C.F.G.S. DESARROLLO DE APLICACIONES INFORMÁTICAS

MÓDULO:

Sistemas Informáticos Multiusuario y en Red

Unidad 1

Sistemas Informáticos

INDICE DE CONTENIDOS

1. INTRODUCCIÓN

Un **ordenador** es un dispositivo que acepta datos en una determinada forma, los procesa y produce otros datos o información de una forma diferente a la original, las formas en que el ordenador acepta los datos o produce la información puede variar de un instante a otro, por ello cuando el ordenador procesa datos está realizando una serie de funciones distintas:

DATOS → PROCESO → INFORMACIÓN

Las **funciones** básicas de un ordenador son cuatro:

- **Entrada de datos**: los datos que provienen del exterior, procedentes de alguna fuente de información, son introducidos para ser procesados.
- **Almacenamiento**: el ordenador almacena o conserva internamente los datos en forma codificada; antes, durante y después del proceso.
- **Proceso:** el ordenador realiza operaciones con los datos que tiene almacenados en la memoria donde guardará también los resultados codificados hasta el siguiente paso.
- 4. **Salida:** el ordenador produce nuevos datos descodificados, o información para uso externo.

ENTRADA DE DATOS → ALMACENAMIENTO Y PROCESO → SALIDA DE DATOS

2. EL SISTEMA INFORMÁTICO: SOFTWARE Y HARDWARE

El ordenador se puede definir como una máquina compuesta de elementos físicos, en su mayoría de origen electrónico capaz de realizar una gran variedad de trabajos a gran velocidad y con gran precisión. El conjunto de órdenes o instrucciones que se introducen en un ordenador para realizar un proceso determinado se denomina **programa**. El conjunto de varios programas se denomina **aplicación informática.**

EJEMPLO 1

Supongamos que queremos realizar un programa que nos solicite dos números por teclado y que visualice la suma de los mismos.

Programa Suma

Las instrucciones que tendríamos que introducir al ordenador, utilizando para ello un lenguaje de programación de terminado, serían parecidas a lo siguiente:

- Instrucción1 - Introduce número1
- Instrucción2 - Introduce número2
- Instrucción3 - Suma número1 y número2 dejando el resultado en número3
- Instrucción4 - Visualiza número3

Pues bien, estas cuatro instrucciones forman lo que se denomina **programa** (Programa Suma).

Si tuviésemos varios programas que sirviesen para realizar otros tratamientos,

- Programa Resta: Restar 2 números
- Programa División: Dividir 2 números
- .....

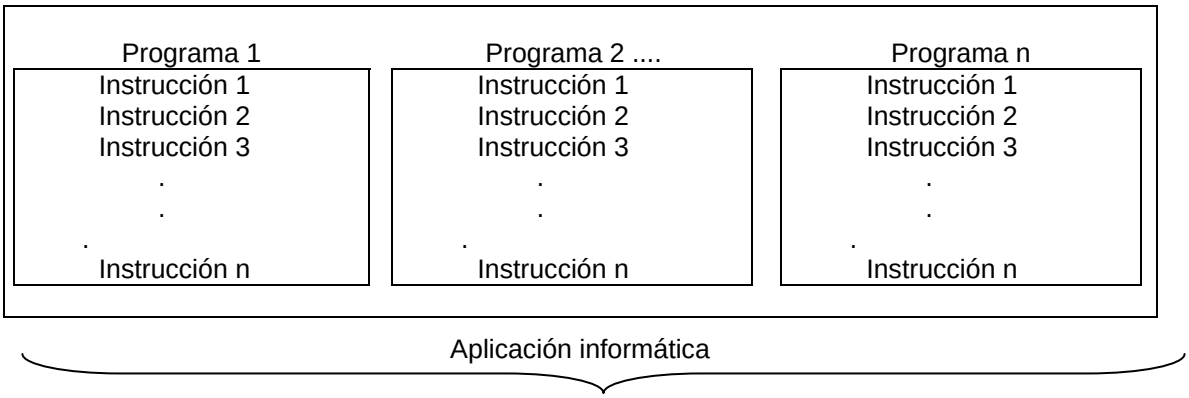
Constituirían una **aplicación informática** con el nombre de *"Calculadora"*

En definitiva, una aplicación es un macroprograma que consta de varios programas independientes aunque interrelacionados; es decir, programas que funcionan de forma autónoma, pero que pueden necesitar información que se ha procesado por otros programas dentro del macroprograma

Pues bien, instrucciones, programas y aplicaciones informáticas, en general, quedan definidos bajo el término **software**.

Por otro lado, hay que considerar que para que estos programas funcionen y puedan generar la información que el usuario precisa, se necesitan determinados componentes físicos. Estos componentes físicos se agrupan bajo la denominación de **hardware**. El conjunto de componentes hardware constituyen un **sistema informático**.

Esquemáticamente:



SOFTWARE

Para que estos programas funcionen:

Componentes físicos ≅ **HARDWARE** ≅ **Sistema informático**

Los componentes físicos ≅ Hardware son tangibles: el usuario puede verlos y tocarlos. Ejemplo – El monitor, la impresora, la unidad de disquetes, el microprocesador, la memoria interna, la fuente de alimentación, los cables, la tarjeta gráfica, etc.

El software es intangible (no se puede tocar). Ejemplo – El software con el que están programadas la memorias ROM.

1 AUTOEVALUACIÓN

El conjunto de varios programas se denomina:

a) Software

b) Aplicación informática

c) Sistema informático

3. LOS COMPONENTES FÍSICOS DE UN ORDENADOR. EL HARDWARE

Los componentes físicos de un ordenador se pueden clasificar:

1) **Microprocesador** (MP) ≅ procesador ≅ **Unidad central de proceso** (UCP ≅ CPU).

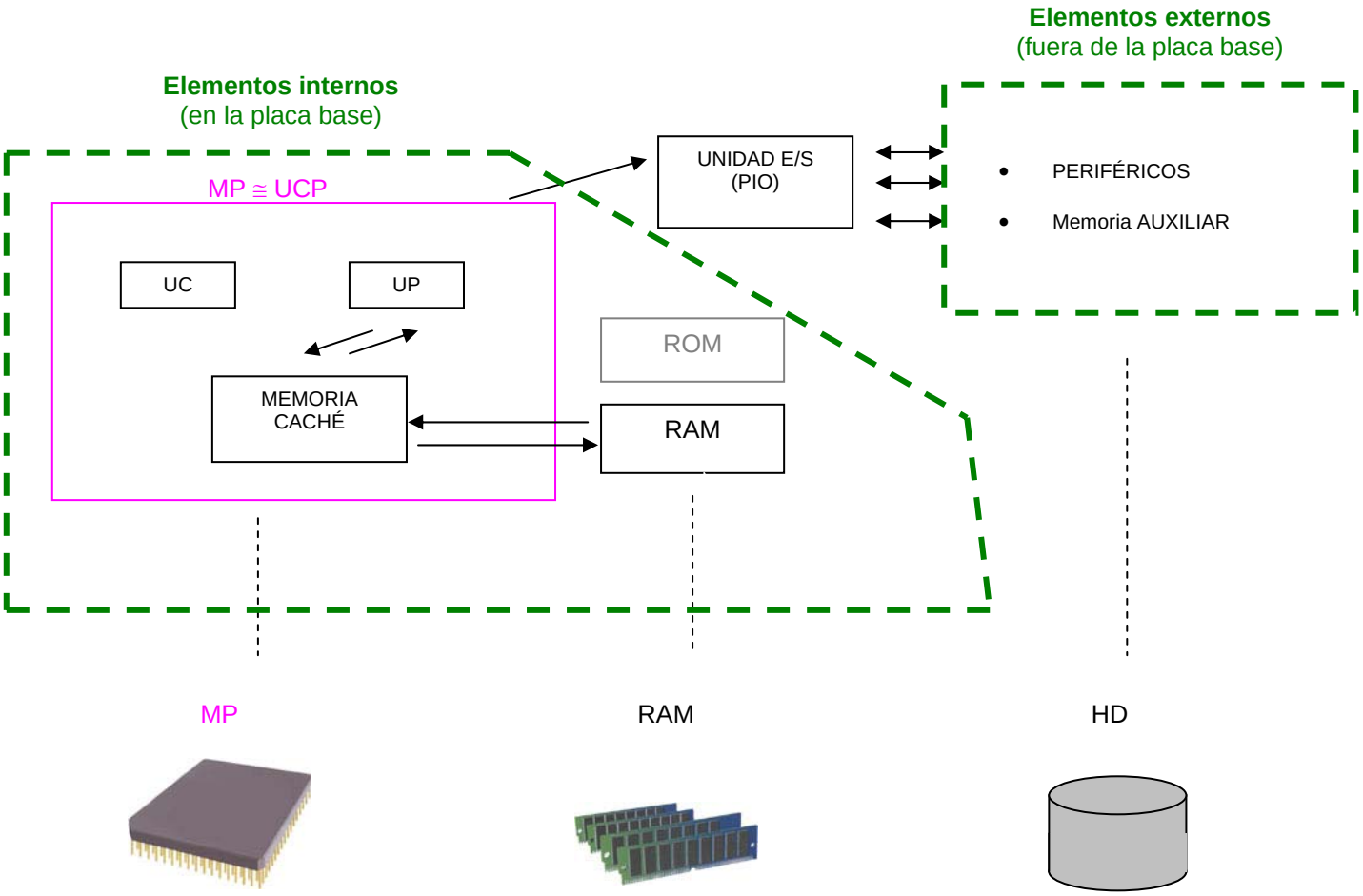
2) **Memoria**

3) **Buses**

4) **Periféricos**

Dichos componentes se encuentran interconectados entre si, con objeto de realizar la función principal del ordenador, que como es sabido es ejecutar programas.

