativos

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 doble representación del 0. En efecto, tenemos que 00000000 y 11111111 lo representan.

Representación números negativos

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.

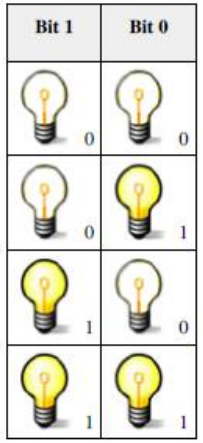
En este caso, el 0 (decimal) sólo tiene una única representación 00000000 y las operaciones aritméticas se pueden realizar mediante sumadores.

Sistemas de codificación

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.

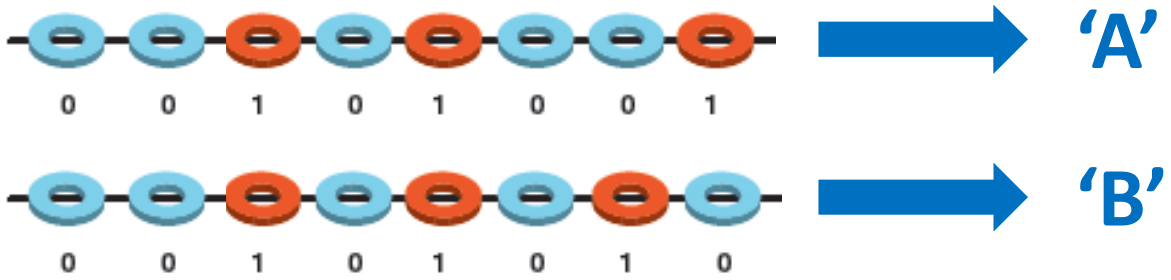
Los bits se pueden combinar para dar distintos resultados y generar distintos sistemas de codificación:



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.



Sistemas de codificación

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)

- **Codificaciones multimedia:**

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.

Sistemas de codificación

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

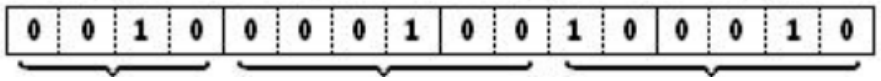
- **Código de caracteres:**

  Código numérico asignado a cada carácter de un alfabeto:

  - **ASCII:** Alfabeto inglés (EEUU) codificado en 7 bits por carácter
  - **UNICODE:** Código internacionalizado que soporta muchas lenguas

Char	Code
@	80
A	81
B	82
C	83

- **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.

Codificación alfanumérica ASCII

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 $\mathbf{2^7=128}$ 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 alfanumérica ASCII

- Codificación de la tabla ASCII original (7 bits): https://ascii.cl/es/

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	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	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
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	Y	121	79	y
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	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Codificación alfanumérica ASCII

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 alfanumérica ASCII

- Codificación de la tabla ASCII extendida (8 bits):

Caracteres ASCII de control		
00	NULL	(carácter nulo)
01	SOH	(inicio encabezado)
02	STX	(inicio texto)
03	ETX	(fin de texto)
04	EOT	(fin transmisión)
05	ENQ	(consulta)
06	ACK	(reconocimiento)
07	BEL	(timbre)
08	BS	(retroceso)
09	HT	(tab horizontal)
10	LF	(nueva línea)
11	VT	(tab vertical)
12	FF	(nueva página)
13	CR	(retorno de carro)
14	SO	(desplaza afuera)
15	SI	(desplaza adentro)
16	DLE	(esc.vínculo datos)
17	DC1	(control disp. 1)
18	DC2	(control disp. 2)
19	DC3	(control disp. 3)
20	DC4	(control disp. 4)
21	NAK	(conf. negativa)
22	SYN	(inactividad sínc)
23	ETB	(fin bloque trans)
24	CAN	(cancelar)
25	EM	(fin del medio)
26	SUB	(sustitución)
27	ESC	(escape)
28	FS	(sep. archivos)
29	GS	(sep. grupos)
30	RS	(sep. registros)
31	US	(sep. unidades)
127	DEL	(suprimir)

Caracteres ASCII imprimibles					
32	espacio	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

ASCII extendido (Página de código 437)							
128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

Codificación alfanumérica ASCII

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	
o		c	
r		a	
m			

Código Unicode

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.

