Fundamentos de Hardware

UD1
Sistemas informáticos.
Estructura funcional.

Profesor: Francisco de Asís GONZÁLEZ CAVERO

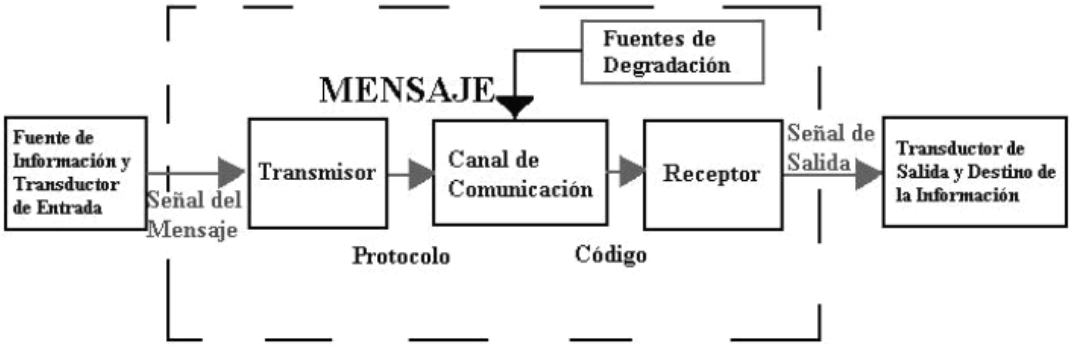
Índice de contenido

1.1. Introducción a los sistemas informáticos

1.1.1. Definición de un sistema informático

En su acepción más general, llamamos **sistema** a aquel conjunto ordenado de elementos que se relacionan entre sí y contribuyen a un determinado objetivo.

Es evidente que existen múltiples tipos de sistemas pero para lo que nos ocupa, tomamos como punto de partida la idea de los **sistemas de comunicación** entendidos como *aquél conjunto de elementos que emiten, reciben e interpretan información.*

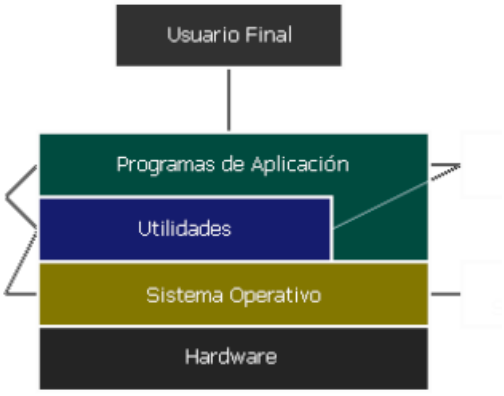


Sistema De Comunicación

En la actualidad, debido al auge de las redes de ordenadores y la evolución de las nuevas tecnologías, el término sistema informático ha desplazado en el ámbito profesional, a otros términos como ordenador o computador.

Un **sistema informático (SI)** es un conjunto de componentes físicos (*hardware*) y lógicos (*software*).

La **parte física o hardware** está formada por todos los elementos electrónicos y mecánicos. Son los elementos del ordenador como, por ejemplo, el chasis del ordenador, los circuitos internos del ordenador, el microprocesador; los dispositivos de entrada y salida (E/S) de la información hacia o desde el ordenador, como la pantalla, el ratón, el teclado, la unidad de DVD, la tarjeta de red, etc.



La **parte lógica o software** está formada por todos los elementos no físicos, como el **sistema operativo**, los **programas** de aplicaciones, los **datos** almacenados dentro del ordenador, etc.

Entre los programas y el hardware se encuentra una aplicación informática especial, que hace de intermediario entre ambos. Ese software se denomina **sistema operativo**.

También os encontrareis con otro concepto, el **firmware**. En realidad es simplemente un software que viene integrado directamente dentro de un hardware, en una memoria especial. Así una grabadora de DVD cuenta con un chip de memoria especial, donde hay almacenado un software que le indica a que velocidad puede grabar y de que forma lo hace. El software que se encuentra en ese chip se puede denominar firmware.

A lo largo de este tema, junto con el siguiente, veremos con más detalle las características funcionales y físicas de un SI entendiendo por:

- **Estructura funcional del SI**. Aquella asociada al soporte físico o hardware que se encarga de estudiar las arquitecturas de organización y funcionamiento de los diversos componentes del mismo. Veremos la arquitectura tradicional de un ordenador, que toma como punto de partida la Arquitectura de Von Neumann.
- **Estructura física del SI**. También asociada al hardware. En este caso estudiaremos el hardware comercial. Veremos cómo son físicamente, para qué sirven y qué características tienen los diferentes componentes actuales que componen un PC.

1.1.2. Evolución histórica de los SI

Para ver cómo han evolucionado los sistemas operativos a lo largo de la historia, tenemos que tener muy presentes las arquitecturas de los ordenadores, es decir la evolución del hardware sobre el que se instalan. En general, podemos hablar de cuatro generaciones de sistemas operativos:

- **Primera generación (1945 – 1955).** Se utilizaban las **válvulas de vacío**. Estas computadoras, que no ordenadores, eran máquinas programadas en lenguaje máquina puro (lenguaje de muy bajo nivel) y surgen por una necesidad vital al considerarse un instrumento armamentístico durante la Segunda Guerra Mundial. Eran de **gran tamaño, elevado consumo de energía y muy lentas.** Las operaciones se reducían a simples cálculos matemáticos. La forma de introducir los datos en estas computadoras se hacía a modo de centralita de teléfonos antigua, pinchando clavijas en unos paneles enormes llenos de agujeros. Según se pinchaba la clavija en uno u otro lugar, se indicaba qué números se iban a procesar y qué operación se iba a realizar. Posteriormente, a principios de los años cincuenta, para introducir datos en el ordenador se utilizaban las **tarjetas perforadas**, pudiendo introducir más datos de forma más rápida y, lo más importante, se podía repetir el mismo proceso sin tener que volver a introducir de nuevo todos los datos de forma manual.

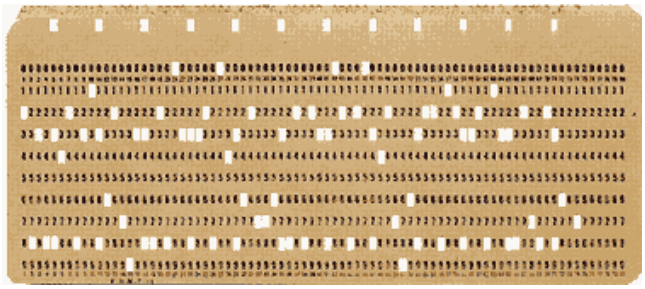


Ilustración 1: Tarjeta perforada.

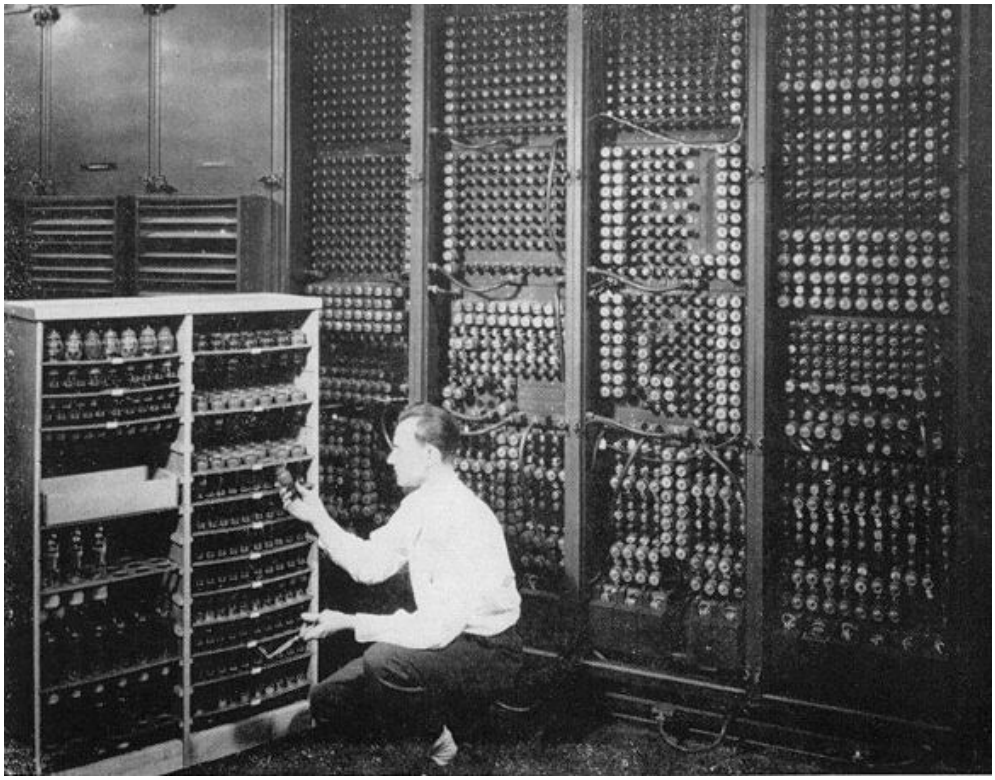


Ilustración 2: ENIAC. Compuesto de unos 17468 tubos de vacío.