Esquema de los componentes físicos de un ordenador:



Memoria CACHE ≅ Memoria Interna al MP

Memoria RAM ≅ Memoria Interna ≅ Memoria Principal ≅ Memoria Central (MC)

Memoria AUXILIAR ≅ Memoria Externa ≅ Memoria Secundaria - Ejemplo - HD ≅ Disco duro

3.1. El MICROPROCESADOR (UCP)

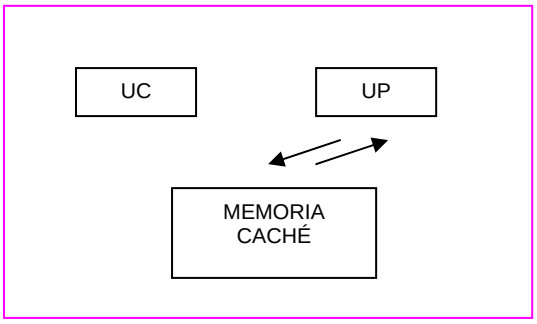
El microprocesador tiene como misión controlar las operaciones del ordenador, es decir, leer las instrucciones, interpretarlas, y ejecutarlas, incluyendo en dicha ejecución tanto las funciones necesarias para el procesamiento de datos, operaciones aritméticas y lógicas, como el envió de las órdenes necesarias a las unidades externas, como puede ser la memoria, los periféricos,

etc, con el fin de realizar el tratamiento automático de la información. Es decir, realmente el microprocesador es el encargado de gobernar el funcionamiento del ordenador. El microprocesador es la parte pensante del ordenador.

Microprocesador (MP) ≅ procesador ≅ **Unidad central de proceso** (UCP) ≅ **Central process unit** (CPU). Está formado por:

- Unidad de control (UC)
- Unidad de proceso (UP)
- Memoria Caché

MP ≅ UCP



El microprocesador tiene los siguientes componentes:

3.1.1. UNIDAD DE CONTROL (UC)

Realiza lo siguiente:

- Controla, coordina e interpreta las instrucciones.
- Gestiona y supervisa el correcto funcionamiento de la unidad de proceso.
- Controla y dirige los componentes externos a ella mediante el envió de señales de control. Entonces les envía esas señales (órdenes) por medio de impulsos para decirles cuando tienen que hacer algo, o sea les dice cuando tienen que ponerse en funcionamiento y cuando tienen que parar.

3.1.2. UNIDAD DE PROCESO (UP)

Trata los datos, realiza operaciones sobre los datos y obtiene un resultado. Es la que realiza el trabajo. La Unidad de proceso está formada por:

- Unidad Aritmético – Lógica (UAL ≅ ALU - Arithmetic Logic Unit)
- Registros

3.1.3. MEMORIA CACHE

Memoria de pequeña capacidad que incorporan los procesadores para que los cálculos de las instrucciones sean mas rápidos, pues la memoria caché es mucho mas rápida que la RAM. O sea si tiene que ejecutar una instrucción, en vez de ejecutarla en la RAM, la ejecuta en la caché para que vaya más rápido.

Esta memoria se coloca entre la CPU y la memoria RAM y almacena y procesa temporalmente la información. Es una memoria ultrarrápida que ayuda al micro en operaciones con datos que maneja constantemente.

2 AUTOEVALUACIÓN

El componente de la UCP que se encarga de realizar las operaciones aritméticas como suma, resta, etc es:

a) La unidad de control

b) La unidad aritmético-lógica

c) Los registros

3.2. MEMORIA

Es el dispositivo donde se almacena los datos y los programas con los que vamos a trabajar. La memoria con la que puede trabajar el ordenador puede ser de dos tipos:

3.2.1. MEMORIA INTERNA

Dos tipos de memorias internas:

- **RAM (Random Access Memory** ≅ Memoria de acceso aleatorio**)**

  Es una memoria de lectura y escritura.

- **ROM (Read Only Memory** ≅ Memoria de sólo lectura**)**

  Es una memoria de sólo lectura, cuya información no se puede modificar.

3.2.2. MEMORIA EXTERNA ≅ Memoria secundaria ≅ Memoria Auxiliar

Permite guardar información en grandes cantidades. La memoria externa o soportes de almacenamiento son: Los disquetes, los CD, los DVD, …

3 AUTOEVALUACIÓN

La memoria interna de un ordenador se divide en los siguientes tipos:

a) Memoria RAM y memoria secundaria

b) Memoria RAM y memoria ROM

c) Memoria RAM y memoria ROM y memoria secundaria

3.3. BUSES

Son un conjunto de líneas eléctricas que permiten la transmisión de señales (información) entre los diferentes componentes del ordenador.

El Bus sirve de interconexión:

- De los periféricos con la memoria interna (RAM).
- De la memoria interna (RAM) con el procesador.
- De la conexión entre las diferentes partes del procesador.

Los buses transmiten la información en paralelo, esto quiere decir, que los datos van por todos los hilos del bus simultáneamente.

El bus es como una autopista en la que el tráfico es muy intenso. Por eso, el tipo de bus que incorpore nuestro ordenador contribuirá a que este sea más rápido o más lento.

3.3.1. TIPOS DE BUSES

Hay diferentes tipos de buses:

3.3.1.1. Bus de control

Envía las “señales” de la UC a todos los demás elementos. Así por ejemplo, se el bus de control se utiliza, entre otras cosas, para comunicarle a la memoria si lo que se quiere es leer o escribir en ella, para que ésta sepa, respectivamente, si tiene que poner el contenido de la celda que indique el bus de direcciones y ponerlo en el bus de datos, o tiene que recoger lo que le llegue por el bus de datos y almacenarlo en la celda de memoria que indique el bus de direcciones.

3.3.1.2. Bus de datos

Envía “datos” entre los periféricos, la RAM y el MP. Conforme han ido evolucionando los ordenadores, el tamaño del bus de datos ha ido creciendo y pasando por tamaños de 8, 16, 32 y 64 bits. Se puede pensar en este tamaño como si fuese el número de carriles que tiene una autovía, cuantos más carriles más coches pueden circular por ella por segundo. Del mismo

modo, cuanto mayor es el ancho de este tipo de buses, mayor es el rendimiento de la máquina, pues mayor caudal de datos puede transportarse en menos tiempo y, de esta manera, se minimiza el tiempo que el procesador tiene que estar esperando a que le lleguen los datos que ha pedido leer o escribir, generalmente de memoria. Por otra parte, los buses también tienen una velocidad asociada que, evidentemente, influye en el rendimiento de la máquina. Evidentemente, no es lo mismo una autovía en el límite de velocidad sea de 80 Km/hora que una en la que se pueda circular a 120.

3.3.1.3. Bus de dirección

Envía y recibe "direcciones" de todos los elementos para conocer donde están los datos.

Permiten al microprocesador seleccionar una de las tantas posiciones de memoria, ya sea para lectura o escritura. Se dice que es unidireccional, pues tan sólo es el procesador el que puede poner información en este bus; el resto de elementos del sistema tan sólo puede leerlo. Cuanto mayor sea este bus, mayor será la cantidad de memoria que el microprocesador puede direccionar o encontrar y, por tanto, marca el máximo de memoria principal que un ordenador puede tener. Así, por ejemplo, con un bus de direcciones de 32 bits, se pueden direccionar 2^{32} posiciones de memoria, o lo que es lo mismo, la memoria puede ser de 2^{32} bytes. Esto es, aproximadamente, un tamaño de 4 GB (Gigabytes).

4 AUTOEVALUACIÓN

El bus de dirección:

a) Envía "datos" entre los periféricos, la RAM y el MP.

b) Envía y recibe "direcciones" de todos los elementos para conocer donde están los datos

c) Envía las "señales" de la UC a todos los demás elementos.

