SISTEMAS INFORMÁTICOS UD-1

CONCEPTOS BÁSICOS DE LOS SISTEMAS INFORMÁTICOS (III)

LOS SISTEMAS DE NUMERACIÓN

ÍNDICE

[INTRODUCCIÓN 3](#_Toc116318560)

[SISTEMAS DE NUMERACIONES POSICIONALES 3](#_Toc116318561)

[CONVERSIONES ENTRE SISTEMAS BINARIO Y DECIMAL 3](#_Toc116318562)

[1. CONVERSIÓN DE BINARIO A DECIMAL: 4](#_Toc116318563)

[2. CONVERSIÓN DE DECIMAL A BINARIO 5](#_Toc116318564)

[3. CÁLCULOS ARITMÉTICOS BÁSICOS 7](#_Toc116318565)

[SISTEMAS INTERMEDIOS 8](#_Toc116318566)

[A. SISTEMA OCTAL 8](#_Toc116318567)

[1. CONVERSIÓN DE BINARIO A OCTAL 8](#_Toc116318568)

[2. CONVERSIÓN DE OCTAL A BINARIO 8](#_Toc116318569)

[B. SISTEMA HEXADECIMAL 9](#_Toc116318570)

[1. CONVERSIÓN DE BINARIO A HEXADECIMAL 9](#_Toc116318571)

[2. CONVERSIÓN DE HEXADECIMAL A BINARIO 9](#_Toc116318572)

[REPRESENTACIÓN DE NÚMEROS ENTEROS 10](#_Toc116318573)

[A. SIGNO Y MAGNITUD 10](#_Toc116318574)

[B. COMPLEMENTO A 1 (C – 1) 10](#_Toc116318575)

[C. COMPLEMENTO A 2 (C – 2) 11](#_Toc116318576)

[REPRESENTACIÓN DE NÚMEROS REALES 11](#_Toc116318577)

[A. REPRESENTACIÓN EN COMA FIJA 11](#_Toc116318578)

[B. ESTÁNDAR IEEE 754 12](#_Toc116318579)

[REPRESENTACIÓN ALFANUMÉRICA 13](#_Toc116318580)

[A. CODIGO ASCII (American Standard Code for Information Interchange) 14](#_Toc116318581)

[B. CÓDIGOS EBCDIC (Extended Binary Coded Decimal Interchange Code) Y UNICODE 14](#_Toc116318582)

[MEDIDAS DE INFORMACIÓN: CAPACIDAD 15](#_Toc116318583)

[MEDIDAS DE INFORMACIÓN: VELOCIDAD 17](#_Toc116318584)

# INTRODUCCIÓN

Un sistema de numeración es un conjunto de reglas que permiten nombrar y escribir un número partiendo de un número finito de símbolos.

El sistema de numeración más conocido por el humano consta de 10 dígitos (del 0 al 9).

Un sistema de numeración posicional en base b usa una cardinalidad de b y un alfabeto de b símbolos distintos (o cifras).

# SISTEMAS DE NUMERACIONES POSICIONALES

La base 10 (decimal) es la más conocida, pero existen más:

* **Base 2**: Binario. Será de 0 hasta 1.
* **Base 8**: Octal. Será de 0 hasta 7.
* **Base 10**: Decimal. Será de 0 hasta 9.
* **Base 16**: Hexadecimal. Sera de 0 hasta 9 y de la A hasta la F.

La cantidad de números diferentes representados en un sistema de numeración con dígitos será bn (b el número de la base y n el número de dígitos disponibles).

Siendo n=3, tendríamos los siguientes límites:

* **Binario** = 23 = 8 dígitos binarios. Del 0 al 7 decimal.
* **Octal** = 83 = 512 números octales. Del 0 al 511 decimal.
* **Decimal** = 103 = 1000 números decimales. Del 0 al 999 decimal.
* **Hexadecimal** = 163 = 4096 números hexadecimales. Del 0 al 4095 decimal.

Para expresar las bases usamos subíndices de la siguiente forma:

* **Binario** = 101001112.
* **Octal** = 1074358.
* **Decimal** = 10234723810.
* **Hexadecimal** = 1AD564F16.