- **Segunda generación (1955 – 1965)**. Aparecen los **transistores**, que se introducen dentro de la arquitectura de las computadoras. Desaparecen las válvulas de vacío, por lo que las **computadoras** se hacen **más pequeñas y baratas**, consumen menos y despiden menos calor. En esta generación aparece lo que se denomina **procesamiento por lotes**. Este proceso consiste en que los datos se introducen en la computadora no de forma manual mediante clavijas, ni mediante tarjetas perforadas, sino a través de otro pequeño computador o componente hardware que previamente ha sido cargado con la información a procesar. Como podemos apreciar, ya aparece, si así se puede decir, el concepto de **periférico**.

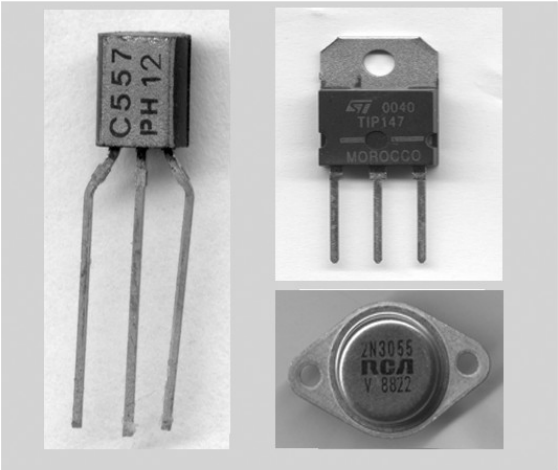


Ilustración 3: Transistor.

- **Tercera generación (1965 – 1971)**. La tercera generación de computadoras emergió con el desarrollo de **circuitos integrados** (pastillas de silicio) en las que se colocan miles de componentes electrónicos en una integración en miniatura. Las computadoras nuevamente se hicieron más pequeñas, más rápidas, desprendían menos calor y eran energéticamente más eficientes. Aparecen los primeros **SOMU** (Sistemas operativos multiusuario), varios usuarios pueden introducir programas que se ejecutan simultáneamente. En esta década se crea la computadora **IBM 360**, era una maquina capaz de realizar operaciones de tipo matemático y de tipo lógico.

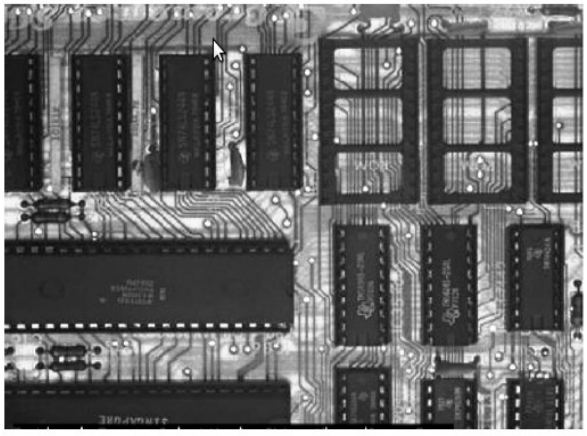


Ilustración 4: Circuito integrado.

Ilustración 5: IBM 360.

- **Cuarta generación (1971-1984).** Podemos citar, dentro de la cuarta generación, las siguientes características:
    1. Se desarrolla el microprocesador.
    2. Se utiliza la integración **LSI** (*large scale integration*, 'gran escala de integración'): tecnología de construcción de los circuitos integrados que permite tener entre 1000 y 10000 componentes lógicos.
    3. Se utiliza la integración **LSI** de **microordenadores** y **ordenadores personales**. La tecnología permite integrar más circuitos en una sola pastilla. Esto reduce el espacio y el consumo haciendo asequible el ordenador a cualquier persona.
    4. Surgen una gran cantidad de lenguajes de programación.
    5. Muchas familias comenzaron a tener computadores en sus casas como las famosas Commodore 64 y 128, ZX Spectrum o Amstrad CPC.
- **Quinta generación (1982-1991).** El microprocesador sigue evolucionando reduciendo su tamaño y permitiendo muchas más operaciones y funcionalidades. Sus características son:
    1. Tecnología **VLSI** o de muy **alta escala de integración** (de 10.000 a 100.000 transistores).
    2. Evolución muy rápida de la tecnología, lo que se fue a llamar la **Ley de Moore.**
    3. Desarrollo y expansión de la tecnología multimedia.
    4. Aparecen microprocesadores trabajando en paralelo, y se extiende el uso generalizado de las redes.

- **Sexta generación (1992-actualidad).** Se utilizan tecnologías superiores de integración como **ULSI** (Ultra Large Scale Integration) que emplea entre 100.000 y 1.000.000 de transistores y la **GLSI** (Giga Large Scale Integration) con más de un millón de transistores. Además, se ha extendido la conectividad de las computadoras mediante el empleo de **redes**, y cada vez crece más el uso de aplicaciones soportadas por la propia red como Internet.

1.1.3. Funcionamiento básico de un sistema informático

Un sistema informático maneja información de todo tipo (números, texto, imágenes, sonidos, vídeo, etc), dándole entrada, salida o procesándola. Para ello utilizará mecanismos de representación, almacenamiento y presentación como veremos a continuación.

Un ordenador, debido a su construcción basada fundamentalmente en circuitos electrónicos digitales, trabaja con el sistema binario (1 = 5V, 0 = 0V), usando una serie de códigos que permiten su funcionamiento.

Este es el motivo que nos obliga a transformar internamente todos nuestros datos, tanto numéricos como alfanuméricos, a una representación binaria para que la máquina sea capaz de procesarlos. Además del sistema de numeración binario, el ordenador también trabaja para la codificación numérica con los sistemas de numeración octal y hexadecimal.

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

Tienen como característica una base a la que referencian y que determina el diferente número de símbolos que lo componen. Nosotros utilizamos el sistema de numeración en base 10, compuesto por diez símbolos diferentes (del 0 al 9).

Los sistemas de numeración que utilizamos son sistemas posicionales, es decir, el valor relativo que cada símbolo representa quedará determinado por su valor absoluto y la posición que ocupe dicho símbolo en un conjunto.

Todos los sistemas posicionales están basados en el **Teorema Fundamental de la Numeración (TFN)**, que sirve para relacionar una cantidad expresada en cualquier sistema de numeración con la misma cantidad expresada en el sistema decimal.

Viene dado por la fórmula siguiente, donde *x* es el número, *X* es el valor absoluto del dígito en cuestión, *j* es la posición que ocupa el dígito con respecto al punto decimal, *i* es el total de dígitos utilizados para representar el número y *b* es la base.

$$x = \sum_{j=0}^{i-1} X_j \cdot b^j$$

1.1.3.1. Sistemas de numeración usuales en informática.

En la representación interna se utiliza un código **binario natural** que es distinto del código de E/S.

También se utilizan los **códigos octal y hexadecimal** como códigos intermedios (los cuales son una simplificación por la que se representan secuencias de ceros y unos abreviadamente, que son más próximos a nuestro sistema decimal, y que permiten traducir rápidamente a y desde binario).

1.1.3.1.1 Representación posicional de los números

Un sistema de numeración posicional en base *b* usa un alfabeto de b símbolos distintos (o cifras), y cada posición tiene un peso especifico. Así, cada número se representará como una secuencia de cifras, contribuyendo cada una de ellas con un valor que dependerá de:

- La cifra en sí.
- La posición de la cifra dentro de la secuencia.

$$N \equiv \ldots n_4\ n_3\ n2\ n_1\ n_0 \cdot n_{-1}\ n_{-2}\ n_{-3}\ n_{-4} \ldots$$

(Número expresado como secuencia de cifras, donde cada n_i pertenece al conjunto de símbolos).

$$N = \ldots + n_4 \times b^4 + n_3 \times b^3 + n_2 \times b^2 + n_1 \times b^1 + n_0 \times b^0 + n_{-1} \times b^{-1} + n_{-2} \times b^{-2} + n_{-3} \times b^{-3} + n_{-4} \times b^{-4} + \ldots$$

(Valor numérico del número N interpretado en base *b*).

Ejemplo.