PARA SABER MAS: *Tipos de buses existentes en el mercado*

3.4. PERIFÉFICOS

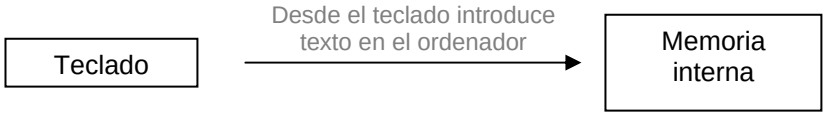
Son dispositivos que sirven para introducir y sacar datos del ordenador.

3.4.1. TIPOS DE PERIFÉRICOS

3.4.1.1. Periféricos de entrada

Son los que sirven para introducir datos en el ordenador.

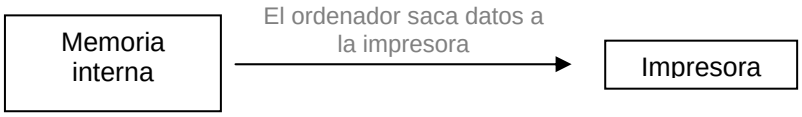
Ejemplos - Teclado, escáner, unidad lectora de CD-ROM, ratón (mouse), joystic....



3.4.1.2. Periféricos de salida

Son los que sirven para sacar datos del ordenador.

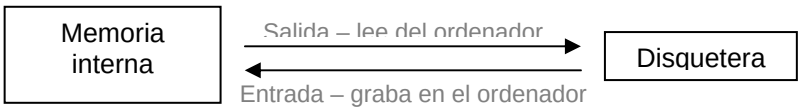
Ejemplos - Impresora, plotter, monitor (pantalla), ...



3.4.1.3. Periféricos de entrada/salida

Son los que sirven para ambas cosas: introducir datos en el ordenador y sacar datos del ordenador.

Ejemplos - Unidades de disquete ≅ disqueteras, los módems, pantallas táctiles, unidades ZIP, CD-RW, discos duros.....



5 AUTOEVALUACIÓN

El protter, es un periférico:

a) De salida

b) De entrada

c) De entrada/salida

4. REPRESENTACIÓN DE LA INFORMACIÓN

4.1. TIPOS DE DATOS

Hay diferentes formar de clasificar los datos:

4.1.1. CLASIFICACIÓN DE LOS DATOS

La primera clasificación que podemos hacer de los datos es la siguiente:

- Datos de entrada

  Son los que se suministran al ordenador desde los periféricos de entrada (teclado, ratón, módem, escáner, etc) o desde los soportes de información (disquetes, discos duros, CD-ROM, etc).

  Forman la 1ª fase de tratamiento automático de la información: ENTRADA

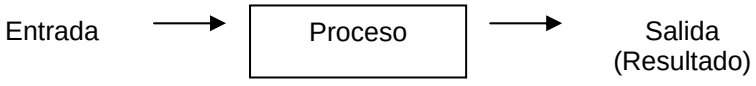
- Datos intermedios

  Son aquellos que se obtienen en la segunda fase del tratamiento automático de la información: PROCESO

- Datos de salida

  También llamados resultados, completan el proceso del tratamiento automático de la información: SALIDA.

  Pueden obtenerse a través de las diferentes unidades periféricas de salida (monitor, impresora, plotter, etc) y, con su posterior distribución y análisis, completan el proceso.



4.1.2. CLASIFICACIÓN DE LOS DATOS: SEGÚN VARIEN O NO DURANTE EL PROCESO

- Datos fijos o constantes

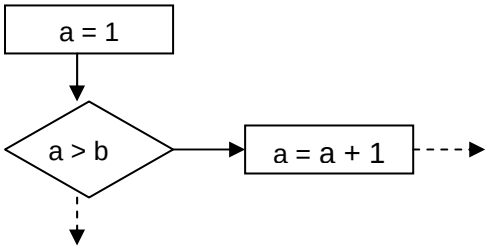
  Son los que permanecen constantes durante el proceso o programa que se les aplique.

Ejemplo – Programa que emite facturas en euros y pesetas. Es evidente que el cambio del euro será el mismo en todo el proceso.

- Datos variables

  Son aquellos que se modifican a lo largo del proceso según sucedan determinadas condiciones o acciones realizadas por los programas.

  Ejemplo – Una variable "a", que al principio del programa vale 1, pero luego se va incrementando (a=a+1) según unas condiciones:



⇒ El valor de "a" no es constante, si no que es variable.

4.1.3. CLASIFICACIÓN DE LOS DATOS: SEGÚN LA FORMA DE SER UTILIZADOS POR EL ORDENADOR

- Datos Numéricos

  Son dígitos del 0 al 9.

- Datos Alfabéticos

  Son letras mayúsculas y minúsculas de la "a" a la "Z"

- Datos Alfanuméricos

  Son una combinación de los anteriores (alfabeticos), más una serie de caracteres especiales.

4.2. SISTEMAS DE NUMERACIÓN

Un sistema de numeración es el conjunto de símbolos y reglas que se utilizan para representar cantidades o datos numéricos.

Cada sistema de numeración se caracteriza por la base.

Base = Nº de símbolos que lo componen.

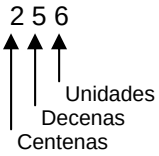
Nosotros utilizamos el sistema de numeración decimal, base 10, símbolos – 0, 1, ..., 9

4.2.1. SISTEMAS POSICIONALES

Los sistemas de numeración son sistemas posicionales. Eso significa que cada dígito tiene un valor definido por su posición ⇒

- El nº más a la derecha representa las unidades
- El siguiente mas a la izquierda representa las decenas
- El siguiente las centenas
- Etc

2 **EJEMPLO**



6 **AUTOEVALUACIÓN**

En el sistema decimal los símbolos utilizados son:

a) 0,1,2,3,4,5,6,7,8,9

b) 0,1,2,3,4,5,6,7,8

c) 1,2,3,4,5,6,7,8,9,10

4.2.2. TEOREMA FUNDAMENTAL DE LA NUMERACIÓN (TFN)

Todos los sistemas posicionales están basados en el Teorema Fundamental de la Numeración (TFN) – Que relaciona una cantidad expresada en cualquier sistema de numeración con la misma cantidad expresada en el sistema decimal, o sea pasa (codifica) números de un sistema a otro.

4.2.2.1. Pasar un nº en cualquier base a base 10:

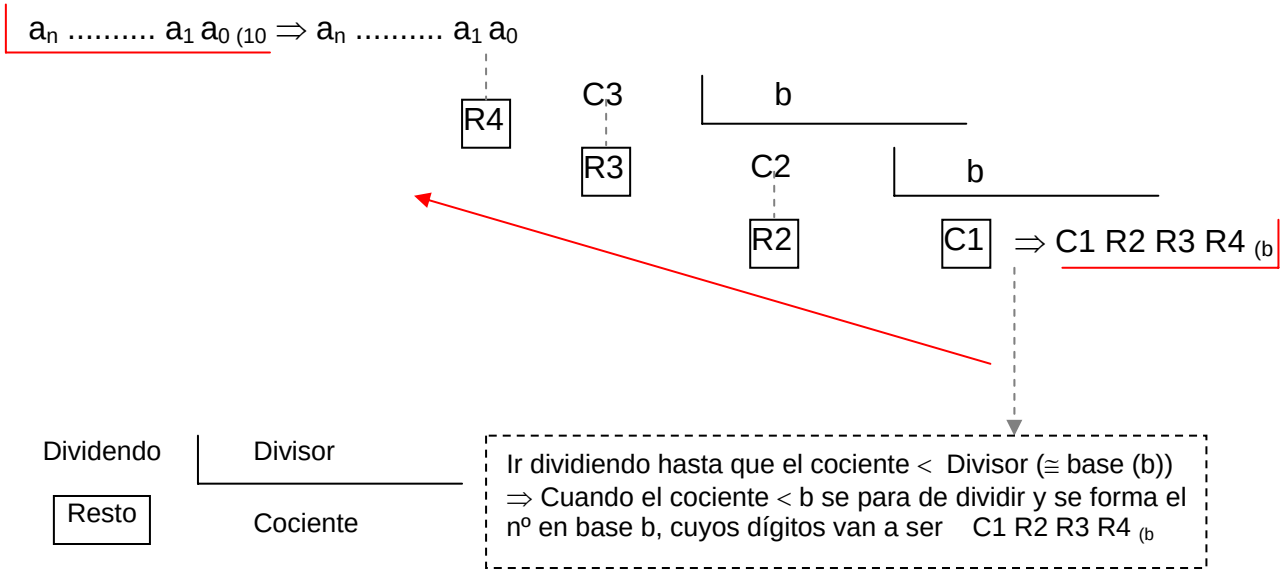
a_n ..... $a_1\ a_0$, $a_{-1}\ a_{-2}$ ..... $a_{-n\ (b} \Rightarrow \sum a_i \cdot b^i{}_{(10} =$

$$a_n . b^n + ... + a_1 . b^1 + a_0 . b^0 + a_{-1} . b^{-1} + a_{-2} . b^{-2} + ... a_{-n} . b^{-n}{}_{(10}$$

Donde:

- a – Valor absoluto del dígito
- b – Base
- i – Posición que ocupa el dígito con respecto al punto decimal

4.2.2.2. Pasar un nº de base 10 a cualquier base:



4.3. DIFERENTES SISTEMAS DE NUMERACIÓN

4.3.1. SISTEMA BINARIO

Base – 2

Símbolos – 0, 1

Es el sistema que maneja el ordenador internamente. Cada uno de estos símbolos recibe el nombre de bit.

bit – es la mínima unidad de información.