# CONVERSIONES ENTRE SISTEMAS BINARIO Y DECIMAL

Centrándonos en la base 2, si queremos saber el rango de representación decimal habrá que seguir la fórmula: 0 <= X <= 2n-1, siendo n el número de bits (número de dígitos) a manejar. Por ejemplo, si tenemos 4 bits, tendremos desde el 0 hasta el 15 (24 = 16; 16 – 1 = 15).

1. CONVERSIÓN DE BINARIO A DECIMAL:

Para pasar de binario a decimal se usa el teorema fundamental de la numeración:

N = dn \* bn + … + d1 \* b1 + d0 \* b0 + d-1 \* d-1

d = dígito // b = base

Es decir, el número decimal será la suma de todos los dígitos binarios multiplicados por su base (2) elevada a n siendo n la posición que ocupan en el número siendo el número más a la derecha la posición 0, el número a su izquierda la posición 1… Para decimales se empieza desde la coma hacia la derecha, de forma que el primer dígito post-coma es la posición -1, el siguiente la posición -2…

Por ejemplo, para pasar el 10112 a decimal habría que hacer lo siguiente:

* + - **Paso 1**: Numerar de atrás hacia delante los dígitos empezando por 0, es decir:

|  |  |  |  |
| --- | --- | --- | --- |
| 23 | 22 | 21 | 20 |
| 1 | 0 | 1 | 1 |

* + - **Paso 2**: Multiplicar el dígito por su base correspondiente:

1 \* 23 + 0 \* 22 + 1 \* 21 + 1 \* 20 = 8 + 0 + 2 + 1 = 11

El número 10112 es el 1110.

En el caso de tener decimales habría que considerar como exponentes negativos a los dígitos post-coma. Por ejemplo, el número 111000101,00112 se realizaría así:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 |  | 2-1 | 2-2 | 2-3 | 2-4 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | , | 0 | 0 | 1 | 1 |

1 \* 28 + 1 \* 27 + 1 \* 26 + 0 \* 25 + 0 \* 24 + 0 \* 23 + 1 \* 22 + 0 \* 21 + 1 \* 20 + 0 \* 2-1 +

+ 0 \* 2-2 + 1 \* 2-3 + 1 \* 2-4 = 256 + 128 + 64 + 0 + 0 + 0 + 4 + 0 + 1 + 0 + 0 +

+ 0,125 + 0,0625 = 453,1875.

Es decir, el número 111000101,00112 es igual a 453,187510.

Para agilizar el proceso es mejor memorizar algunos exponentes positivos (además del 0 que será 1):

210 = 1024

29 = 512

28 = 256

27 = 128

26 = 64

25 = 32

24 = 16

23 = 8

22 = 4

21 = 2

Y algunas bases negativas (al menos hasta 2-4):

2-1 = 0,5

2-2 = 0,25

2-3 = 0,125

2-4 = 0,0625

2-5 = 0,03125

Otra forma de hacerlo es a través de una tabla donde apuntemos los valores de las bases según su posición y, a continuación, multiplicar cada valor por el dígito binario correspondiente.

Por ejemplo para pasar el número 1011110111’10012 a base 10 usando la tabla se haría lo siguiente:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **EXPONENTE** | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 |  | 2-1 | 2-2 | 2-3 | 2-4 |
| **N. BINARIO** | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | , | 1 | 0 | 0 | 1 |
| **VALOR EXPONENTE** | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | , | 0,5 | 0,25 | 0,125 | 0,0625 |
| **VALOR REAL\*** | 512 | 0 | 128 | 64 | 32 | 16 | 0 | 4 | 2 | 1 | , | 0,5 | 0 | 0 | 0,0625 |
| **NÚMERO DECIMAL** | 759 | | | | | | | | | | , | 5625 | | | |

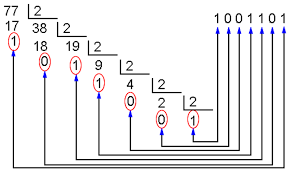
\*Multiplicar el valor del exponente por el dígito binario.