Supongamos que la base *b* es 10. El conjunto de símbolos será: {0,….,9}. Vemos que el número 3278,52 puede verse como:

$$3 \cdot 10^3 + 2 \cdot 10^2 + 7 \cdot 10^1 + 8 \cdot 10^0 + 5 \cdot 10^{-1} + 2 \cdot 10^{-2}$$

La base 10 es la que estamos acostumbrados a utilizar. Pero puede utilizarse cualquier *b*. Nosotros, en particular, estaremos interesados en las siguientes bases, sobre todo:

- Base 2 (*b*=2): Sistema binario natural. El alfabeto de símbolos será { 0,1 }
- Base 8 (*b*=8): Sistema octal. El alfabeto de símbolos será { 0,...,7 }
- Base 10 (*b*=10): Sistema decimal. El alfabeto de símbolos será { 0,...,9 }
- Base 16 (*b*=16): Sistema hexadecimal. El alfabeto de símbolos será { 0,...,9,A,...,F }

1.1.3.1.2. Sistema de numeración binario

Utilizará la base *b*=2 y, por tanto, el alfabeto de símbolos será { 0,1 }. Veamos a continuación como pasar de binario a decimal y viceversa.

Conversión de binario a decimal

Simplemente aplicaremos la fórmula del Teorema Fundamental de la Numeración, tomando *b*=2.

Ejemplo.

Obtener el valor decimal de N=$111000101{,}0011_2$.[1]

Cogemos la parte entera y de derecha a izquierda se determinan las posiciones:

$$N_{entero} = 1\cdot2^8+1\cdot2^7+1\cdot26+0\cdot2^5+0\cdot2^4+0\cdot2^3+1\cdot2^2+0\cdot2^1+1\cdot2^0 = 45310.$$

$$N_{decimal} = 0\cdot2^{-1}+0\cdot2^{-2}+1\cdot2^{-3}+1\cdot2^{-4} = 0{,}187510.$$

$$N = 453{,}187510.$$

Conversión de decimal a binario

a) **Parte entera** : Simplemente vamos dividiendo por la base el número original, sin decimales (parte entera), y vamos repitiendo el procedimiento para los cocientes que vamos obteniendo.

   Los restos de estas divisiones y el último cociente son las cifras buscadas (observar que siempre deberán estar entre 0 y b-1). El último cociente es el dígito más significativo, y el primer resto el menos significativo.

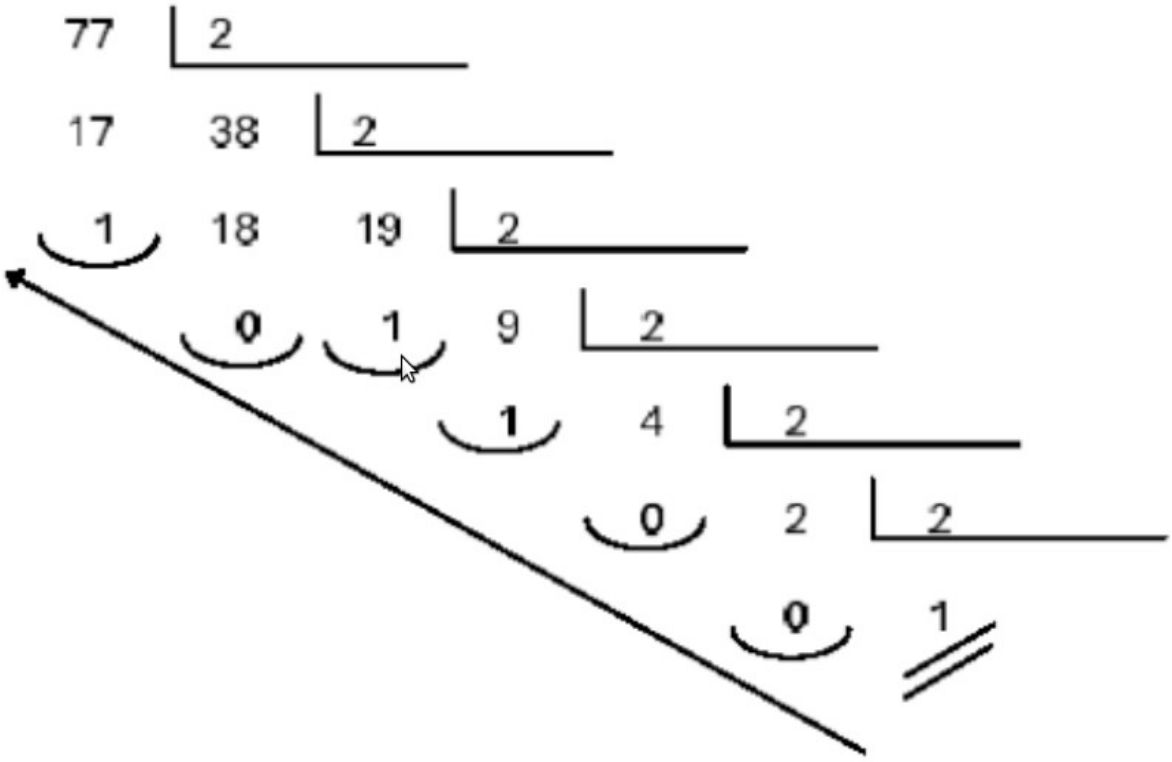
b) **Parte fraccionaria**: Vamos multiplicando por la base la parte fraccionaria del número original, y sucesivamente repetimos el procedimiento con las partes fraccionarias de los números obtenidos. La secuencia de dígitos que vamos obteniendo es la representación en base b buscada de la parte fraccionaria del número.

Ejemplo.

Vamos a pasar el número 77.187510 a base 2 :

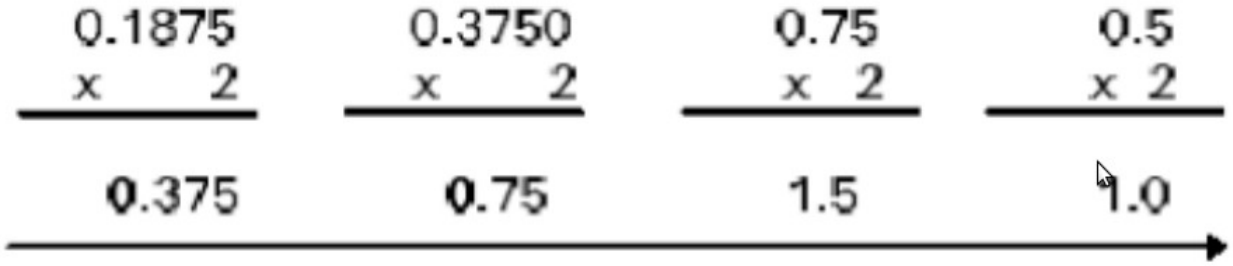
a) Cogemos la parte entera , vamos dividiendo por la base, y tenemos en cuenta los dígitos el resto, así como el último cociente:

1 Se aprecia que el símbolo 2 del subíndice no pertenece al conjunto de símbolos que forma el alfabeto de numeración binario (recordemos que éstos son {0, 1}); en este caso, el 2 indica la base binaria. Esta notación donde la base aparece como subíndice de la secuencia de números que se quiere interpretar es muy habitual.

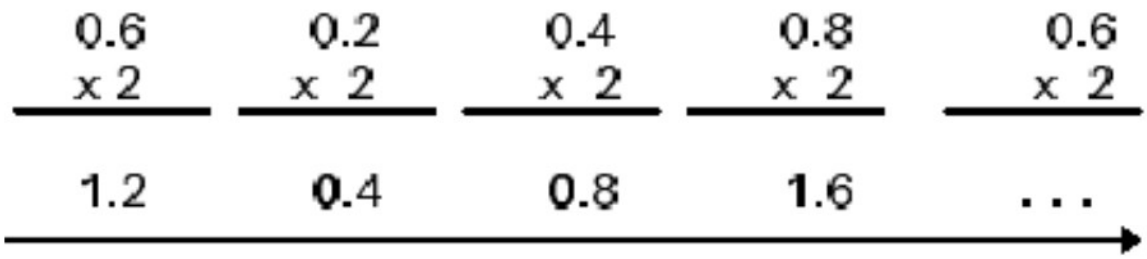


Luego la parte entera seria $\mathbf{N_{entera} = 77_{10}=1001101_2}$

b) Ahora cogemos la parte decimal , la vamos multiplicando por la base y vamos teniendo en cuenta los dígitos enteros resultantes. Para el paso siguiente, obviamente, sólo tendremos en cuenta la parte fraccionaria resultante.



Se puede comprobar fácilmente que un número decimal con cifras fraccionarias puede dar lugar a un número binario con mayor número de cifras detrás de la coma decimal. Incluso infinitas cifras, como se observa en el siguiente ejemplo :



Es decir, $0.6_{10} = 0.1001_2$.(periódico). Vemos, pues, como un número con un sólo dígito de parte fraccionaria en base diez da lugar a un número con infinitos dígitos fraccionarios en base dos. Si un número binario se almacena en un computador con un número prefijado de bits, y por tanto finito, deberíamos recortar el número de cifras con la que lo representemos, a fin de guardarlo en dicho tamaño finito.