Código Unicode

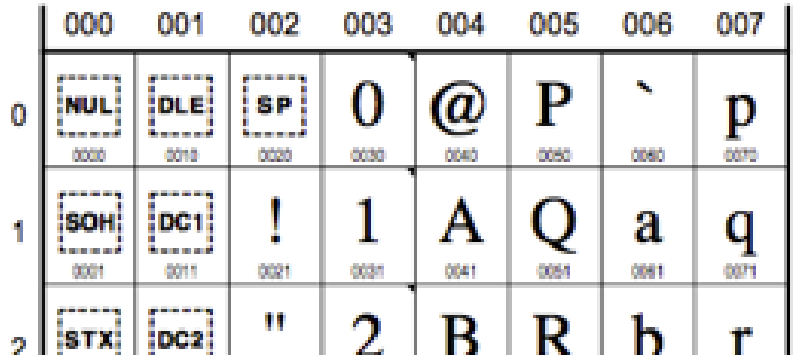
En Unicode existen distintas formas de codificar un mismo carácter según el formato. Existen los siguientes formatos *UTF-8, UTF-16 y UTF-32.*

- **UTF-8** utiliza 1 byte 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 2 bytes 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

Código Unicode

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

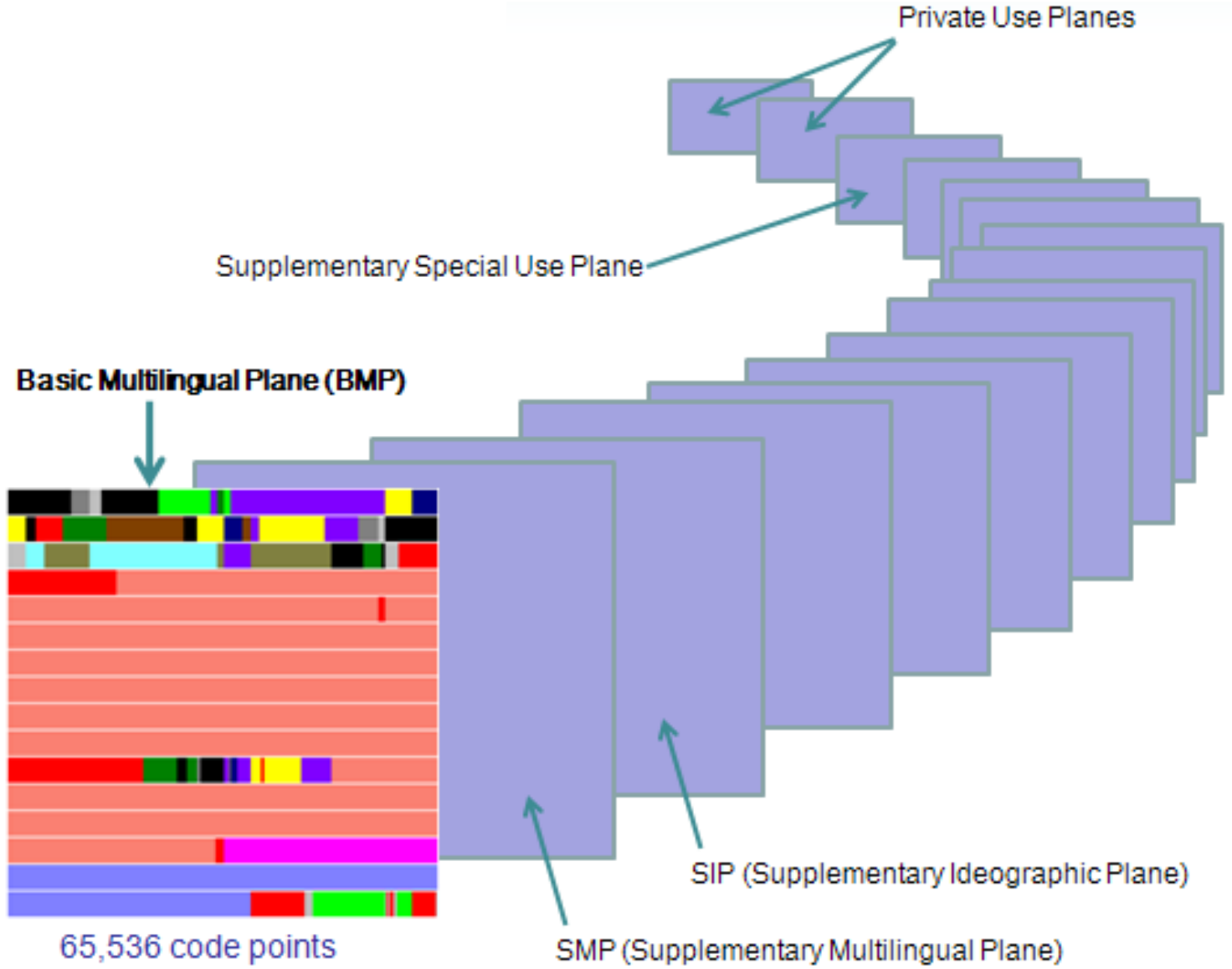


	000	001	002	003	004	005	006	007
0	NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	` 0060	p 0070
1	SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
2	STX	DC2	"	2	B	R	b	r

- 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
  - **SIP** (Supl. Ideographic Plane): Ext. CJK (China, Japón, Korea)
  - **TIP** (Tiertary Ideographic Plane): Lenguas antiguas asiáticas
  - **SSP** (Suplementary Special-purpose Plane): usos especiales