El resultado es 759,562510.

1. CONVERSIÓN DE DECIMAL A BINARIO

Para realizar el paso de decimal a binario hay que realizar lo siguiente en función de si queremos pasar la parte entera del número o la decimal:

* Para la parte entera: se va dividiendo el número entre 2 hasta que no haya posibilidad de dividirlo más (cuando el resultado de una operación de 1). Una vez realizadas todas las operaciones, se cogen los valores del último resultado y todos los restos y se ordenan de abajo hacia arriba. Por ejemplo:



En este caso dividimos el número 77 múltiples veces hasta que llegamos a un resultado 1 y cogemos el último resultado (1) y todos los restos desde el último hasta el primero (001101) por lo que el 7710 es el 10011012.

* Para la parte decimal. En el caso de la parte decimal hay que hacer el paso contrario, multiplicar el número por 2. Cada vez que realicemos una multiplicación guardamos el digito entero para el número binario y luego multiplicamos la parte decimal resultante otra vez por 2. En caso de que alguna multiplicación dé como resultado un número superior a 1 (por ejemplo, 1.5) se guardará el 1 para el binario y se multiplicará la parte decimal seguiría multiplicándose (en ese caso el 1 se guardaría para el binario y el 0.2 se multiplicándose).

Para recoger el valor binario tomamos desde el primer resultado de la multiplicación hasta el último.

En caso de decimales binarios largos se puede aproximar a 4-5 cifras, aunque por lo general se llegará a 2 situaciones:

* + - Situación periódica. Ciclo continuo de repetición de un bucle de números. Por ejemplo, como pasa con la parte decimal .1 cuya forma binaria es: .00011001100110011. Es decir, el bloque 0011 se repite de forma continua.
    - Situación de 0. Sucede cuando el resultado final tras multiplicación es 1.0. En este caso ese 1 será el último número binario ya que no tendría sentido seguir multiplicando 0 x 2 ya que el resultado siempre sería 0.

Texto

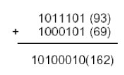
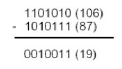
Descripción generada automáticamente con confianza mediaPor ejemplo:

En este caso se multiplica el 0.1875 x 2 y obtenemos un 0.375 por lo que guardamos el primer dígito (0) para el binario y volvemos a multiplicar por 2, obtenemos un 0.75 así que guardamos el primer dígito (0) y volvemos a multiplicar, como el resultado de multiplicar 0.75 nos da un número mayor a 1 (1.5) guardamos la parte entera (1) para el binario y seguimos multiplicando por 2 la parte decimal. Por último, como el resultado de multiplicar 0.5 x 2 es 1, terminamos la obtención del decimal binario con ese número.

Por tanto, el 0.187510 es el 0.00112.

1. CÁLCULOS ARITMÉTICOS BÁSICOS

Con los números binarios se pueden realizar operaciones aritméticas básicas siguiendo una serie de reglas:

* + **Suma**: Se parte de las siguientes reglas:
    - * 1 + 1 = 0 (y llevo 1).
      * 1 + 0 = 1
      * 0 + 1 = 1
      * 0 + 0 = 0
  + **Resta**: Se parte de las siguientes reglas:
    - * 1 – 1 = 0
      * 1 – 0 = 1
      * 0 – 1 = 1 (y debo 1)
      * 0 – 0 = 0
  + **Multiplicación**: Se parte de las siguientes reglas:
    - * 1 \* 1 = 1
      * 1 \* 0 = 0
      * 0 \* 1 = 0
      * 0 \* 0 = 0
  + **División**: Se parte de las siguientes reglas:
    - * 1 / 1 = 1
      * 1 / 0 = Infinito
      * 0 / 1 = 0
      * 0 / 0 = Indeterminado

# SISTEMAS INTERMEDIOS

Trabajar en binario es muy laborioso y puede llevar a errores, estos sistemas facilitan la labor de programación.

Existen varios tipos, pero para la representación numérica se usan el:

* Octal.
* Hexadecimal.