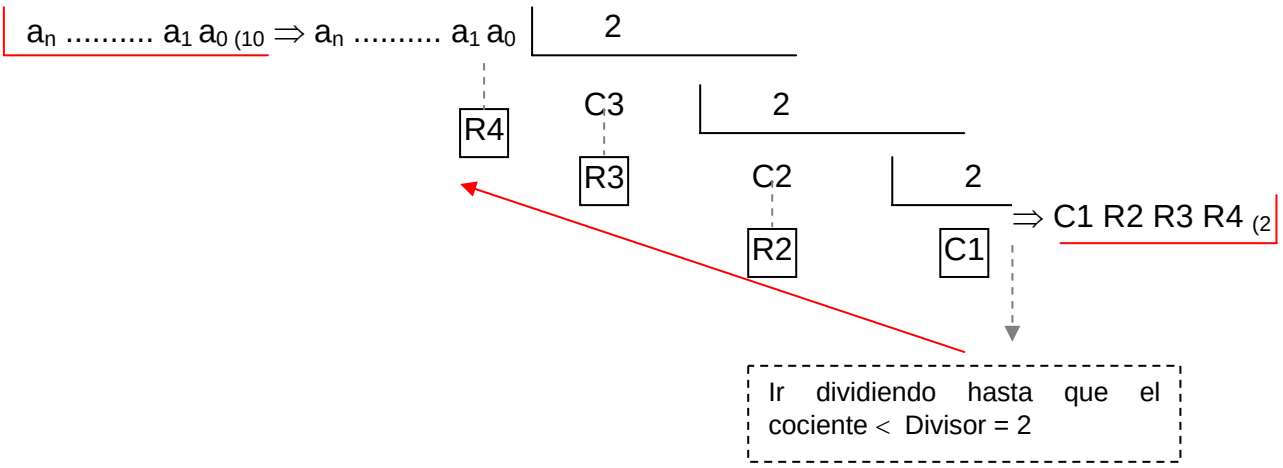
4.3.1.1. Según el Teorema Fundamental de la Numeración (TFN):

- *Pasar un nº en base 2 a base 10:*

a_n ..... $a_1\, a_0$, $a_{-1}\, a_{-2}$ ..... $a_{-n}\,{}_{(2} \Rightarrow \sum a_i \cdot 2^i{}_{(10} =$

$a_n \cdot 2^n$ + ... + $a_1 \cdot 2^1 + a_0 \cdot 2^0 + a_{-1} \cdot 2^{-1} + a_{-2} \cdot 2^{-2}$ + ...+ $a_{-n} \cdot 2^{-n}{}_{(10}$

- *Pasar un nº de base 10 a base 2:*



1 EJERCICIO

Pasar el nº $10101_{(2}$ a base 10:

$$10101_{(2} = a_4 \cdot 2^4 + a_3 \cdot 2^3 + a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0 =$$
$$= 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 =$$
$$= 1 \cdot 16 + 0 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 =$$
$$= 16 + 0 + 4 + 0 + 1 = 21_{(10}$$

2 EJERCICIO

Pasar el nº $21_{(10}$ a base 2:

$21_{(10}$

21	2	
01	10	2
1	0	5	2

1	2	2
0	1	⇒ 10101 (2

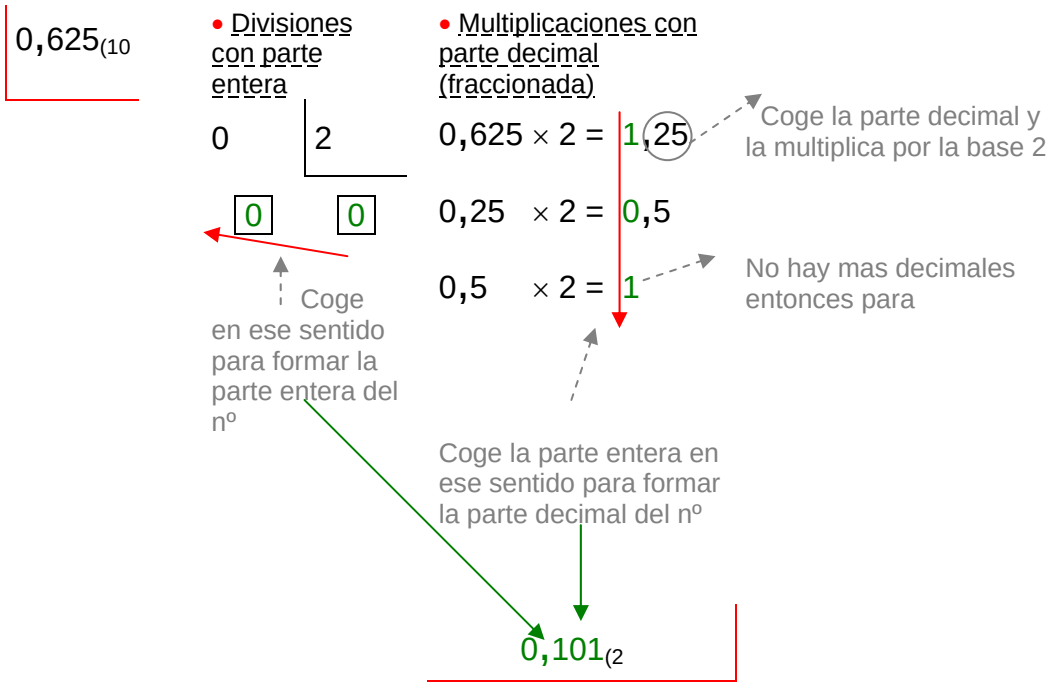
3 EJERCICIO (Con decimales)

Pasar el nº 0,101 (2 a base 10:

$0,101_{(2} = a_0 . 2^0 + a_{-1} . 2^{-1} + a_{-2} . 2^{-2} + a_{-3} . 2^{-3} =$
$= 0 . 2^0 + 1 . 2^{-1} + 0 . 2^{-2} + 1 . 2^{-3} =$
$= 0 . 1 + 1 . 1/2 + 0 . 1/2^2 + 1 . 1/2^3 =$
$= 0 . 1 + 1 . 1/2 + 0 . 1/4 + 1 . 1/8 =$
$= 0 + 1/2 + 0 + 1/8 =$
$= (4+1)/2^3 = 5/8 = 0,625_{(10}$

4 EJERCICIO

Pasar el nº $0{,}625_{(10}$ a base 2:



El **problema** que tiene la conversión de decimales es que sólo pueden convertirse números que acaben en 5 o en 0, ya que cualquier otro número no va a dar un número exacto.

3 EJEMPLO

Pasar el nº $0,3_{(10}$ a base 2:

$0,3_{(10}$

• Divisiones con parte entera

$0 \mid 2$

0 **0**

• Multiplicaciones con parte decimal (fraccionada)

$0,3 \times 2 = 0,6$

$0,6 \times 2 = 1,2$

$0,2 \times 2 = 0,4$

$0,4 \times 2 = 0,8$

$0,8 \times 2 = 1,6$

$0,6 \times 2 = ...$

No pararía pues sigue habiendo decimales.

$0,01001_{(2}$

4 EJEMPLO

Pasar el nº $0,01001_{(2}$ a base 10:

$$0,01001_{(2} = a_0 . 2^0 + a_{-1} . 2^{-1} + a_{-2} . 2^{-2} + a_{-3} . 2^{-3} + a_{-4} . 2^{-4} + a_{-5} . 2^{-5} =$$
$$= 0 . 2^0 + 0 . 2^{-1} + 1 . 2^{-2} + 0 . 2^{-3} + 0 . 2^{-4} + 1 . 2^{-5} =$$
$$= 0 + 0 + 1 . 1/2^2 + 0 + 0 + 1 . 1/2^5 =$$
$$= 1/4 + 1/32 = (8 + 1) / 2^5 = 9/32 = 0,28125_{(10}$$

Vemos que el resultado no es el inicial (0,3). Aunque si obtenemos mas decimales, el error sería menor, pero nunca llegaríamos al 0,3.