https://unicode-table.com

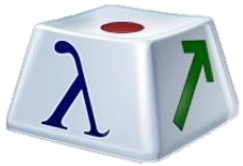
Código Unicode

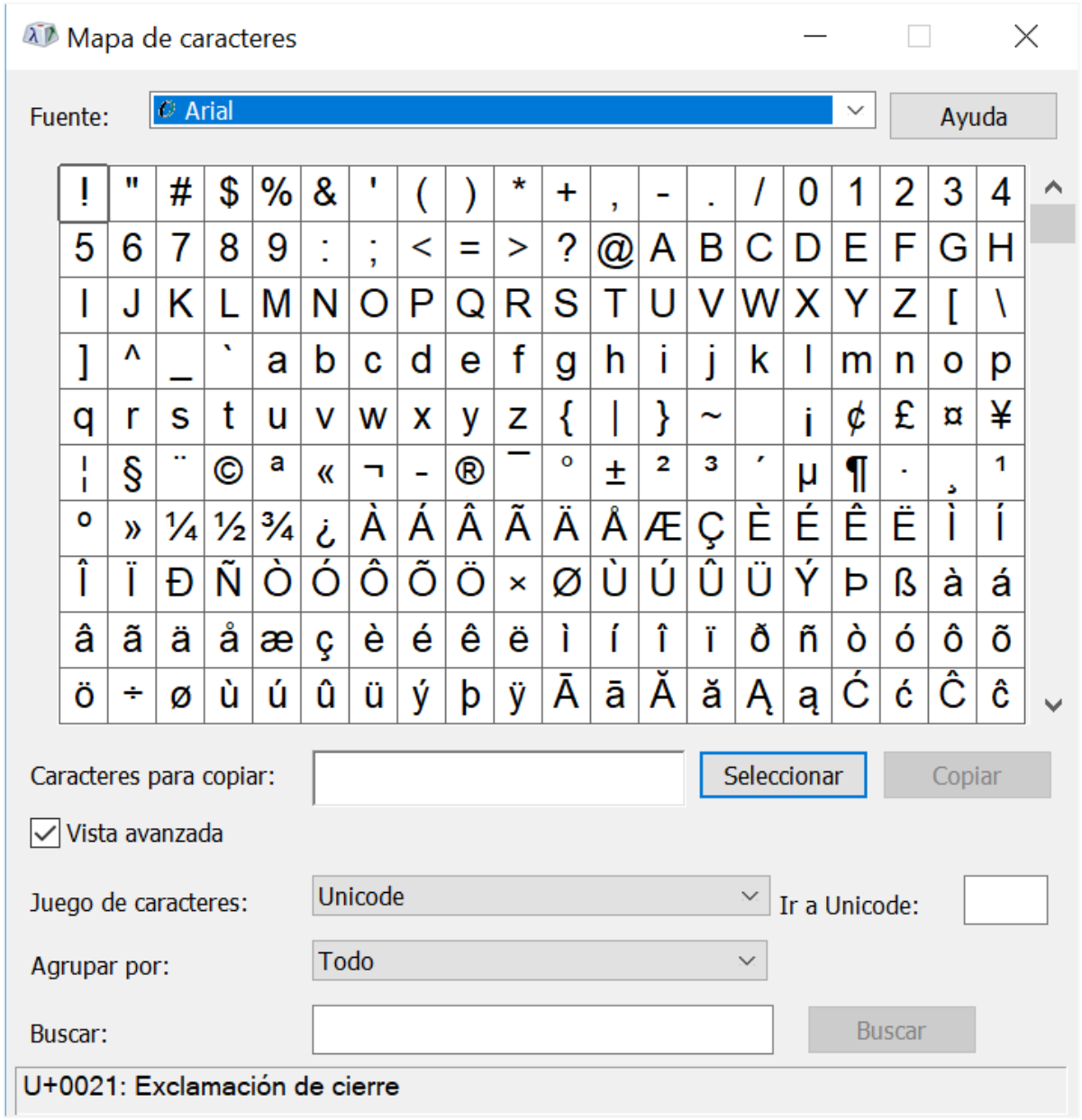


Código Unicode

- 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.

Código Unicode





Código Unicode

Unicode BMP: subconjunto ASCII (basic Latin)