El error que se comete al hacer dicho recorte (por ejemplo, almacenar sólo 0.10011001 para representar un 0.6 en base diez, porque sólo reserváramos 8 bits para las partes fraccionarias de los números) se denomina **error de truncamiento**. Con una apariencia o con otra, dependiendo del tipo de representación elegida, estaremos cometiendo errores de este tipo cuando intentemos almacenar valores reales en el computador. En realidad, es un problema análogo al que hay en las calculadoras convencionales, que sólo permiten 8 ó 10 dígitos significativos, por ejemplo, y nos obligan a trabajar con números aproximados.

1.1.3.1.3. Operaciones aritméticas y lógicas con variables binarias

Estas son las tablas correspondientes a las **operaciones aritméticas** básicas:

Suma aritmética	Resta aritmética	Multiplicación	División
0 + 0 = 0	0 - 0 = 0	0 * 0 = 0	0 : 0 = indeterminado
0 + 1 = 1	0 - 1 = 1 (y debo 1)	0 * 1 = 0	0 : 1 = 0
1 + 0 = 1	1 - 0 = 1	1 * 0 = 0	1 : 0 = infinito
1 + 1 = 0 (y llevo 1)	1 - 1 = 0	1 * 1 = 1	1 : 1 = 1

Ejemplos.

```
  1011101 (93)        1011101 (93)        1101010 (106)       1101010 (106)
+ 1000101 (69)      - 1000101 (69)      - 1010111 (87)      x     101 (5)
-------------       -------------       -------------       -------------
 10100010(162)       0011000 (24)        0010011 (19)         1101010
                                                             0000000
                                                            1101010
                                                           ------------
                                                           1000010010 (530)
```

```
1101.01  (13.25)      | 101      (5)
- 101                 ------------
 00110                10.101...  (2.625) (inexacto, puesto que
  -101                                    faltan dígitos.)
  00110
   -101
    001...
```

1.1.3.2. Códigos intermedios y conversiones

Facilitan la labor de programación (trabajar en binario es muy laborioso y puede llevar a errores por cualquier cambio en la secuencia de ceros y unos). Se basan en la facilidad de transformar un número en base dos a otra base mayor que es potencia de dos. De esta manera, cada dígito de la nueva base se traducirá inmediatamente a una secuencia concreta de ceros y unos, pero nos servirá para expresarnos más abreviadamente. Existen dos tipos de códigos intermedios: octal y hexadecimal.

1.1.3.2.1. Código octal

Base octal (b=8) dígitos = {0,1,2,...,7}. Un dígito octal por cada tres binarios.

Conversión de binario a octal

Dado un numero N, en binario, lo dividimos en ternas de tres valores; si el numero de cifras no fuese un múltiplo entero de 3, las cifras que no formen parte de la terna se “rellenarían” con ceros hasta formar una terna, hacia la izquierda, si es en la parte entera (izquierda de la coma), o hacia la derecha, si es en la parte fraccionaria (derecha de la coma).

Ejemplo.

$$N=10110010,11100001_{(2} = \begin{matrix} 010 & 110 & 010 & , & 111 & 000 & 010_{(2} \\ 2 & 6 & 2 & & 7 & 0 & 2 \end{matrix}$$

$$N = 262,702_{(8}$$

Conversión de octal a binario

Se hace en sentido inverso: Cada dígito octal se sustituye por tres binarios.

Ejemplo.

$$N = 376,3_{(8} = 011\ 111\ 110\ ,\ 011\ _{(2}$$

Conversión de octal a decimal

Se hace como se explicó antes, utilizando la fórmula del TFN.

Ejemplo.

$$N = 376,3_{(8} = 3\cdot 8^2 + 7\cdot 8^1 + 6\cdot 8^0 + 3\cdot 8^{-1} = 254,375_{(10}$$

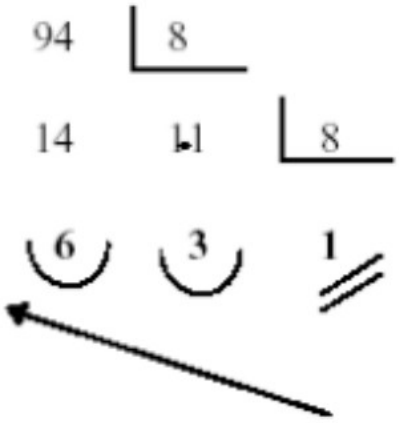
Conversión de decimal a octal

Se hace de la misma manera que cuando pasábamos de decimal a binario pero dividiendo y multiplicando ahora por la base *b*=8.

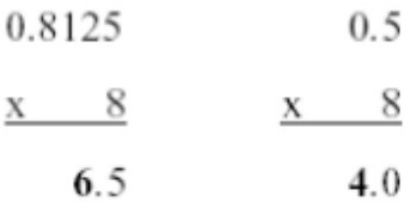
Ejemplo.

$$N=94,8125_{(10}$$

La parte entera es $N_{entera}=94$



La parte decimal seria $N_{decimal}=0,8125$



Luego $N=136,64_{(8}$

1.1.3.2.2. Código hexadecimal

Conversión de binario a hexadecimal

Ahora, en vez de coger una terna, cogemos cuatro cifras tomando como punto de referencia la coma decimal (si la hay).

Ejemplo.

$N = 10110111001101,101011 = 0010\ 1101\ 1100\ 1101\ ,\ 1010\ 1100_{(2}$.

Según la tabla que vemos a continuación, y que se deduce directamente de escribir en binario natural con cuatro dígitos los valores decimales 0 a 15 (o, lo que es lo mismo, en hexadecimal, 0 a F), seria **N = 2DCD,AC$_{(16.}$**

Decimal	Binary	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

De hexadecimal a binario

Seguimos el proceso inverso, sustituyendo cada dígito hexadecimal por sus cuatro binarios correspondientes.

Ejemplo.

$$N=27CB.0A_{(16} = \underset{2}{0010}\ \underset{7}{0111}\ \underset{C}{1100}\ \underset{B}{1011}, \underset{0}{0000}\ \underset{A}{1010}_{(2}$$

De hexadecimal a decimal

Se hace igual que en la base octal, y que en cualquier otra base : Siguiendo la fórmula.

Ejemplo.

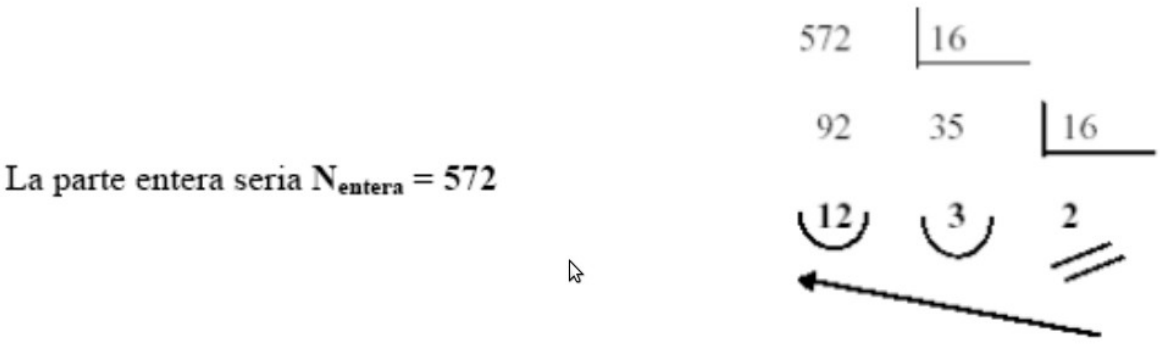
$$N = 23C, A_{16} = 2 \cdot 16^2 + 3 \cdot 16^1 + 12 \cdot 16^0 + 10 \cdot 16^{-1} = 572{,}625_{10}$$ [2]

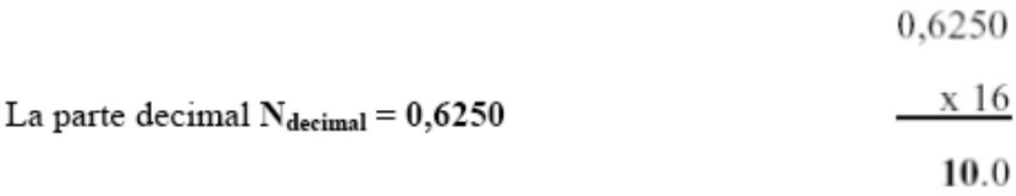
De decimal a hexadecimal

También de modo análogo a como lo hicimos en el octal.

Ejemplo.

N=572,6250.