4.3.2. SISTEMA OCTAL

Base – 8

Símbolos – 0, ...,7

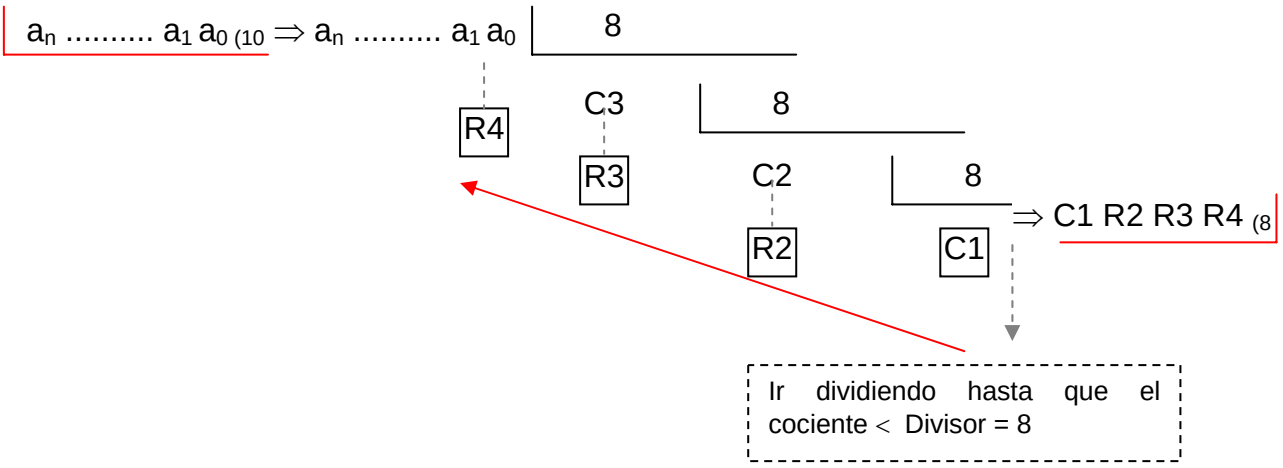
4.3.2.1. Según el Teorema Fundamental de la Numeración (TFN):

- *Pasar un nº en base 8 a base 10:*

$$a_n \ldots\ldots\ldots a_1\, a_0 \,,\, a_{-1}\, a_{-2} \ldots\ldots\ldots a_{-n\,(8} \Rightarrow \sum a_i \,.\, 8^{i}{}_{(10} =$$

$$a_n \,.\, 8^n + \ldots\ldots\ldots + a_1 \,.\, 8^1 + a_0 \,.\, 8^0 + a_{-1} \,.\, 8^{-1} + a_{-2} \,.\, 8^{-2} + \ldots\ldots\ldots + a_{-n} \,.\, 8^{-n}{}_{(10}$$

- *Pasar un nº de base 10 a base 8:*



5 **EJERCICIO**

Pasar el nº $167_{(8}$ a base 10:

$$\begin{aligned} 167_{(8} &= a_2 \,.\, 8^2 + a_1 \,.\, 8^1 + a_0 \,.\, 8^0 = \\ &= 1 \,.\, 8^2 + 6 \,.\, 8^1 + 7 \,.\, 8^0 = \\ &= 1 \,.\, 64 + 6 \,.\, 8 + 7 \,.\, 1 = \\ &= 64 + 48 + 7 = 119_{(10} \end{aligned}$$

6 EJERCICIO

Pasar el nº $119_{(10}$ a base 8:

119(10

119	8
39	14
7	6

1

⇒ 167 (8

7 EJERCICIO (Con decimales)

Pasar el nº $3,2_{(8}$ a base 10:

$3,2_{(8} = a_0 . 8^0 + a_{-1} . 8^{-1} =$
$= 3 . 8^0 + 2 . 8^{-1} =$
$= 3 . 1 + 2 . 1/8 = (24 + 2) / 8 = 26/8 = 3,25_{(10}$

8 EJERCICIO (Con decimales)

Pasar el nº $3,25_{(10}$ a base 8:

$3,25_{(10}$

• Divisiones con parte entera

3	8
3	0

• Multiplicaciones con parte decimal (fraccionada)

$0,25 \times 8 = 2,00$

Para porque no hay decimales

$3,2_{(8}$

4.3.3. SISTEMA HEXADECIMAL

Base – 16

Símbolos – 0, ..., 9 y letras A B C D E F

Estas letras representan respectivamente los símbolos:

10 11 12 13 14 15

A B C D E F

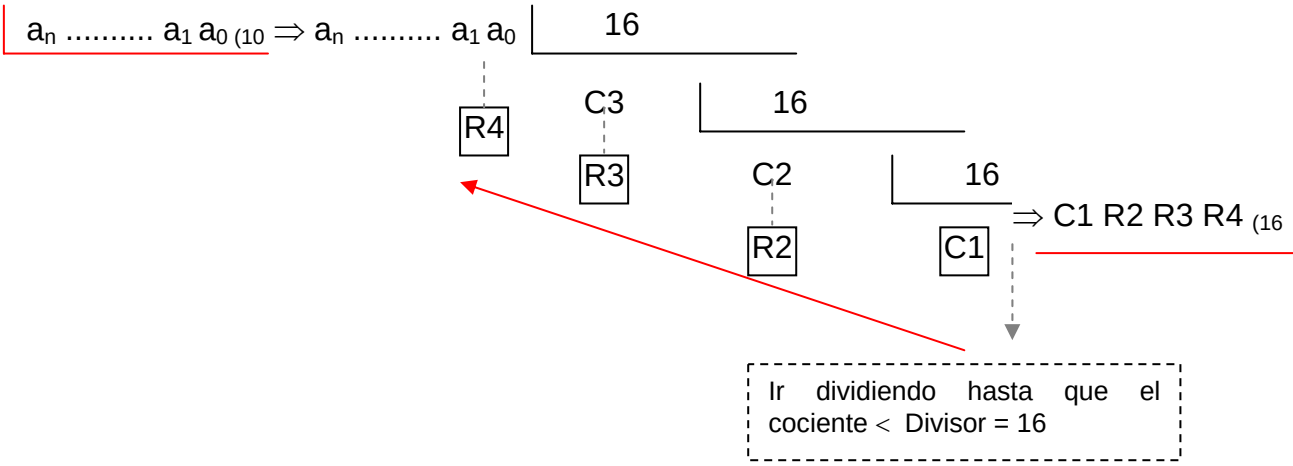
4.3.3.1. Según el Teorema Fundamental de la Numeración (TFN):

- *Pasar un nº en base 16 a base 10:*

$$a_n \ldots\ldots\ldots a_1\, a_0 \,,\, a_{-1}\, a_{-2} \ldots\ldots\ldots a_{-n\,(16} \Rightarrow \sum a_i \,.\, 16^i{}_{(10} =$$

$$a_n \,.\, 16^n + \ldots\ldots\ldots + a_1 \,.\, 16^1 + a_0 \,.\, 16^0 + a_{-1} \,.\, 16^{-1} + a_{-2} \,.\, 16^{-2} + \ldots . + a_{-n} \,.\, 16^{-n}{}_{(10}$$

- *Pasar un nº de base 10 a base 16:*



9 EJERCICIO

Pasar el nº $3CF_{(16}$ a base 10:

$$3CF_{(16} = a_2 \,.\, 16^2 + a_1 \,.\, 16^1 + a_0 \,.\, 16^0 =$$
$$= 3 \,.\, 16^2 + C \,.\, 16^1 + F \,.\, 16^0 =$$
$$= 3 \,.\, 16^2 + 12 \,.\, 16^1 + 15 \,.\, 16^0 =$$
$$= 768 + 192 + 15 = 975_{(10}$$

10 EJERCICIO

Pasar el nº $975_{(10}$ a base 16:

$975_{(10}$

975 | 16

015 | 60 | 16

15= F | 12= C | 3

$\Rightarrow 3CF_{(16}$

11 EJERCICIO (Con decimales)

Pasar el nº $5A2,7C_{(16}$ a base 10:

$$5A2,7C_{(16} = a_2 . 16^2 + a_1 . 16^1 + a_0 . 16^0 + a_{-1} . 16^{-1} + a_{-2} . 16^{-2} =$$
$$= 5 . 16^2 + A . 16^1 + 2 . 16^0 + 7 . 16^{-1} + C . 16^{-2} =$$
$$= 5 . 16^2 + 10 . 16^1 + 2 . 16^0 + 7 . 16^{-1} + 12 . 16^{-2} =$$
$$= 1280 + 160 + 2 + 7/16 + 12/16^2 =$$
$$= (327680 + 40960 + 512 + 112 + 12) / 16^2 = 369276 / 256$$
$$= 1442,484375_{(10}$$

12 EJERCICIO (Con decimales)

Pasar el nº $1442,484375_{(10}$ a base 16:

$1442,484375_{(10}$

• Divisiones con parte entera

1442 | 16

002 | 90 | 2

2 | 10= A | 5

• Multiplicaciones con parte decimal (fraccionada)

0,484375 × 16 = 7,75

0,75 × 16 = 12 = C

Para pues no hay decimales

$5A2,7C_{(16}$

4.3.4. PASAR DE BASE N A BASE M

Podemos transformar un número en base n a otro en otra base m.