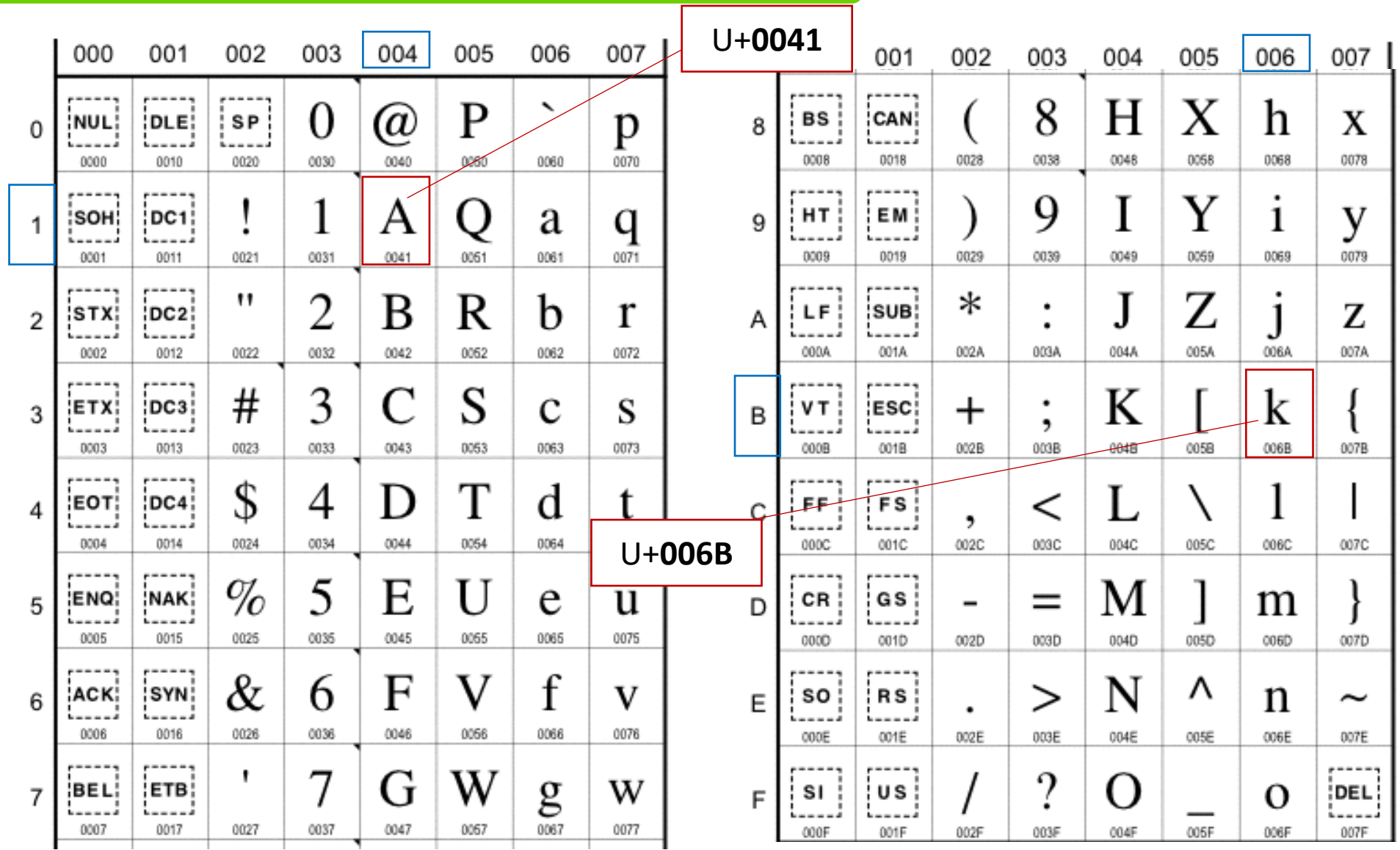
www.unicode.org/charts/PDF/U0000.pdf

	000	001	002	003	004	005	006	007
0	NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	` 0060	p 0070
1	SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
2	STX 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
3	ETX 0003	DC3 0013	# 0023	3 0033	C 0043	S 0053	c 0063	s 0073
4	EOT 0004	DC4 0014	$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074
5	ENQ 0005	NAK 0015	% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
6	ACK 0006	SYN 0016	& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
7	BEL 0007	ETB 0017	' 0027	7 0037	G 0047	W 0057	g 0067	w 0077
8	BS 0008	CAN 0018	(0028	8 0038	H 0048	X 0058	h 0068	x 0078
9	HT 0009	EM 0019	) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
A	LF 000A	SUB 001A	* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
B	VT 000B	ESC 001B	+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B
C	FF 000C	FS 001C	, 002C	< 003C	L 004C	\ 005C	l 006C	| 007C
D	CR 000D	GS 001D	- 002D	= 003D	M 004D	] 005D	m 006D	} 007D
E	SO 000E	RS 001E	. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
F	SI 000F	US 001F	/ 002F	? 003F	O 004F	_ 005F	o 006F	DEL 007F

Código Unicode

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	0 0030	@ 0040	P 0050	` 0060	p 0070
1	SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
2	STX 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
3	ETX 0003	DC3 0013	# 0023	3 0033	C 0043	S 0053	c 0063	s 0073
4	EOT 0004	DC4 0014	$ 0024	4 0034	D 0044	T 0054	d 0064	t
5	ENQ 0005	NAK 0015	% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
6	ACK 0006	SYN 0016	& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
7	BEL 0007	ETB 0017	' 0027	7 0037	G 0047	W 0057	g 0067	w 0077

	001	002	003	004	005	006	007
8	BS 0008	CAN 0018	(0028	8 0038	H 0048	X 0058	h 0068	x 0078
9	HT 0009	EM 0019	) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
A	LF 000A	SUB 001A	* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
B	VT 000B	ESC 001B	+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B
C	FF 000C	FS 001C	, 002C	< 003C	L 004C	\ 005C	l 006C	| 007C
D	CR 000D	GS 001D	- 002D	= 003D	M 004D	] 005D	m 006D	} 007D
E	SO 000E	RS 001E	. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
F	SI 000F	US 001F	/ 002F	? 003F	O 004F	_ 005F	o 006F	DEL 007F

Código Unicode

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

Carácter Unicode	Valor hexadecimal
=	
µ	
T	
æ	
&	
@	
☺	
Φ	
♠	

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

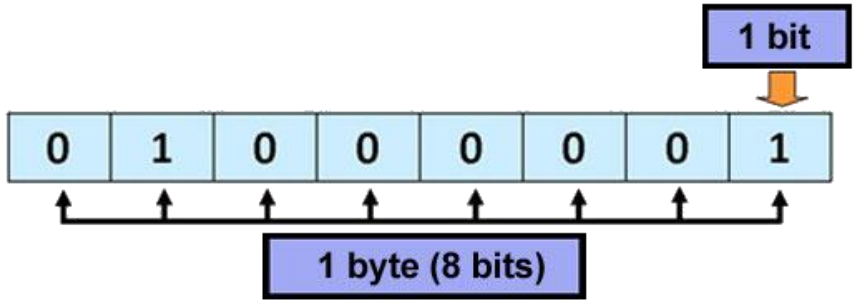
Unidades de medida de información

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	**T**	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