Es decir, el resultado final en hexadecimal es: $N=23C,A_{(16}$

2 La A y C se sustituyen por su equivalente decimal (en este caso, 10 y 12 respectivamente).

1.1.3.3. Representación de números enteros

Los ordenadores utilizan varios métodos para la representación interna de los números (positivos y negativos), que son:

- MÓDULO Y SIGNO
- COMPLEMENTO A 1 (Ca1, C-1)
- COMPLEMENTO A 2 (Ca2, C-2)
- EXCESO A 2^{N-1}

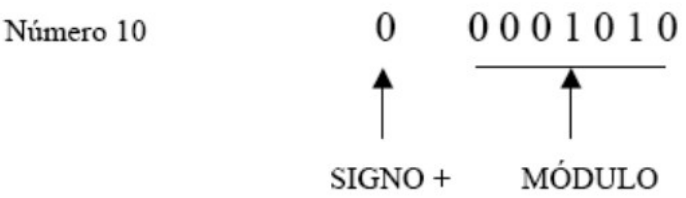
En estas representaciones de números se utiliza el sistema binario y se considera que tenemos un número limitado de bits para casa dato numérico (la cantidad de bits suele coincidir con la longitud de la palabra del ordenador que suele ser de 16, 32 o 64 bits).

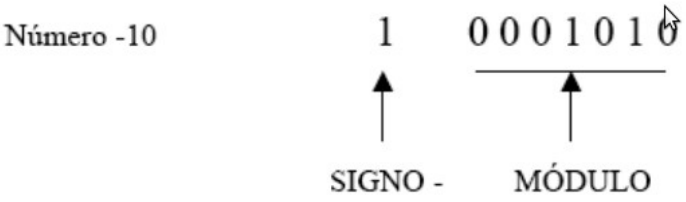
Este número de bits disponibles se representa por n.

1.1.3.3.1. Módulo y signo

En este sistema de representación el bit que está situado más a la izquierda representa el signo, y su valor será 0 para el signo positivo y 1 para el signo negativo. El resto de bits (n-1) representa el módulo del número.

Si tenemos (a nivel didáctico) una palabra de 8 bits (n=8) y queremos representar los números 10 y –10 sería:





1.1.3.3.2. Complemento a 1 (ca1)

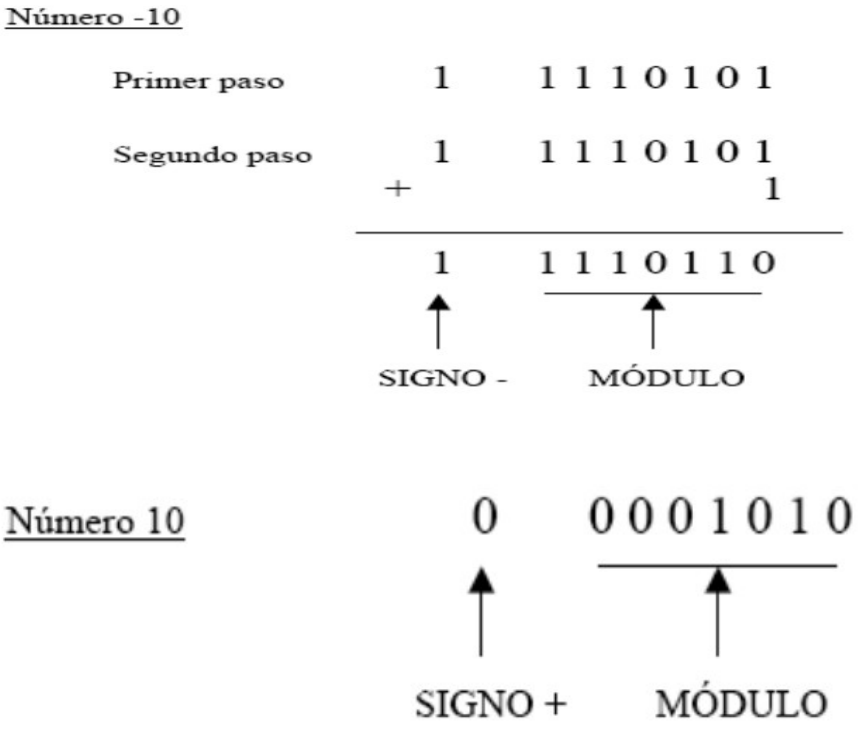
Este sistema de representación utiliza el bit de más a la izquierda para el signo, correspondiendo el 0 para el signo + y el 1 para el signo -. Para los números **positivos**, los n-1 bits de la derecha representan el módulo (igual que en el caso anterior). El **negativo** de un número positivo se obtiene complementando todos sus dígitos (cambiando 0 por 1 y viceversa), incluido el bit de signo.

1.1.3.3.3. Complemento a 2 (Ca2)

Este sistema de representación utiliza el bit más a la izquierda para el signo, correspondiendo el 0 para el signo + y el 1 para el -. En el caso de los números **positivos**, los n-1 bits de la derecha representan el módulo (igual en en los dos casos anteriores). El **negativo** de un número se obtienen en dos pasos:

- Se complementa el número positivo en todos sus bits (cambiando los ceros por 1 y viceversa), incluido el bit de signo, es decir se realiza el complemento a 1.
- Al resultado obtenido anteriormente se le suma 1 (en binario) despreciando el último acarreo si existe.

La representación en C-2 de los números 10 y –10 sería:



1.1.3.3.4. Exceso a 2^{n-1}

Este método de representación no utiliza ningún bit para el signo, con lo cual todos los bits representan un módulo o valor. Este valor se corresponde con el número representado más el exceso, que para n bits viene representado por 2^{n-1}.

Por ejemplo, para 8 bits (n= 8) el exceso es de $2^{8-1} = 2^7 = 128$, con lo cual el número 10 vendrá representado por 10 + 128 = 138 (en binario)

Para el caso del número –10 tendremos –10 + 128 = 118 (en binario)

Número 10: 1 0 0 0 1 0 1 0

Número -10: 0 1 1 1 0 1 1 0

1.1.3.4. Representación de caracteres

Los **caracteres alfanuméricos** se representan a través de unas tablas en las que cada símbolo se le asocia una combinación de bits concreta. Los sistemas de codificación más usados son ASCII y UNICODE.

- **ASCII**: Utiliza 7 u 8 bits para representar los símbolos. Por tanto, podrán representarse 128 o 256 símbolos distintos respectivamente.

ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo
0	0	NUL	16	10	DLE	32	20	(espacio)	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	"	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	'	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	,	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?

ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	\	108	6C	l	124	7C	|
77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	□

- **UNICODE**: Es más moderno y es el código más utilizado hoy en día en la mayoría de sistemas operativos. Permite representar los símbolos de todos los idiomas y países.

1.1.3.5. Representación de imágenes y audio

En el caso de almacenamiento de otro tipo de datos como imágenes, vídeo o audio, se necesita una codificación mucho más compleja, no bastando una correlación símbolo-secuencia de bits.

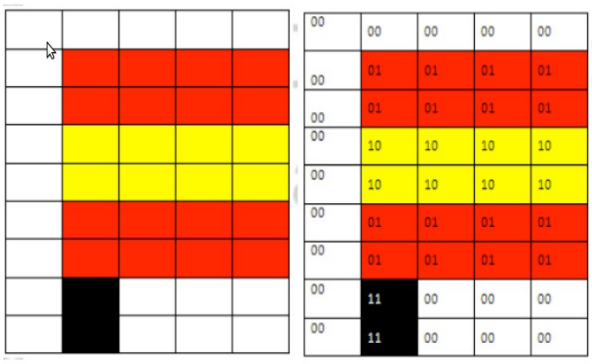
Para las imágenes una forma básica de codificarlas en binario son las llamadas **imágenes rasterizadas, matriciales o de mapa de bits**, en las que se almacena la información de cada píxel (cada uno de los puntos distinguibles en la imagen) descomponiéndolo en tres colores primarios (R o nivel de rojo, G o nivel de verde, B o nivel de azul), con valores de tamaño dependiendo del número de colores que admita la representación, lo que se conoce como **profundidad de color**.

Se establece un código formado por ceros y unos para cada color.

Por ejemplo, si tenemos una profundidad de color de cuatro colores (blanco, rojo, amarillo y negro), podemos establecer este código:

00-Blanco 01-Rojo 10-Amarillo 11-Negro

Hemos empleado dos bits para cada código y la imagen se representará mediante el código del color de cada punto de la imagen de forma ordenada.



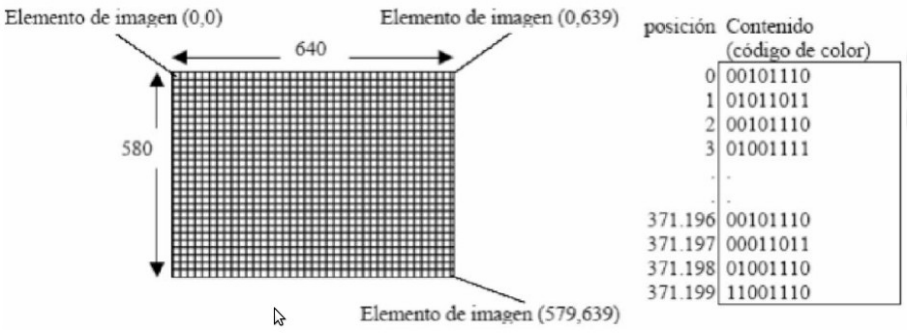
000000000000010101010001010101001010101000101010100001010101000101010100110000000011000000

La cantidad de información que supone la imagen viene dada por:

Tamaño total = bits para cada color x resolución horizontal x resolución vertical.