## SISTEMA OCTAL

También conocida como base 8 es un sistema en el que se tienen un total de 8 símbolos para expresar los números. Todos los símbolos son dígitos que cubren los valores del 0 al 7.

1. CONVERSIÓN DE BINARIO A OCTAL

Para pasar de binario a octal, hay que dividir el número binario en bloques de 3 dígitos.

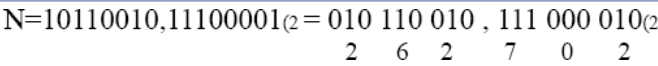
Esta descomposición en bloques se hace de derecha a izquierda para la parte entera del número binario y de izquierda a derecha para la parte decimal.

En caso de no haber dígitos suficientes para completar un bloque se añaden 0 (a la izquierda del todo en el caso de la parte entera, a la derecha del todo en el caso de la parte decimal).

Una vez dividido el número en esos bloques pasamos cada bloque a octal siguiendo la siguiente conversión:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **BINARIO** | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| **OCTAL** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Por ejemplo:



1. CONVERSIÓN DE OCTAL A BINARIO

Para pasar un dígito de octal a binario se hace algo similar a lo anterior, en este caso se coge cada dígito octal y se pasa a bloques de 3 dígitos binarios.

Por ejemplo:



## SISTEMA HEXADECIMAL

También conocido como base 16. Es un sistema que el que se tienen 16 símbolos para representar los números. De esos 16 símbolos, 10 son dígitos (del 0 al 9) y 6 son letras (de la A hasta la F).

1. CONVERSIÓN DE BINARIO A HEXADECIMAL

Para pasar un digito de binario a hexadecimal, hay que dividir el número binario en bloques de 4 dígitos.

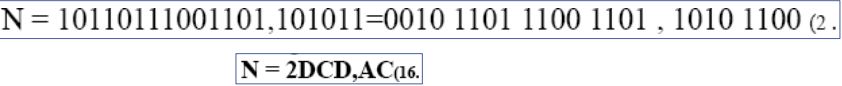
Esta descomposición en bloques se hace de derecha a izquierda para la parte entera del número binario y de izquierda a derecha para la parte decimal.

En caso de no haber dígitos suficientes para completar un bloque se añaden 0 (a la izquierda del todo en el caso de la parte entera, a la derecha del todo en el caso de la parte decimal).

Una vez dividido el número en esos bloques pasamos cada bloque a hexadecimal siguiendo la siguiente conversión:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BINARIO** | **HEXADECIMAL** |  | **BINARIO** | **HEXADECIMAL** |
| 0000 | 0 |  | 1000 | 8 |
| 0001 | 1 |  | 1001 | 9 |
| 0010 | 2 |  | 1010 | A |
| 0011 | 3 |  | 1011 | B |
| 0100 | 4 |  | 1100 | C |
| 0101 | 5 |  | 1101 | D |
| 0110 | 6 |  | 1110 | E |
| 0111 | 7 |  | 1111 | F |

Por ejemplo:



1. CONVERSIÓN DE HEXADECIMAL A BINARIO

Para pasar un dígito de hexadecimal a binario se hace algo similar a lo anterior, en este caso se coge cada dígito o letra hexadecimal y se pasa a bloques de 4 dígitos binarios.

Por ejemplo:



# REPRESENTACIÓN DE NÚMEROS ENTEROS

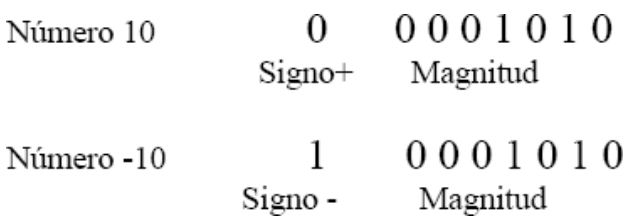
Los ordenadores utilizan varios métodos para la representación interna de los números (positivos y negativos).

La ventaja de estas representaciones es que las restas se realizarán como sumas y la UAL no tendrá que incorporar un restador.