Unidades de medida de información

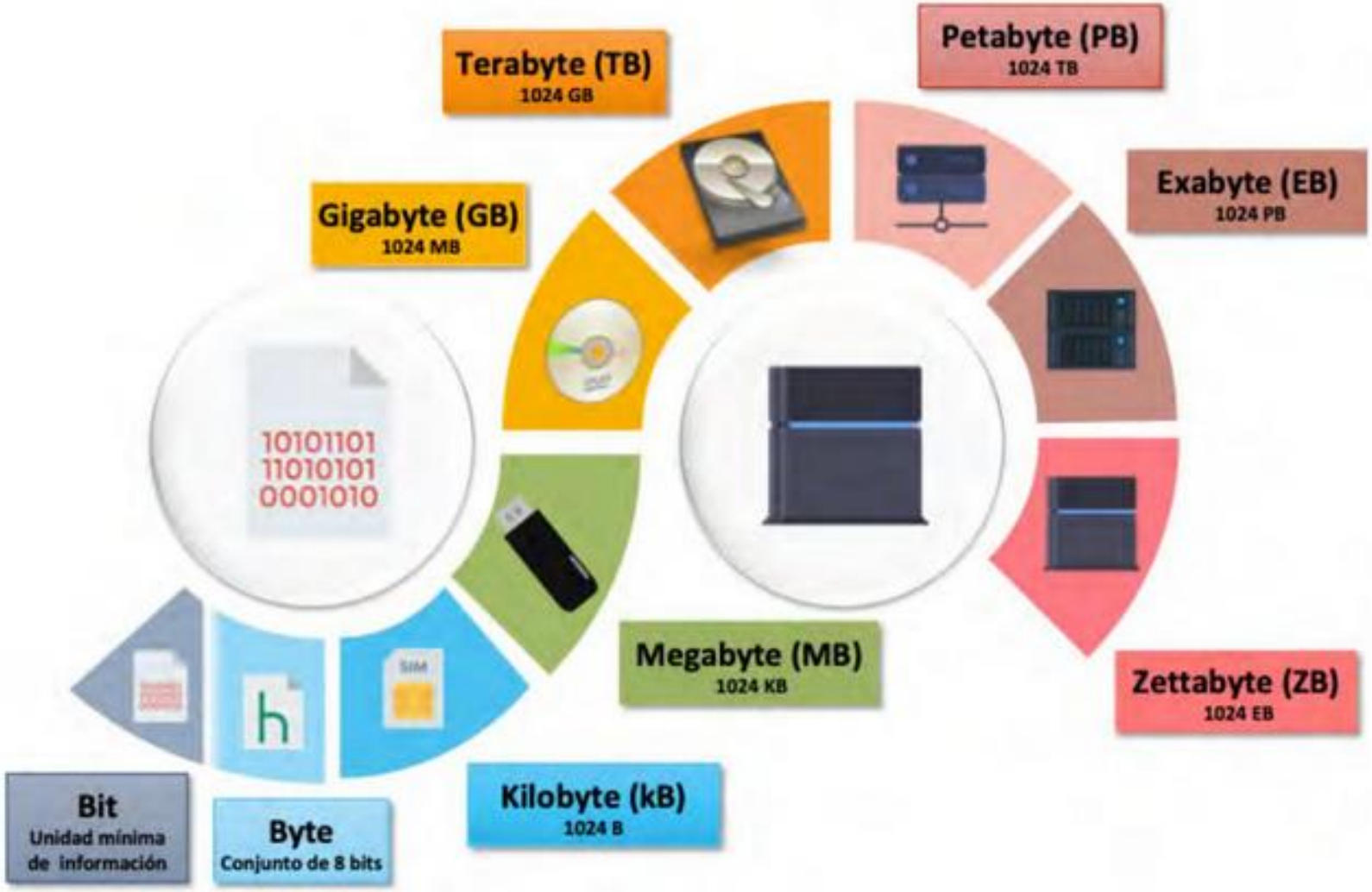
- **1 Bit**: es la unidad básica de medida de almacenamiento de información utilizada en una computadora.
- **1 Byte**: combinación de 8 bits.