Para nuestro ejemplo, Tamaño total = 2x5x9 = 90 bits

Naturalmente, en una imagen no sólo se graban los píxeles sino el tamaño de la imagen, el modelo de color, etc. De ahí que representar estos datos sea tan complejo para el ordenador (y tan complejo entenderlo para nosotros).



Existen otro tipo de imágenes llamadas **imágenes vectoriales, vectorizadas o escalables** donde la imagen se construye a partir de vectores, que son objetos formados matemáticamente como segmentos, polígonos, arcos y otras figuras, almacenándose distintos atributos matemáticos de los mismos (por ejemplo, un círculo

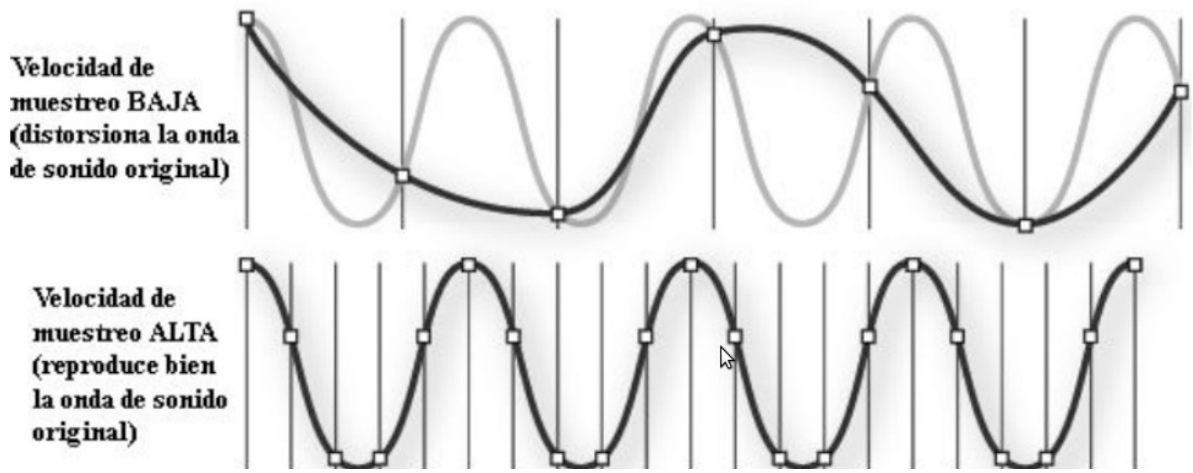
blanco se define por la posición de su centro, el tamaño de su radio, el grosor y color de la línea y el color de relleno.

En el caso del **audio** o **sonido**, que es por naturaleza información analógica o continua, es una onda que transcurre durante un tiempo. Para almacenar ese sonido habrá que representar de alguna forma esa onda para que después se pueda mandar la señal adecuada a dispositivos de salida de audio.

La onda de sonido suele tener un aspecto similar al siguiente, donde el eje horizontal representa el tiempo y el vertical la amplitud del sonido.

Para guardar el sonido de esa onda se toma el valor de la amplitud (altura de la onda) en binario con un número de bits, llamado **calidad del muestreo**, que determinará la calidad del mismo, habitualmente 16 o 32 bits. Esta operación se hace cada cierto tiempo, tomándose un número de puntos por segundo a lo que se llama frecuencia que se mide en hercios (puntos por segundos).

Para reproducir el sonido se reconstruye la onda a partir de los valores almacenados. Es evidente que a mayor frecuencia (más cerca estén los puntos), la onda formada se parecerá más a la original.



Hay que tener en cuenta que la música puede ser **sonido mono**, con un solo canal de sonido o **sonido estéreo** lo que implica dos ondas o dos canales, uno para cada altavoz.

El tamaño del sonido almacenado vendría dado por:

Tamaño = número de canales x calidad del muestreo x frecuencia x duración (s)

Por ejemplo, si tenemos 30 segundos de sonido estéreo con una calidad de 32 bits y con una frecuencia de 22 kHz, el tamaño que ocupará será:

Tamaño = 2 x 32 x 22000 x 30 = 42.240.000 bits = 5.280.000 bytes = 5.156,26 kB

Por último almacenar **video** es bastante sencillo si partimos de la base de que es una representación de imágenes o **frames** y sonido en el tiempo.

Una película no es más que una serie de cuadros desplegados unos tras otros para crear una ilusión de movimiento. El ritmo de imágenes por segundo es una característica del video llamada **frames por segundo** (fps).

Vamos a ver un ejemplo con un video de 30 segundos grabado a una resolución 640x480x32 bits de profundidad de color a 30 fps con sonido estéreo de 32 bits de calidad con frecuencia de 22 kHz ¿cuanto ocupa todo el vídeo?

Empezamos calculando el tamaño de las imágenes:

Tamaño imagen = 640 x 480 x 32 = 9.830.400 bits = 1.200 kB.

El número total de imágenes será: 30 (fps) x 30 (segundos) = 900 imágenes.

La secuencia ocupará: 1.200 kB x 900 (imágenes) = 1.080.000 kB = 1.054,68 MB

Por otro lado calculamos el tamaño del sonido:

Tamaño sonido = 2 (estéreo) x 32 (bits) x 22.000 (Hz) x 30 (segundos) = 42.240.000 bits = 5.280.000 bytes = 5.156,25 kB = 5,03 MB.

El tamaño total del video sería la suma de ambas cantidades:

1.054,68 MB + 5,03 MB = 1.059.71 MB = 1,034 GB.

1.1.3.6. Unidades de medida

Un dígito binario es un bit. Es la unidad de información más pequeña. Pero con un bit podemos almacenar muy poca información, tan solo un 0 un 1. Para almacenar mayor información tendremos que recurrir a otras unidades de medida mayores al bit. Las unidades de medida de almacenamiento se establecen como múltiplos de bits.

Estas unidades son:

- Nibble o cuarteto. 4 bits
- Byte u octeto. 8 bits

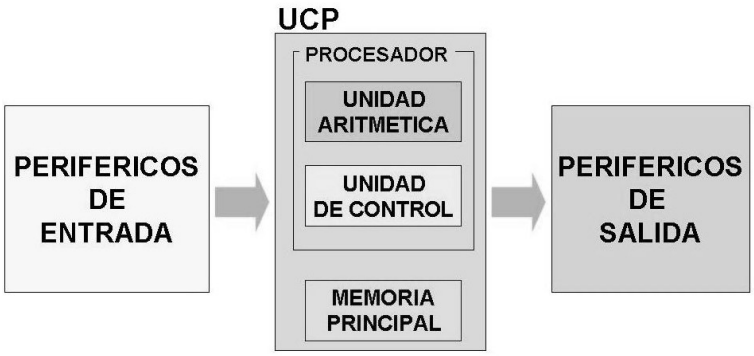
1.2. Estructura funcional de un sistema informático

1.2.1. Arquitectura Von Neumann. Elementos funcionales de un SI

John Von Neumann (1903- 1957) publica en 1946 un artículo que sentó las bases de lo que es un computador hoy en día. Von Neumann propone la **división del ordenador en unidades funcionales**:

- **Unidad Central de Proceso (CPU)**. En ella radica la "inteligencia" de la máquina, es decir, la capacidad de procesar información. Debe funcionar electrónicamente con números binarios y realizar las operaciones de forma **secuencial**, es decir, una tras otra.
- **Memoria**. Es el lugar donde se almacenan las instrucciones y los datos.
- **Dispositivos de entrada y salida**. Son los encargados de tomar datos del mundo exterior y transmitirle el resultado de los procesos realizados.
- **Buses**. Son cables de varios hilos que conectan las unidades funcionales.

A esta estructura se le llama ***arquitectura de Von Neumann*** y aún hoy se sigue usando para la mayoría de los ordenadores.