## SIGNO Y MAGNITUD

En este sistema el primer bit (el que se encuentra más a la izquierda) representa el signo del número entero siendo 0 para los valores positivos y 1 para los valores negativos.

El resto de bits (n-1) representan la magnitud del número, de forma que, por ejemplo, para 7 bits de magnitud podremos representar 128 valores positivos (del 0+ al 127) y 128 valores negativos (del 0- al -127) como el 0+ y el 0- representan el mismo valor (0), de los 256 valores totales sólo se pueden representar 255.

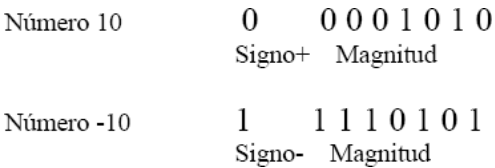


## COMPLEMENTO A 1 (C – 1)

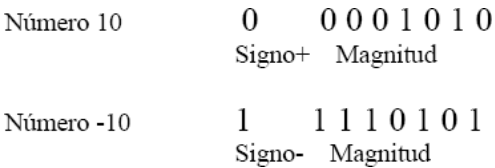
Al igual que en signo y magnitud, el valor más a la izquierda indica el signo siguiendo la misma normativa.

El resto de bits (n-1):

* En los números positivos: representan la magnitud.



* En los números negativos: los dígitos binarios son complementados (es decir, cambian su valor: los 0 pasan a ser 1 y los 1 pasan a ser 0).

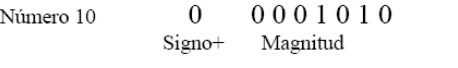


## COMPLEMENTO A 2 (C – 2)

Se mantiene la norma del bit a la izquierda para indicar el signo del número.

El resto de bits (n-1):

* En los números positivos: al igual que en el complemento a 1 representan la magnitud.

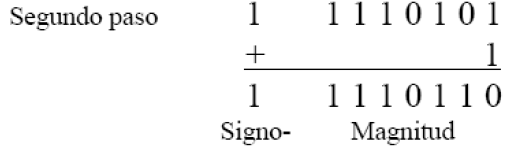


* En los números negativos: se realizan dos pasos primero una complementación y luego a ese número complementado se le suma 1.



* + - Paso 1: Se complementan los dígitos, es decir, pasamos los 1 a 0 y los 0 a 1.



* + - Paso 2: Al número complementado, se le suma 1 despreciando el último acarreo en caso de existir.

# REPRESENTACIÓN DE NÚMEROS REALES

En el caso de querer representar magnitudes con parte fraccionaria (decimal) se necesita una forma de representar la posición que ocupa la coma fraccionaria.

Dicha coma puede mantenerse fija (coma fija) o desplazarse (coma flotante).

## REPRESENTACIÓN EN COMA FIJA

Es un sistema obsoleto porque el programador debe vigilar el tratamiento de los datos, de forma que especifica el lugar de la coma.

Por ejemplo, tomando un número de 8 bits (por ejemplo, el 10101,110) podemos decidir darle 5 dígitos para la parte entera (10101) y 3 para la parte fraccionara (110).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **PARTE ENTERA** | | | | | **COMA** | **PARTE DECIMAL** | | |
| **BINARIO** | 1 | 0 | 1 | 0 | 1 | , | 1 | 1 | 0 |
| **EXPONENTES** | 16 | 8 | 4 | 2 | 1 | , | 0’5 | 0’25 | 0’125 |
| **DECIMAL** | 21 | | | | | , | 75 | | |

La coma está fija y sirve para separar la parte entera de la parte fraccionaria.

Al usar coma fija, queda muy limitado el número de cantidades a representar.

## ESTÁNDAR IEEE 754

Es el método que se utiliza actualmente para la notación de números reales. Se siguen una serie de pasos. Son números de 32 bits formados por: 1 bit de signo, 8 bits de exponente y 23 bits de mantisa.