- **1 Kbyte (KB)**: 1024 bytes
- **1 Megabyte (MB)**: 1024 Kbytes
- **1 Gigabyte (GB)**: 1024 Mbytes

Unidad	Abreviatura	Se habla de	Representa
1 Kilobyte	KB	kas	1 024 bytes
1 Megabyte	MB	megas	1 024 KB (1 048 576 bytes)
1 Gigabyte	GB	gigas	1 024 MB (1 073 741 824 bytes)
1 Terabyte	TB	teras	1 024 GB (un billón de bytes)

Unidades de medida de información



Unidades de medida de información

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}	1.000.000.000.000

Unidades de medida de información

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 →

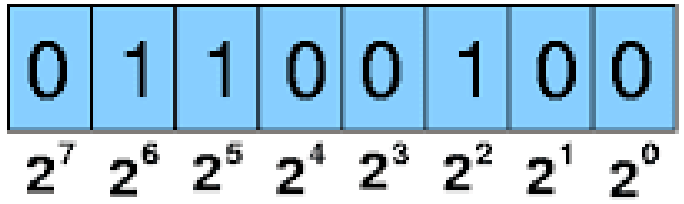
  Base 16: FFC, FFD, FFE →

  Base 16: FFEF →

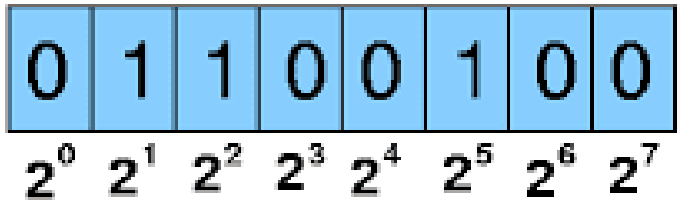
Little endian y Big endian

Little endian y Big endian se refieren al orden 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.