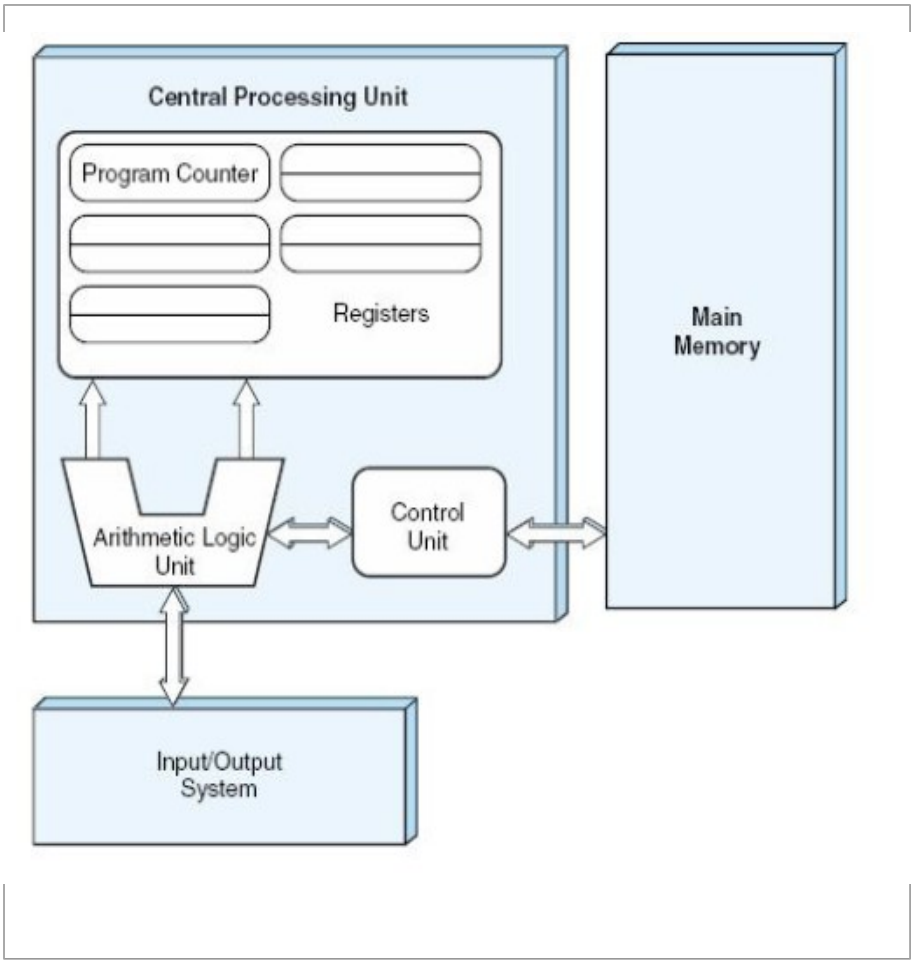


1.2.2. Unidad Central de Proceso

La **Unidad Central de Proceso** (**CPU**), también denominada **procesador**, es el elemento encargado del control y ejecución de las operaciones que se efectúan dentro del ordenador con el fin de realizar el tratamiento automático de la información.

El procesador es la parte fundamental del ordenador; se encarga de controlar todas las tareas y procesos que se realizan dentro de él. Está formado por la **unidad de control (UC)**, la **unidad aritmético-lógica (UAL)** y su **propia memoria interna**, integrada en él. El procesador es la parte que gobierna el ordenador; se encarga de todo: controla los dispositivos periféricos, la memoria, la información que se va a procesar, etc.

Para que el procesador pueda trabajar necesita utilizar la memoria principal o central del ordenador. En la mayoría de los casos, también será necesaria la intervención de la unidad de entrada-salida y de los periféricos de entrada-salida.



Son también elementos característicos de la UCP la **frecuencia del reloj**, la **longitud de palabra** de datos y la **tecnología empleada** (RISC/CISC).

1.2.2.1. Unidad de Control

Es el elemento más importante del microprocesador y se encarga de gobernar el funcionamiento global del mismo. La **unidad de control** recibe, interpreta y ejecuta las instrucciones máquina almacenadas en la memoria principal y genera las señales de control necesarias para ejecutarlas.

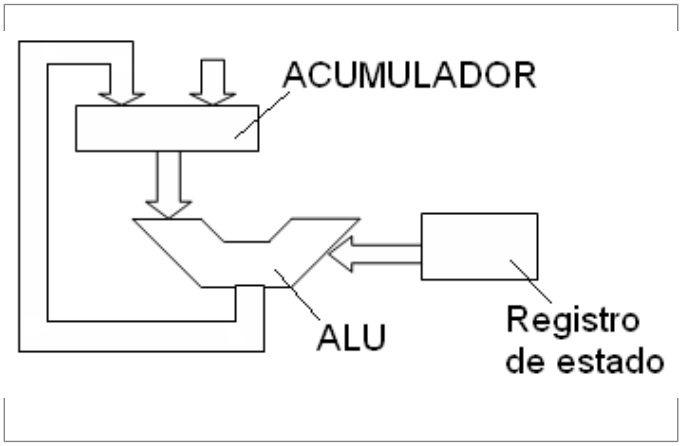
Existen dos posibilidades de unidad de control:

- **UC microprogramada**. Las instrucciones están implementadas mediante una memoria de control que contiene microprogramas asociadas a microinstrucciones en las que se descomponen. Mediante un elemento llamado **descodificador** transformará la información de este registro en datos comprensibles por otro componente llamado **secuenciador** encargado de analizar e interpretar la salida del descodificador, y, según su valor, ejecutará un microprograma contenido en la memoria de control, que cuenta con las microinstrucciones necesarias para que se ejecute la instrucción. Son las más extendidas.
- **UC cableada**. Las instrucciones están implementadas por hardware. Al finalizar la ejecución de una instrucción el registro contador de programa (CP) contiene información sobre la dirección de memoria donde se encuentra la siguiente instrucción a ejecutar. La circuitería de la UC lee de este registro CP la instrucción siguiente.

1.2.2.2. Unidad Aritmético-Lógica

La **unidad aritmético-lógica** (ALU, *Arithmetic and Logical Unit*) recibe los datos sobre los que efectúa operaciones aritméticas (suma, multiplicación…) y lógicas (AND, OR, NOR…) y devuelve luego el resultado, todo ello bajo la supervisión de la unidad de control.

Estas operaciones son ordenadas por las instrucciones que se están ejecutando, con las informaciones presentes en los registros de entrada y/o en posiciones determinadas de la memoria central y devuelve los resultados a la memoria central.



1.2.2.3. Registros

Los **registros** son celdas de memoria de alta velocidad que permiten almacenar datos de forma temporal mientras se ejecuta alguna instrucción. Constituyen la memoria interna de la CPU. Están formados por conjuntos de bits que se manipulan en bloque. El número de bits varía dependiendo de la CPU, pero siempre son múltiplos de ocho (8, 16,32,…). Cuanto mayor sea el tamaño de los registros, más potente será una CPU.

1.2.2.3.1. Registros visibles al usuario

Estos registros son aquellos que pueden ser referenciados por lenguaje ensamblador o máquina con el fin de optimizar el uso de los recursos. Se distinguen:

- **Registros de dirección**. Contienen las direcciones de memoria donde se encuentran los datos. Algunos de los registros de dirección más usados son los registros índices y los punteros de pila.
- **Registros de datos**. Se usan para contener datos. Su utilización aumenta la velocidad de proceso, ya que cuando es dato es solicitado, si se encuentra en uno de estos registros no es necesario acceder a memoria.
- **Registros de condición** o ***flags***. Son bits que cambian de estado (0-1) como consecuencia del resultado de la última operación efectuada, indicando, por ejemplo, si una operación entrega un resultado negativo.

1.2.2.3.2. Registros de control y de estado

Son los que intervienen en la ejecución de instrucciones.

- **Registro de programa, CP** (PC, P*rogram Counter*), o **contador de instrucciones**. Contiene la dirección de la siguiente instrucción a ejecutar. La CPU actualiza su valor después de capturar una instrucción.
- **Registro de instrucción, RI** (IR, *Instruction Register*). Contiene el código de la instrucción actual.
- **Registro de dirección de memoria, RDM** o **RDIR** (MAR, *Memory Address Register*). Contiene la dirección de una posición de memoria donde se encuentra o va a ser almacenada la información. Se utiliza a través del bus de direcciones.
- **Registro de intercambio de memoria, RIM,** o **registro de datos, RDAT** (MBR, *Memory Buffer Register*). Envía o recibe el contenido de la dirección apuntada por el registro de dirección de memoria. El intercambio se produce a través del bus de datos.

1.2.2.4. Reloj

Para que el microprocesador genere todas las señales necesarias para controlar los restantes bloques del sistema y para que todo el sistema funcione en sincronía, se parte de ondas cuadradas de frecuencia constante generadas por un cristal de cuarzo. La *frecuencia* de este **reloj** se mide en MHz y determina la *velocidad de funcionamiento y proceso* del microprocesador y por tanto, de todo el sistema. El periodo de esta señal de reloj (medido en segundos) se denomina *ciclo de reloj*.

A partir de esta señal de reloj, el microprocesador realiza una serie de ciclos de trabajo denominados *ciclo máquina*. Este ciclo máquina está formado por dos fases:

- **Fase de búsqueda**: se realiza la búsqueda de una instrucción en memoria y la guarda en el registro correspondiente.
- **Fase de ejecución**: se ejecuta o realiza la transferencia de datos ordenada.