4.3.4.1. Según el Teorema Fundamental de la Numeración (TFN):

ara pasar un número de base n a base m, debemos pasar primero por la base 10. Que consiste en pasar el número de la base n a base 10 y posteriormente pasarlo a base m:

1º) Pasar el número de base n a base 10

2º) Pasar el número de base 10 a base m

13 EJERCICIO

Pasar el nº 132(8 a base 16:

$132_{(8} \rightarrow {}_{(10}$

1º) $132_{(8} = 1 \cdot 8^2 + 3 \cdot 8^1 + 2 \cdot 8^0 = 64 + 24 + 2 = 90_{(10}$

2º) $90_{(10} \rightarrow {}_{(16}$

$90_{(10}$ 90 | 16

10 = A | 5 $\Rightarrow 5A_{(16}$

4.3.5. TABLA COMPARATIVA DE LOS SISTEMAS DE NUMERACIÓN

En la siguiente tabla podemos ver los primeros 20 dígitos decimales y sus correspondencias en binario, base 8 y base 16:

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
18	10010	22	12
19	10011	23	13

4.4. SISTEMAS DE CODIFICACIÓN ALFANUMÉRICA

Con los datos alfanuméricos podemos construir instrucciones y programas. El ordenador no solamente procesará datos numéricos; procesa además, datos alfabéticos, y combinaciones de los anteriores, como datos alfanuméricos.

Los sistemas de codificación alfanumérica sirven para representar una cantidad determinada de símbolos en binario. A cada símbolo le corresponderá una combinación de un número de bits.

4.4.1. DIFERENTES SISTEMAS DE CODIFICACIÓN ALFANUMÉRICA

4.4.1.1. Ascii (American Standard Code for Informatión Interchange)

Este sistema utiliza una combinación de 7 u 8 bits (dependiendo del fabricante) para representar cada símbolo. Es el más utilizado.

El que emplea 8 bits puede representar un total de 256 símbolos diferentes (28).

Con el código ASCII podemos representar:

- Los dígitos de 0 al 9
- Letras mayúsculas de la “A” a la “Z”
- Letras minúsculas de la “a” a la “z”
- Caracteres especiales
- Y algunos otros denominados de control

Tabla del código ASCII de 7 bits:

	000 0	001 1	010 2	011 3	100 4	101 5	110 6	111 7
0000 0	NUL 0	DLE 16	SP 32	0 48	@ 64	P 80	` 96	p 112
0001 1	SOH 1	DC1 17	! 33	1 49	A 65	Q 81	a 97	q 113
0010 2	STX 2	DC2 18	" 34	2 50	B 66	R 82	b 98	r 114
0011 3	ETX 3	DC3 19	# 35	3 51	C 67	S 83	c 99	s 115
0100 4	EOT 4	DC4 20	$ 36	4 52	D 68	T 84	d 100	t 116
0101 5	ENQ 5	NAK 21	% 37	5 53	E 69	U 85	e 101	u 117
0110 6	ACK 6	SYN 22	& 38	6 54	F 70	V 86	f 102	v 118
0111 7	BEL 7	ETB 23	' 39	7 55	G 71	W 87	g 103	w 119
1000 8	BS 8	CAN 24	(40	8 56	H 72	X 88	h 104	x 120
1001 9	HT 9	EM 25	) 41	9 57	I 73	Y 89	i 105	y 121
1010 A	LF 10	SUB 26	* 42	: 58	J 74	Z 90	j 106	z 122
1011 B	VT 11	ESC 27	+ 43	; 59	K 75	[91	k 107	{ 123
1100 C	FF 12	FS 28	´ 44	< 60	L 76	\ 92	l 108	| 124
1101 D	CR 13	GS 29	- 45	= 61	M 77	] 93	m 109	} 125
1110 E	SO 14	RS 30	. 46	> 62	N 78	^ 94	n 110	~ 126
1111 F	SI 15	US 31	/ 47	? 63	O 79	_ 95	o 111	DEL 127

La letra “A” en código ASCII será el nº 65, pero en realidad será el nº 65 en binario, pues el ordenador sólo trabaja en binario.

4.4.1.2. Ebcdic (Extended BDC Interchange Code)

Cada símbolo se representa por una combinación de 8 bits agrupados en dos bloques de cuatro. Es el formato extendido de BCD.

Tabla del código EBCDIC:

	0000 0	0001 1	0010 2	0011 3	0100 4	0101 5	0110 6	0111 7	1000 8	1001 9	1010 A	1011 B	1100 C	1101 D	1110 E	1111 F
0000 0	**NUL** 0	**DLE** 16	**DS** 32	48	**SP** 64	**&** 80	- 96	112	128	144	160	176	**{** 192	**}** 208	\ 224	**0** 240
0001 1	**SOH** 1	**DCI** 17	**SOS** 33	49	65	/ 81	97	113	**a** 129	**j** 145	161	177	**A** 193	**J** 209	225	**1** 241
0010 2	**STX** 2	**DC2** 18	**FS** 34	**SYN** 50	66	82	98	114	**b** 130	**k** 146	**s** 162	178	**B** 194	**K** 210	**S** 226	**2** 242
0011 3	**ETX** 3	**TM** 19	35	51	67	83	99	115	**c** 131	**l** 147	**t** 163	179	**C** 195	**L** 211	**T** 227	**3** 243
0100 4	**PF** 4	**RES** 20	**BYP** 36	**PN** 52	68	84	100	116	**d** 132	**m** 148	**u** 164	180	**D** 196	**M** 212	**U** 228	**4** 244
0101 5	**HT** 5	**NL** 21	**LF** 37	**RS** 53	69	85	101	117	**e** 133	**n** 149	**v** 165	181	**E** 197	**N** 213	**V** 229	**5** 245
0110 6	**LC** 6	**BS** 22	**ETB** 38	**UC** 54	70	86	102	118	**f** 134	**o** 150	**w** 166	182	**F** 198	**O** 214	**W** 230	**6** 246
0111 7	**DEL** 7	**IL** 23	**ESC** 39	**EOT** 55	71	87	103	119	**g** 135	**p** 151	**x** 167	183	**G** 199	**P** 215	**X** 231	**7** 247
1000 8	8	**CAN** 24	40	56	72	88	104	120	**h** 136	**q** 152	**y** 168	184	**H** 200	**Q** 216	**Y** 232	**8** 248
1001 9	**RLF** 9	**EM** 25	41	57	73	89	105	\ 121	**i** 137	**r** 153	**z** 169	185	**I** 201	**R** 217	**Z** 233	**9** 249
1010 A	**SMM** 10	**CC** 26	**SM** 42	58	**cent** 74	**!** 90	| 106	**:** 122	138	154	170	186	202	218	234	250
1011 B	**VT** 11	**CU1** 27	**CU2** 43	**CU3** 59	**.** 75	**$** 91	**,** 107	# 123	139	155	171	187	203	219	235	251
1100 C	**FF** 12	**IFS** 28	44	**DC4** 60	< 76	* 92	**%** 108	@ 124	140	156	172	188	204	220	236	252
1101 D	**CR** 13	**IGS** 29	**ENQ** 45	**NAK** 61	**(** 77	**)** 93	_ 109	' 125	141	157	173	189	205	221	237	253
1110 E	**SO** 14	**IRS** 30	**ACK** 46	62	+ 78	**;** 94	> 110	= 126	142	158	174	190	206	222	238	254
1111 F	**SI** 15	**IUS** 31	**BEL** 47	**SUB** 63	| 79	¬ 95	**?** 111	" 127	143	159	175	191	207	223	239	255

4.4.1.3. Fieldata

Utiliza bloques de 6 bits para representar los diferentes símbolos. De poco uso.

7 AUTOEVALUACIÓN

El sistema de codificación alfanumérica Ascii para representar cada símbolo utiliza:

a) 8 bits

b) Una combinación de 7 u 8 bits

c) 6 bits

4.5. REPRESENTACIÓN EN EL ORDENADOR

Vamos a ver como se almacenan en el ordenador los números.