Big Endian
= 0x64 = 100

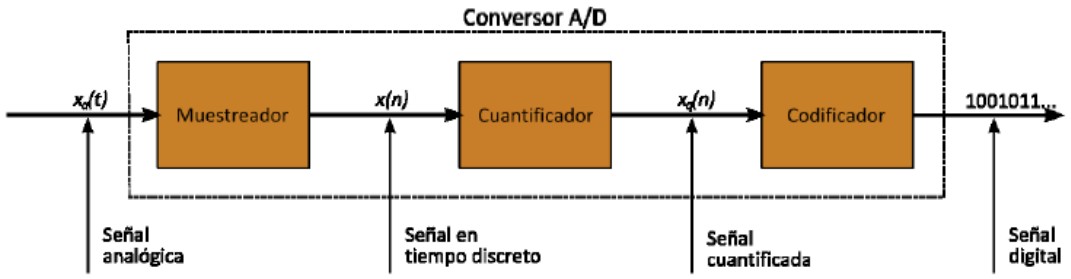


Little Endian
= 0x26 = 38

Conversión analógica-digital

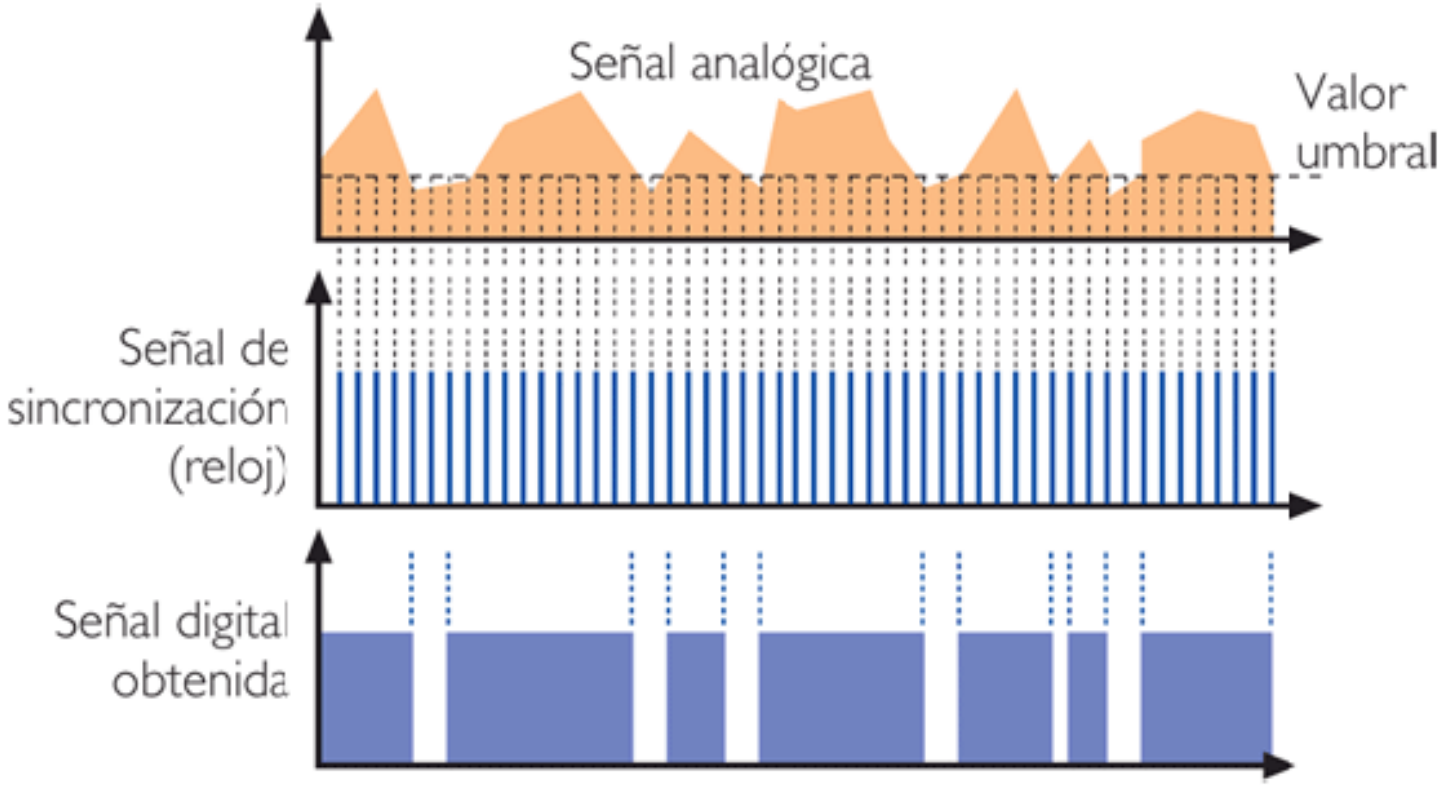
Un ordenador o cualquier sistema de control basado en un microprocesador no puede interpretar señales analógicas, ya que sólo utiliza señales digitales como bien sabemos. Es necesario traducir, o transformar en señales binarias, lo que se denomina proceso de digitalización o conversión de señales analógicas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

La conversión **analógica-digital** (CAD) o **digitalización** 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.