Por ejemplo, para pasar el número decimal -118,625 a binario usando este estándar:

1. Al ser un número negativo, el dígito del signo será el 1.
2. Se pasa el número a binario:

118 / 2 = 59 (0)

59 / 2 = 29 (1)

29 / 2 = 14 (1)

14 / 2 = 7 (0)

7 / 2 = 3 (1)

3 / 2 = 1 (1)

Parte entera: 1110110

0,625 \* 2 = 1,250 (1)

0,250 \* 2 = 0,5 (0)

0,5 \* 2 = 1,0 (1)

Parte decimal: 101

Número binario: 1110110,101

1. Debido a que los números siempre van a empezar por un 1, este se separa del resto con coma, es decir, movemos la coma hasta después del primer 1:

1,110110101

Y lo multiplicamos por 2n (siendo n el número de espacios que se ha movido la coma, en este caso 6):

1,110110101 \* 26

Este sería el número en coma flotante normalizado.

1. Rellenamos el valor después de la coma con tantos 0 como sean necesario para completar la mantisa (23 bits):

11011010100000000000000

1. Para calcular el exponente, hay que sumar 127 + n (en este caso 127 + 6 = 133) y pasarlo a binario.

133 / 2 = 66 (1)

66 / 2 = 33 (0)

33 / 2 = 16 (1)

16 / 2 = 8 (0)

8 / 2 = 4 (0)

4 / 2 = 2 (0)

2 / 2 = 1 (0)

Número binario: 10000101

1. Por último, hay que juntar todo:

|  |  |  |
| --- | --- | --- |
| SIGNO | EXPONENTE | MANTISA |
| 1 | 10000101 | 11011010100000000000000 |
| 1 bit | 8 bits | 23 bits |

# REPRESENTACIÓN ALFANUMÉRICA

Para representar los 10 dígitos y las 26 letras minúsculas necesitamos como mínimo 6 bits. Esto es debido a que necesitamos 36 combinaciones de 1 y 0 por lo que necesitamos 26 combinaciones (64 espacios para combinación) ya que 25 sólo nos daría 32 espacios para combinación.

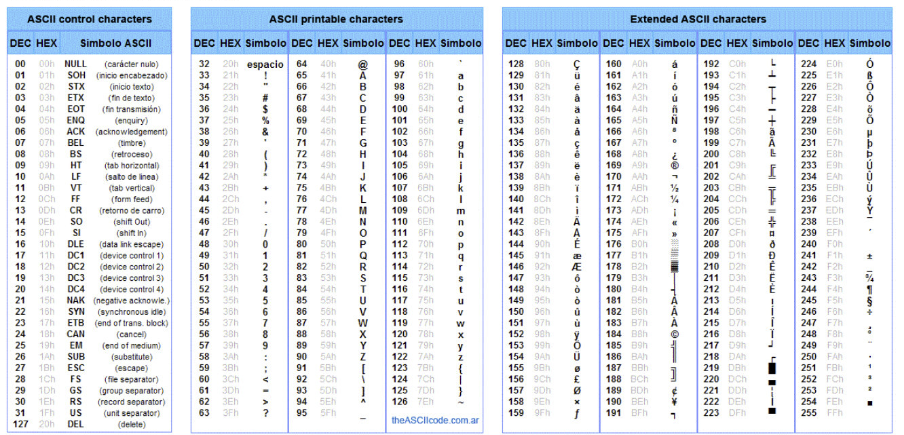
Si además se quieren representar letras mayúsculas y otros símbolos de utilidad se necesitarán más bits.

Los códigos alfanuméricos incluyen:

* Las letras. A-Z (mayúsculas) y a-z (minúsculas).
* Los números. Del 0 al 9.
* Los símbolos. Por ejemplo @, !, #, $, +, -…
* Los caracteres de control. <CR>, <LF>, etc.

## CODIGO ASCII (*American Standard Code for Information Interchange*)

Originalmente utilizaba 7 bits para representar los caracteres.

El estándar ISO-8859-1 es una extensión que utiliza 8 bits para proporcionar caracteres adicionales usados en idiomas distintos al inglés, como el español.

## CÓDIGOS EBCDIC (*Extended Binary Coded Decimal Interchange Code*) Y UNICODE

El EBCDIC es un código de 8 bits usados por PCs de IBM.