Un ordenador sólo procesa información digital ⇒ Hay que traducir los datos que se introducen en el ordenador en forma de señales eléctricas:

0 voltios – equivale a un 0 lógico

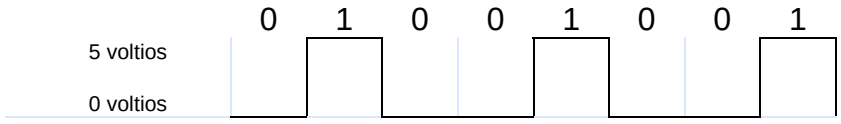
5 voltios – equivale a un 1 lógico

5 EJEMPLO

Si queremos introducir el carácter “a” debemos codificarlo previamente:

a = 01001001

Esto significa que por los circuitos del ordenador aparecerá la siguiente señal:



⇒ En el ordenador los números se almacenan en código binario

4.5.1. FORMAS DE ALMACENAR LOS NÚMEROS EN EL ORDENADOR

4.5.1.1. Coma fija o punto fijo

Se usa para representar números enteros.

Hay 3 formas de representar los números en coma fija:

- *Binario puro*

  Se utiliza una combinación de 32 bits, en la que:

  - El bit de la izquierda sirve para representar el signo:

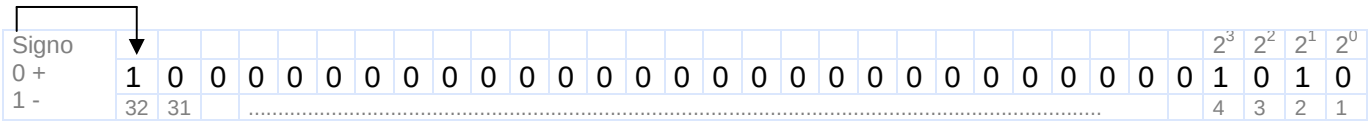
El 0 para el signo +

El 1 para el signo –

- Los restantes 31 bits sirven para representar el valor del número

6 **EJEMPLO**

El nº -10 tendría la siguiente forma:



- *Decimal desempaquetado*

  Representa cada número decimal, de tal forma que cada una de sus cifras ocupa un byte o octeto (8 bits), en los que:

  - Los 4 bits de la izquierda son cuatro unos denominados bits de zona.
  - Los 4 bits de la derecha representan las cifras

  Exceptuando los 8 bits de la derecha:

  - Los 4 bits de la izquierda representan el signo:

    El 1100 para el signo +

    El 1101 para el signo –

  - Los 4 bits de la derecha representan la cifra de la derecha

7 **EJEMPLO**

Representar el nº 2371 decimal en decimal desempaquetado:

1 1 1 1 0 0 1 0 | 1 1 1 1 0 0 1 1 | 1 1 1 1 0 1 1 1 | 1 1 0 0 0 0 0 1

2 | 3 | 7 | Signo + | 1

8 **EJEMPLO**

Representar el nº -2371 decimal en decimal desempaquetado:

1 1 1 1 0 0 1 0 | 1 1 1 1 0 0 1 1 | 1 1 1 1 0 1 1 1 | 1 1 0 1 0 0 0 1

2 | 3 | 7 | Signo - | 1

- *Decimal empaquetado*

  Representa cada cifra con un conjunto de 4 bits. El conjunto de 4 bits de la derecha se usa para representar el signo con la misma combinación que en el caso anterior.

9 EJEMPLO

Representar el nº 2371 decimal en decimal empaquetado:

0 0 1 0 0 0 1 1 0 1 1 1 0 0 0 1 1 1 0 0

2 3 7 1 Signo +

4.5.1.2. Coma flotante

Se utiliza para representar números reales y enteros con un rango de representación mayor que el que ofrece el punto fijo. Con esto conseguimos que el ordenador pueda tratar números muy grandes o muy pequeños.

Para representar así los números, se utiliza la notación científica, que se representa de la forma:

n1 = mantisa × base de exponenciación exponente

- El exponente también se denomina característica.
- La mantisa es un número real con el punto decimal implícito a la izquierda de los bits que lo representan.
- La base de exponenciación es una potencia de 2 que dependerá del fabricante de componente.

La representación de números en coma flotante se puede hacer de dos formas:

- *Simple precisión*

  Se utilizan 32 bits para representar cualquier cantidad numérica.

- *Doble precisión*

  Se utiliza una combinación de 64 bits para representar una cifra.

8 AUTOEVALUACIÓN

Hay varias formas de representar los números en coma fija:

a) Binario puro, decimal desempaquetado, decimal empaquetado

b) Simple precisión, doble precisión

4.6. OPERACIONES CON NÚMEROS BINARIOS

4.6.1. SUMA BINARIA

Igual que podemos sumar números decimales, también podemos hacerlo con números binarios. Sólo hay que recordar que aquí también podemos tener acarreos (números que sobrepasan la cantidad máxima y hay que sumarlos a las siguientes cifras.

Las cuatro reglas de la suma binaria son:

X + Y	SUMA	ACARREO
0 + 0	0	0
0 + 1	1	0
1 + 0	1	0
1 + 1	0	1

14 **EJERCICIO**

Decimal	*Binario*	
	1	*Acarreos*
10	1010	
+ 3	0011	
13	1101	

15 **EJERCICIO**

1	11	*Acarreos*
12	1100	
+ 39	100111	
51	110011	

4.6.2. RESTA BINARIA

La resta binaria es justamente la suma invertida. Hay que tener en cuenta que al restar un dígito alto a otro mas bajo (restar 1 de 0), el resto es 1 y tomamos 1 de la columna de la izquierda. Si este es 1, lo haremos 0, y si es 0 lo haremos 1 y tomaremos 1 de la siguiente columna. Las reglas de la resta son:

X - Y	RESTA	ACARREO
0 - 0	0	0
0 - 1	1	1
1 - 0	1	0
1 - 1	0	0

16 **EJERCICIO**

Decimal	*Binario*	
19	10011	
- 5	00101	
	11	*Acarreos*
14	01110	

4.6.3. MULTIPLICACIÓN BINARIA

La regla para multiplicar es copiar el multiplicando si la cifra del multiplicador es 1 y copiar ceros si es 0. Después sumar los productos parciales. Solo hay cuatro productos binarios, la tabla será:

X * Y	MULTIPLICACIÓN	ACARREO
0 * 0	0	0
0 * 1	0	0
1 * 0	0	0
1 * 1	1	0

17 **EJERCICIO**

Decimal		*Binario*
	5	101
×	3	11
		101
		101
	15	1111

4.6.4. DIVISIÓN BINARIA

La división binaria por 0 no tiene sentido, por tanto la tabla será:

X / Y	DIVISIÓN	ACARREO
0 / 1	**0**	**0**
1 / 1	**1**	**0**

18 EJERCICIO

Decimal

```
2 0 | 2
0 0   1 0
    0
```

Binario

```
1 0 1 0 0 | 1 0
0 0 1 0     1 0 1 0
      1 0 0 0
          0
```

Cuando los números son muy grandes las operaciones se dificultan, por lo que suelen emplearse los Sistemas Octal y Hexadecimal.

4.7. LA MEDIDA DE LA INFORMACIÓN

- **Bit** - es la mínima unidad de información. Éste queda representado por un 0 (falso) o un 1(verdadero).

  En este sentido, se puede establecer un equivalencia de medidas en múltiplos de bits utilizados para designar cada medida:

- **Nibble o cuarteto** – Conjunto de 4 bits.
- **Byte (b), octeto o carácter** - Conjunto de 8 bits.

  Es la cantidad en bits necesaria para representar un carácter alfanumérico.

  Un carácter es – Una letra, un número, o signo de puntuación. Ocupa un byte, o sea 8 bits.

10 EJEMPLO

Si decimos que un archivo de texto ocupa 4.000 bytes queremos decir que contiene el equivalente a 4.000 letras (que son entre 2 y 3 páginas de texto sin formato).

11 EJEMPLO

4 bytes = 32 bits

1 byte = 2 cuartetos

- **Kilobyte (Kb) (Kbyte)** - Conjunto de 1.024 bytes.

  Kilo (K) = 2^{10} = 1.024

Se toma el valor de 1.024 en vez de 1.000 precisamente por ser 1.204 una potencia de 2, y en consecuencia, un valor mucho más conveniente para máquinas que trabajan en sistema binario.

- **Megabyte (Mb) (Mbyte)** - Conjunto de 1.024 Kb.

  Mega (M) = 220 = 1.048.576 Kb = 210 Kb

- **Gigabyte (Gb) (Gbyte)** - Conjunto de 1.024 Mb.

  Giga (G) = 230 = 1.073.741.824 Mb = 210 Mb

- **Terabyte (Tb) (Tbyte)** - Conjunto de 1.024 Gb.

