

# ANEXO 1: FORMATO DE LOS DATOS.

## A.1.1. NUMEROS BINARIOS, OCTALES Y HEXADECIMALES.

El sistema de numeración utilizado habitualmente es la base 10; es decir, consta de 10 dígitos (0-9) que podemos colocar en grupos, ordenados de izquierda a derecha y de mayor a menor.

Cada posición tiene un valor o peso de  $10^n$  donde  $n$  representa el lugar contado por la derecha:

$$1357 = 1 \times 10^3 + 3 \times 10^2 + 5 \times 10^1 + 7 \times 10^0$$

Explícitamente, se indica la base de numeración como  $1357_{10}$ .

En un ordenador el sistema de numeración es *binario* -en base 2, utilizando el 0 y el 1- hecho propiciado por ser precisamente dos los estados estables en los dispositivos digitales que componen una computadora.

Análogamente a la base 10, cada posición tiene un valor de  $2^n$  donde  $n$  es la posición contando desde la derecha y empezando por 0:

$$101_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

Además, por su importancia y utilidad, es necesario conocer otros sistemas de numeración como pueden ser el *octal* (base 8) y el *hexadecimal* (base 16). En este último tenemos, además de los números del 0 al 9, letras -normalmente en mayúsculas- de la A a la F.

Llegar a un número en estos sistemas desde base 2 es realmente sencillo si agrupamos las cifras binarias de 3 en 3 (octal) o de 4 en 4 (hexadecimal):

$$\text{Base 2 a base 8: } 101\ 011_2 = 53_8$$

Base 2 a base 16:  $0010\ 1011_2 = 2B_{16}$

A la inversa, basta convertir cada dígito octal o hexadecimal en binario:

Base 8 a base 2:  $24_8 = 010\ 100_2$

Base 16 a base 2:  $24_{16} = 0010\ 0100_2$

De ahora en adelante, se utilizarán una serie de sufijos para determinar el sistema de numeración empleado:

Sufijo	Base	Ejemplos
b	2	01101010b
o,q	8	175o
d	10	789d
h	16	6A5h

**Tabla A.1.**

En caso de que no aparezca el sufijo, el número se considera decimal; es decir, en base 10.

## **A.1.2. - CAMBIO DE BASE.**

Pese a que las conversiones entre base 2 y base 8 y 16 son prácticamente directas, existe un sistema general para realizar el cambio de una base a otra. El paso de cualquier base a base 10 lo vimos antes:

$$6A5h = 6 \times 16^2 + 10 \times 16^1 + 5 \times 16^0$$

Inversamente, si queremos pasar de base 10 a cualquier otra habrá que realizar sucesivas divisiones por la base y tomar los restos:

$$\begin{array}{r}
 1234 \quad | \quad 16 \\
 114 \quad | \quad 77 \quad | \quad 16 \\
 \underline{2} \quad \quad \underline{13} \quad \underline{4}
 \end{array}
 \qquad 1234d = 4D2h$$

**Figura A.1.**

donde 4 es el último cociente (menor que la base) y los restantes dígitos son los restos en orden inverso.

## **A.1.3. - ESTRUCTURA ELEMENTAL DE LA MEMORIA.**

### ***1.3.1. - BIT.***

Toda la memoria del ordenador se compone de dispositivos electrónicos que pueden adoptar únicamente dos estados, que representamos matemáticamente por 0 y 1. Cualquiera de estas unidades de información se denomina *BIT*, contracción de «binary digit» en inglés.

### ***1.3.2. - BYTE.***

Cada grupo de 8 bits se conoce como *byte* u octeto. Es la unidad de almacenamiento en memoria, la cual está constituida por un elevado número de posiciones que almacenan bytes. La cantidad de memoria de que dispone un sistema se mide en Kilobytes (1 Kb = 1024 bytes), en Megabytes (1 Mb = 1024 Kb), Gigabytes (1 Gb = 1024 Mb), Terabytes (1 Tb = 1024 Gb) o Petabytes (1 Pb = 1024 Tb).

Los bits en un byte se numeran de derecha a izquierda y de 0 a 7, correspondiendo con los exponentes de las potencias de 2 que reflejan el valor de cada posición. Un byte nos permite, por tanto, representar 256 estados (de 0 a 255) según la combinación de bits que tomemos.

### ***1.3.3. - NIBBLE.***

Cada grupo de cuatro bits de un byte constituye un *nibble*, de forma que los dos nibbles de un byte se llaman nibble superior (el compuesto por los bits 4 a 7) e inferior (el compuesto por los bits 0 a 3). El nibble tiene gran utilidad debido a que cada uno almacena un dígito hexadecimal:

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

**Tabla A.2.**

## A.1.4. OPERACIONES ARITMÉTICAS SENCILLAS EN BINARIO.

Para sumar números, tanto en base 2 como hexadecimal, se sigue el mismo proceso que en base 10:

Podemos observar que la suma se desarrolla de la forma tradicional; es decir: sumamos normalmente, salvo en el caso de  $1 + 1 = 10_2$ , en cuyo caso tenemos un acarreo de 1 (lo que *nos llevamos*).

$$\begin{array}{r} 1010 \ 1010b \\ + 0011 \ 1100b \\ \hline 1110 \ 0110b \end{array}$$

## A.1.5. - COMPLEMENTO A DOS.

En general, se define como valor negativo de un número el que necesitamos sumarlo para obtener 00h, por ejemplo:

$$\begin{array}{r} FFh \\ + 01h \\ \hline \underline{100h} \end{array}$$

Como en un byte solo tenemos dos nibbles, es decir, dos dígitos hexadecimales, el resultado es 0 (observar cómo el 1 más significativo subrayado es ignorado). Luego  $FFh = -1$ . Normalmente, el bit 7 se considera como de signo y, si está activo (a 1) el número es negativo.

Por esta razón, el número 80h, cuyo complemento a dos es él mismo, se considera negativo (-128) y el número 00h, positivo. En general, para hallar el complemento a dos de un número cualquiera basta con calcular primero su **complemento a uno**, que consiste en cambiar los unos por ceros y los ceros por unos en su notación binaria; a continuación se le suma una unidad para calcular el **complemento a dos**. Con una calculadora, la operación es más sencilla: el complemento a dos de un número **A** de **n** bits es  $2^n - A$ .

Otro factor a considerar es cuando se pasa de operar con un número de cierto tamaño (ej., 8 bits) a otro mayor (pongamos de 16 bits). Si el número es positivo, la parte que se añade *por la izquierda* son bits a 0. Sin embargo, si era negativo (bit más significativo activo) la parte que se añade por la izquierda son bits a 1. Este fenómeno, en cuya demostración matemática no entraremos, se puede resumir en que el bit más significativo se copia en todos los añadidos: es lo que se denomina la **extensión del signo**: los dos siguientes números son realmente el mismo número (el  $-3_{10}$ ):  $1101_2$  (4 bits) y  $11111101_2$  (8 bits).