Procesos de la conversión A/D.

Conversión analógica-digital

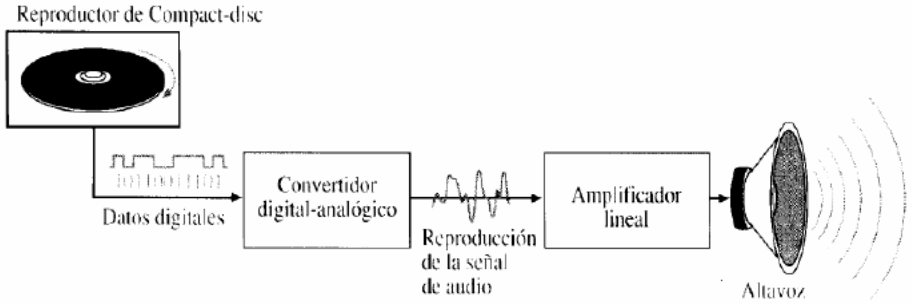


Digitalización por muestreado de una señal analógica

Conversión analógica-digital

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.



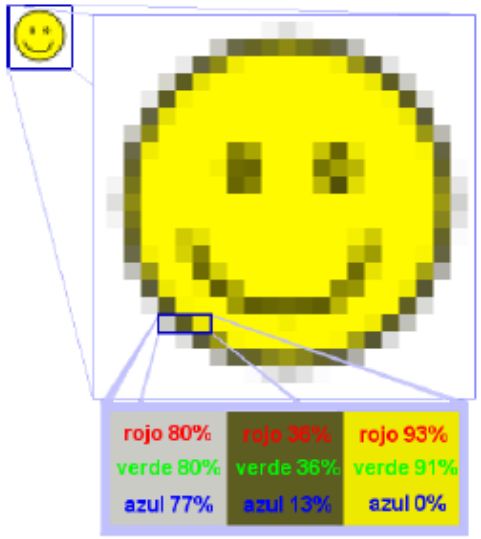
Sistema Digital - Analógico.

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.

Conversión analógica-digital

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.

Conversión analógica-digital

1 bit
(blanco y negro)

2 bits
(4 colores)

3 bits
(8 colores)

5 bits
(32 colores)

24 bits
(16 millones)