  Tera (T) = 240 = 210 Gb

- **Petabyte (Pb) (Pbyte)** - Conjunto de 1.024 Tb.

  Peta (P) = 250 = 210 Tb

- **Exabyte (Eb) (Ebyte)** - Conjunto de 1.024 Pb.

  Exa (E)= 260 = 210 Pb

- Actualmente en los libros suelen poner esta medidas de la siguiente forma:

  KiloByte = KB = 1.204 Bytes

  Kilobit = Kb = 1.204 bits

Unidad 1 - Contenidos _Método directo o rápido

♦ Correspondencia directa entre los Sistemas Binario y Octal. (Método rápido)

Teniendo en cuenta la siguiente correspondencia:

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

Para pasar un número binario a octal agrupamos sus cifras desde la derecha de tres en tres. Cada cifra octal se obtiene del valor obtenido de cada grupo de tres bits.

Ejemplos

$10101_{(2}$ … 010 101
2 5 …….. $25_{(8}$

Para pasar un número octal a binario se procede a la inversa
$43_{(8}$ … 4 3
100 011 …….. $100011_{(2}$

Ejemplos (Con decimales)

$101,01_{(2}$ … 101 010
5 2 …….. $5,2_{(8}$

Para pasar un número octal a binario se procede a la inversa
$43_{(8}$ … 4 3
100 011 …….. $100,011_{(2}$

♦ Correspondencia directa entre los Sistemas Binario y Hexadecimal. (Método rápido)

Teniendo en cuenta la siguiente correspondencia:

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

Para pasar un número binario a hexadecimal agrupamos sus cifras desde la derecha de cuatro en cuatro. Cada cifra hexadecimal se obtiene del valor correspondiente a cada grupo de cuatro bits.

EJEMPLOS

$10101_{(2}$ … 0001 0101
1 5 …….. $15_{(16}$

Para pasar un número hexadecimal a binario se procede a la inversa

$43_{(16}$ … 4 3
0100 0011 ……. $1000011_{(2}$

Ejemplos (Con decimales)

$101,01_{(2}$ … 0101 0100
5 4 …….. $5,4_{(16}$

Para pasar un número octal a binario se procede a la inversa

$4,3_{(16}$ … 4 3
0100 0011 …….. $100,0011_{(2}$

Ejercicios 1

1

El conjunto de varios programas se denomina:

a) Software
b) Aplicación informática
c) Sistema informático

2

El componente de la UCP que se encarga de realizar las operaciones aritméticas como suma, resta, etc es:

a) La unidad de control
b) La unidad aritmético-lógica
c) Los registros

3

La memoria interna de un ordenador se divide en los siguientes tipos:

a) Memoria RAM y memoria secundaria
b) Memoria RAM y memoria ROM
c) Memoria RAM y memoria ROM y memoria secundaria

4

El bus de dirección:

a) Envía “datos” entre los periféricos, la RAM y el MP.
b) Envía y recibe “direcciones” de todos los elementos para conocer donde están los datos
c) Envía las “señales” de la UC a todos los demás elementos.

5

El protter, es un periférico:

a) De salida
b) De entrada
c) De entrada/salida

6

En el sistema decimal los símbolos utilizados son:

a) 0,1,2,3,4,5,6,7,8,9

b) 0,1,2,3,4,5,6,7,8

c) 1,2,3,4,5,6,7,8,9,10

7

El sistema de codificación alfanumérica Ascii para representar cada símbolo utiliza:

a) 8 bits

b) Una combinación de 7 u 8 bits

c) 6 bits

8

Hay varias formas de representar los números en coma fija:

a) Binario puro, decimal desempaquetado, decimal empaquetado

b) Simple precisión, doble precisión

Ejercicios 2

1. Escribe en el sistema de numeración decimal los números $11111111_{(2}$, $1001{,}011_{(2,}$ $460_{(8}$, $175{,}05_{(8}$, $86BF_{(16}$ y $DAFE{,}02_{(16}$.
2. Escribe en el sistema de numeración binario los números $132_{(10}$, $20{,}25_{(10}$.
3. Escribe en el sistema de numeración octal los números $132_{(10}$, $20{,}25_{(10}$.
4. Escribe en el sistema de numeración hexadecimal los números $132_{(10}$, $20{,}25_{(10}$.
5. Completa las siguientes tablas de códigos.

BINARIO	**1110**			
DECIMAL		**123**		
OCTAL			**6256**	
HEXADECIMAL				**FF**

BINARIO				**111111101010**
DECIMAL	**169**			
OCTAL		**753**		
HEXADECIMAL			**1C1**	

BINARIO			**1010101110101011**	
DECIMAL				**21**
OCTAL	**621**			
HEXADECIMAL		**6E**		

BINARIO		**0,101**		
DECIMAL			**41,5**	
OCTAL	**41,5**			
HEXADECIMAL				**E7,CA**

6. ¿Cuántos dígitos necesitarías para representar en binario, el número decimal 1200?
7. Si el número 111111 en base 2 se corresponde con el 63 en decimal... mediante la lógica, y sin realizar muchos cálculos ¿cómo se representaría el número 62? ¿Y el 64?
8. ¿Es cierta la siguiente afirmación?
   $10110001101011_2 \rightarrow 26153_{10}$
9. Convertir el nº $111000_{(2}$ a base 16: (1º usando el Teorema Fundamental de la Numeración y 2º por el método rápido o directo).
10. Convertir el nº $CEB1_{(16}$ a base 2: (1º usando el Teorema Fundamental de la Numeración y 2º por el método rápido o directo).
11. Convertir el nº $35_{(8}$ a base 16: (1º usando el Teorema Fundamental de la Numeración y 2º por el método rápido o directo).
12. Convertir el nº $101_{(16}$ a base 8: (1º usando el Teorema Fundamental de la Numeración y 2º por el método rápido o directo).
13. Dados los siguientes valores del sistema numérico decimal, convertir cada uno de ellos a números binarios y luego sumarlos, expresando la respuesta en el sistema numérico binario.
    $13_{(10} + 5_{(10}$
    $4_{(10} + 2_{(10} + 3_{(10}$
    $7{,}25_{(10} + 6{,}5_{(10}$
14. Dados los siguientes valores del sistema numérico decimal, convertir cada uno de ellos a números binarios y luego restarlos, expresando la respuesta en el sistema numérico binario.
    $29_{10} - 7_{(10}$
    $10_{(10} - 4_{(10}$
    $8{,}25_{(10} - 4{,}5_{(10}$
15. Dados los siguientes valores del sistema numérico decimal, convertir cada uno de ellos a números binarios y luego multiplicarlos, expresando la respuesta en el sistema numérico binario.
    $29_{10} \times 7_{(10}$
    $10_{(10} \times 4_{(10}$
    $12{,}5_{(10} \times 13{,}25_{(10}$
16. Dividir en binario:
    111100 y 1010
    1100000 y 101
    11100001 y 1010
17. Codificar en decimal desempaquetado y empaquetado los siguientes números y sus negativos:
    67 100000

Ejercicios 3

En el ejercicio anterior hemos manejado códigos numéricos, que pueden corresponder por ejemplo a una dirección de memoria, etc... Pero en el ordenador se maneja todo tipo de información, no sólo la numérica. No sólo existen códigos numéricos como BCD (4 bits)..., sino que, como sabemos existen códigos alfanuméricos, ASCII (7 u 8 bits), EBCDIC (8 bits), FIELDATA (6 bits),..... que hacen corresponder cada carácter con una cadena binaria de un nº de bits.

a) Completa la siguiente tabla a partir de las tablas de código ASCII.

Carácter	Número decimal en el código ASCII	Número Hexadecimal en el código ASCII
A	65	41
x		
	91	
		61
*		
		50
	84	
9		

b) Usando ASCII de 8 bits y EBCDIC transcribe a una cadena binaria la palabra CADENA.

c) Usando ASCII de 8 bits y EBCDIC transcribe a una cadena binaria la frase "HOY es 3-10-01".

d) ¿Tiene alguna ventaja o inconveniente usar un código u otro?

e) ¿Cuántos caracteres distintos pueden representarse con estos códigos: BCD, FIELDATA, ASCII, EBCDIC?

f) Descifra el mensaje siguiente, codificado en ASCII de 8 bits.

486173206465736369667261646F20656C2063F36469676F2041534349492E0D0AA1456E686F72616275656E61210D0A

g) Imaginemos que tenemos que inventar un código que haga corresponder cada carácter con una cadena binaria. Como necesito usar, los números 0 al 3, las vocales tanto en mayúsculas como en minúsculas y sólo algunas consonantes {b, c, d}, ¿de cuántos bits será el código que necesito? Diséñalo.