El Unicode es un código de 16 bits que permite codificar todos los caracteres de todos los lenguajes sin necesidad de emplear secuencias de escape o códigos de control.

# MEDIDAS DE INFORMACIÓN: CAPACIDAD

La medida de información básica es el bit y un byte (B) equivale a 8 bits.

Podemos usar 2 abreviaturas:

* Prefijos que representan múltiplos de 1000 (prefijo del sistema internacional) 🡪 KB, MB…

|  |  |  |  |
| --- | --- | --- | --- |
| NOMBRE | ABREVIATURA | DESCRIPCIÓN | BYTES |
| bit | B | Unidad mínima | - |
| Byte | B | 8b | - |
| Kilobyte | kB | 1000 B | 103 |
| Megabyte | MB | 1000 kB | 106 |
| Gigabyte | GB | 1000 MB | 109 |
| Terabyte | TB | 1000 GB | 1012 |
| Petabyte | PB | 1000 TB | 1015 |
| Exabyte | EB | 1000 PB | 1018 |

Si se pasa de una medida mayor a una menor (por ejemplo, de Terabytes a Gigabytes) habrá que multiplicar por 1000 salvo en el caso de la unidad a bits en cuyo caso se multiplicará por 8. Por ejemplo, para pasar 1’2 GB a bits habría que:

* + - Paso 1: Pasar de GB a MB

1’2 GB = 1’2 \* 103 = 1.200 MB

* + - Paso 2: Pasar de MB a kB

1.200 MB = 1.200 \* 103 = 1.200.000 kB

* + - Paso 3: Pasar de kB a B

1.200.000 kB = 1.200.000 \* 103 = 1.200.000.000 B

* + - Paso 4: Pasar de B a bits

1.200.000.000B = 1.200.000.000 \* 8 = 9.600.000.000 bits = 9’6 \* 109 bits

En el caso contrario, si pasamos de una medida menor a una mayor (por ejemplo, de Bytes a Kylobytes) habrá que dividir por 1000 salvo el caso de la unidad bits que se dividirá por 8. Por ejemplo, para pasar de 1.750.000 bits a GB habría que:

* + - Paso 1: Pasar de bits a B:

1.750.000 bits = 1.750.000 / 8 = 218.750 B

* + - Paso 2: Pasar de B a KB:

218.750 B = 218.750 KB / 1000 = 218’75 KB

* + - Paso 3: Pasar de KB a MB:

218’75 KB = 218’75 KB / 1000 = 0’21875 MB

* + - Paso 4: Pasar de MB a GB:

0’21875 MB = 0’21875 / 1000 = 0’00021875 GB = 2’1875 \* 10-4

* Prefijos que representan múltiplos de 1024 (prefijos binarios) 🡪 KiB, MiB…

|  |  |  |  |
| --- | --- | --- | --- |
| NOMBRE | ABREVIATURA | DESCRIPCIÓN | BYTES |
| bit | B | Unidad mínima | - |
| Byte | B | 8b | - |
| Kibibyte | KiB | 1024 B | 210 |
| Mebibyte | MiB | 1024 KiB | 220 |
| Gibibyte | GiB | 1024 MiB | 230 |
| Tebibyte | TiB | 1024 GiB | 240 |
| Pebibyte | PiB | 1024 TiB | 250 |
| Exbibyte | EiB | 1024 PiB | 260 |

Si se pasa de una medida mayor a una menor (por ejemplo, de Tebibytes a Gibibytes) habrá que multiplicar por 1024 salvo en el caso de la unidad a bits en cuyo caso se multiplicará por 8. Por ejemplo, para pasar 1’2 GiB a bits habría que:

* + - **Paso 1**: Pasar de GiB a MiB

1’2 GiB = 1’2 \* 1024 = 1.228’8 MiB

* + - **Paso 2**: Pasar de MiB a KiB:

1.228’8 MiB = 1.228’8 \* 1024 = 1.258.291,2 KiB

* + - **Paso 3**: Pasar de KiB a B

1.258.291,2 KiB = 1.258.291,2 \* 103 = 1.288.490.188,8 B

* + - **Paso 4**: Pasar de B a bits

1.288.490.188,8 B = 1.288.490.188,8 \* 8 = 10.307.921.510,4 bits

En el caso contrario, si pasamos de una medida menor a una mayor (por ejemplo, de Gibibytes a Mebibytes) habrá que dividir por 1024 salvo el caso de la unidad bits que se dividirá por 8. Por ejemplo, para pasar de 1.750.000 bits a GiB habría que:

* + - **Paso 1**: Pasar de bits a B:

1.750.000 bits = 1.750.000 / 8 = 218.750 B

* + - **Paso 2**: Pasar de B a KB:

218.750 B = 218.750 KB / 1024 = 213,62 KB

* + - **Paso 3**: Pasar de KB a MB:

213,62 KB = 213,62 KB / 1024 = 0,20 MB

* + - **Paso 4**: Pasar de MB a GB:

0,20 MB = 0,20 / 1024 = 0,00020 GB = 2 \* 10-4

En el inicio de la informática se utiliza el prefijo del sistema internacional para designar a las potencias de 2, es decir, se entendía que 1 kB eran 1024 B, actualmente Windows sigue este mismo sistema, pero otros sistemas decidieron dividir los sistemas de medición en 2.

Por otra parte, el total de caracteres que se pueden representar con N dígitos binarios será 2n, por ejemplo, con un 1 dígito binario podemos representar 21 números (el 0 y el 1) con 10 dígitos binarios podemos representar 210 = 1024.

A su vez la cifra máxima que se podrá representar será 2n – 1, por ejemplo, con 5 bits podremos representar 25 números, es decir, 32 números: desde el 0 hasta el 31.

# MEDIDAS DE INFORMACIÓN: VELOCIDAD

La información no sólo se almacena, también se transmite. Por ello, también es interesante saber cuánta información es capaz de ser envidada o recibida por unidad de tiempo.

Se usa el bit/segundo como unidad base y siempre se usan múltiplos de 1000.

|  |  |  |
| --- | --- | --- |
| NOMBRE | ABREVIATURA | DESCRIPCIÓN |
| bit por segundo | b/s | Unidad base |
| Kilobit por segundo | Kb/s | 1000 b/s |
| Megabit por segundo | Mb/s | 1000 Kb/s |
| Gigabit por segundo | Gb/s | 1000 Mb/s |
| Terabit por segundo | Tb/s | 1000 Gb/s |
| Petabit por segundo | Pb/s | 1000 Tb/s |
| Exabit por segundo | Eb/s | 1000 Pb/s |

También existen unidades de velocidad en bytes (KB/s, MB/s…) y en binario (KiB/s, MiB/s…)

Para tener una idea del orden de magnitud habría que usar la siguiente tabla:

|  |  |
| --- | --- |
| TIPO DE TRANSMISIÓN | TASA DE TRANSFERENCIA |
| A través del módem | De 10 Kb/s a 56 Kb/s |
| Con ADSL | De 10 Mb/s a 20 Mb/s |
| Por fibra | 100 Mb/s |
| Por USB1 | 12 Mb/s |
| Por USB2 | 480 Mb/s |
| Por USB3 | 10 Gb/s |
| Por ATA | 100 Mb/s |
| Por SATA III | 10 Gb/s |
| Por PCI-E v3 | 16 Gb/s |
| Por 3G | 10 Mb/s |
| Por 4G | 300 Mb/s |
| Por 5G | 7 Gb/s |

De manera que, si queremos calcular el tiempo que tardaríamos en, por ejemplo, descargar un archivo de vídeo de 1’2GB usando fibra para ello habría que:

* **Paso 1**: Pasar los GB a MB

1’2 GB = 1’2 \* 103 = 1200 MB

* **Paso 2**: Pasar los MB a Mb

1200 MB = 1200 \* 8 = 9600 Mb

* **Paso 3**: Calcular el tiempo dividiendo los Mb entre la velocidad de fibra (100Mb/s):

9600 Mb / 100 Mb/s = 96 segundos = 1 min 36 segundos