Este ciclo máquina se tarda en realizar más o menos tiempo dependiendo del fabricante. Por otra parte, para completar la ejecución completa de una instrucción se requieren más de un ciclo máquina. El número de ciclos máquina necesarios para procesar por completo una instrucción se denomina *ciclo de instrucción.*

1.2.2.5. Ejecución de instrucciones

Toda instrucción-máquina residente en memoria principal pasa por una serie de **fases** que van desde su captura a su interpretación y ejecución. Éstas son:

- **Carga, búsqueda o lectura (fetch).** La UC envía a la memoria principal la dirección de la instrucción a ejecutar, que está almacenada en el registro contador de programa (PC) y activa las señales de control necesarias para que ésta le entregue la mencionada instrucción.
- **Decodificación**. La UC recibe la instrucción, la analiza y, en su caso, lee los operandos de la memoria principal, enviando su dirección y activando las correspondientes señales de control.
- **Ejecución**. La UAL, bajo las órdenes de la UC, realiza la operación sobre los operandos, y, si es necesario, se graba el resultado en la memoria principal o en un registro.
- **Incremento del contador de programa** (PC). También denominado puntero **de instrucción** (IP), con lo que se puede pasar a ejecutar la instrucción siguiente (aunque existen instrucciones que pueden modificar el contenido del PC, dando lugar a bifurcaciones). Hay un tipo de bifurcaciones no programadas por causa de interrupciones externas e internas, las llamadas *traps*.

	DESCRIPCIÓN DE LA OPERACIÓN	OPERACIONES CON REGISTROS
1	Llevar el **contenido del CP** al bus de direcciones.	RD := CP
2	Introducir el contenido del bus de direcciones en el **registro de dirección** de memoria para descodificar la posición de la instrucción a ejecutar.	
3	Hacer una **lectura en memoria** y cargar la instrucción.	RM:=M[RD]
4	Llevar el contenido del registro de memoria al bus de memoria.	RI:=RM
5	El contenido del bus de memoria se introduce en el registro de instrucciones.	
6	Se incrementa el contador de programa quedando preparado para la ejecución siguiente.	CP:=CP+1
7	A continuación se hace una lectura en memoria del operando implicado (pasos 8-10)	
8	El contenido de la dirección del operando pasa al bus de direcciones.	RD:= RI.CDO
9	El contenido del bus de direcciones se debe introducir en el registro de dirección de memoria para descodificar la posición del operando implicado en la suma.	
10	Lectura del operando direccionado	RM:= M[RD]
11	El contenido del registro de memoria se debe llevar al **bus de memoria**	RT:= RM
12	El contenido del bus de memoria se debe pasar al **registro temporal** para que acceda a la ALU.	
13	El contenido del acumulador debe entrar en la ALU para sumarse al operando llegado de memoria.	AC:= AC+ RT
14	Se debe indicar a la ALU la **operación a realizar** mediante una señal (señal suma)	
15	El resultado obtenido se debe guardar en el acumulador.	

1.2.2.5.1. El Juego de Instrucciones.

La forma de representación de una instrucción para ser almacenada en memoria se denomina formato de instrucción. Dicho formato especifica el significado de cada uno de los bits que constituyen la instrucción, denominándose longitud del formato al número de bits que la componen.

Para simplificar su decodificación la instrucción se divide en una serie de campos (cadenas de bits contiguos), estando referido cada campo a un tipo de información específico.

El *tipo de información* que debe contener una instrucción es la siguiente:

- Operación.
- Dirección de los operandos.
- Dirección del resultado.
- Dirección de la siguiente instrucción.
- Tipos de representación de operandos.

Existen muchos **tipos de instrucciones** según el tipo de operación a llevar a cabo: de transferencia de información, aritmético-lógicas, de desplazamientos, de transferencias de control (saltos condicionales, bifurcaciones, llamadas y retornos de procedimientos, etc.).

Cada instrucción se suele identificar con un **nemotécnico** que hace referencia a la función que realiza la instrucción (move, store, load, clear, set, push, pop, etc.)

También es interesante conocer las distintas formas que tienen las instrucciones de direccionar a los operandos que en ellas se hacen referencia, los tipos o **modos de direccionamiento**.

1.2.2.6. Arquitecturas de procesador RISC y CISC

La arquitectura **CISC** (C*omplex Instruction Set Computer*), que ya se daba en los primeros diseños de UCP, se caracterizaba por disponer de un **grupo amplio de instrucciones complejas y potentes**. El ordenador era más potente a medida que era más amplio su repertorio de instrucciones.

Toman como principio la *microprogramación*, que significa que cada instrucción de máquina es interpretada, empleando un microprograma localizado en una memoria en el circuito integrado del procesador. Las instrucciones compuestas son codificadas internamente y ejecutadas con una serie de microinstrucciones que se almacenan en una memoria de control.

Esto era efectivo y muy práctico porque al principio la memoria principal era más lenta que la UCP y el tiempo de una instrucción podía ser de varios ciclos de reloj ya que cuando una instrucción era procesada en un único ciclo de reloj, no se podía continuar con la siguiente instrucción inmediatamente ya que todavía no estaba lista (al ser la memoria principal mucho más lenta que la de control).

A finales de los setenta, al aumentar las prestaciones de la memoria principal la consecuencia inmediata fue que ya no tenía que esperar la UC a ésta, lo que permitió trabajar con instrucciones mucho más simples que se completasen en un ciclo de reloj y acelerando la ejecución de instrucciones.

Esta arquitectura es conocida como **RISC** (R*educed Instruction Set Computer*) y está formada por un **juego de instrucciones lo más reducido posible**, la mayoría completadas en un ciclo de reloj.

ARQUITECTURA CISC (pocas instrucciones complejas)	ARQUITECTURA RISC (muchas instrucciones pequeñas)
- Formatos de instrucción de varios tamaños	- Formatos de instrucción de pocos tamaños
- Interpreta microinstrucciones	- Interpreta microoperaciones
- Muchos modos de direccionamiento	- Pocos modos de direccionamiento
- Pocos registros de propósito general	- Muchos registros de propósito general
- Repertorio de instrucciones flexible	- Repertorio de instrucciones rígido.
- Son lentas. Ejecución por software	- Son rápidas. Ejecución directa por Hardware
- UC microprogramada.	- UC cableada.

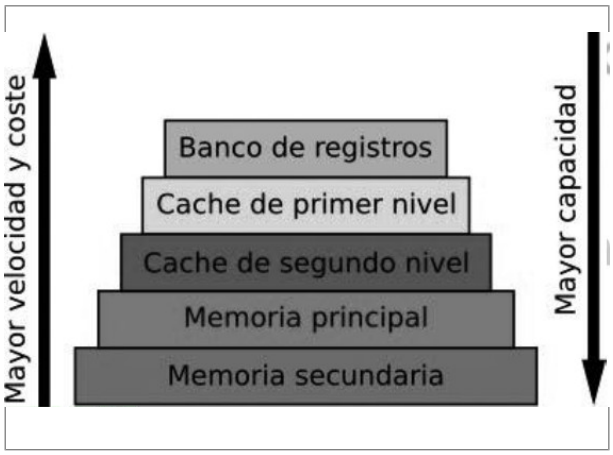
Las CPU actuales combinan elementos de ambas y no son fáciles de encasillar ya que desde hace tiempo se ha empezado a investigar sobre microprocesadores "híbridos", con el propósito de obtener ventajas procedentes de ambas tecnologías.

1.2.3. La memoria. Funciones y tipos

En el modelo de arquitectura de Von Neumann, el término memoria hace referencia a un espacio de almacenamiento donde depositar instrucciones, datos y resultados de los programas[3]. No existen ningún medio de almacenamiento de uso práctico universal y todas las formas de almacenamiento tienen sus desventajas. Por tanto, un sistema informático contiene varios tipos de almacenamiento, cada uno con su propósito individual.

En base a las características de las memorias (el coste por bit, el tiempo de acceso y la capacidad o tamaño), se establece lo que se ha dado en llamar una **jerarquía de memorias**.

3 En la arquitectura Hardvard, sin embargo, se mantienen dos espacios de memoria separados: uno destinado a las instrucciones y otro destinado a los datos.



En general se puede hablar de dos tipos de memoria: **primaria** (o interna) y **secundaria** (o externa).

1.2.3.1. Memoria primaria

La memoria primaria está directamente conectada a la CPU del ordenador. Debe estar presente para que la CPU funcione correctamente. La memoria primaria es más rápida que la secundaria.

Existen tres tipos de memoria primaria utilizados en la actualidad:

- **Registros del procesador**. Están situados en el interior de la CPU, tal como se vio en el apartado correspondiente. Contienen la información con la que trabaja la unidad aritmético-lógica. Técnicamente son el tipo de almacenamiento más rápido de la memoria primaria, pero la cantidad de datos que pueden almacenar es muy limitada.
- **Memoria caché**. Es un tipo especial de memoria primaria usada por la mayoría de las CPU para mejorar su eficiencia o rendimiento. Comparada con los registros, la caché es ligeramente más lenta pero de mayor capacidad. Sin embargo, es más rápida pero de mucha menor capacidad que la memoria principal. En la actualidad se utiliza la memoria caché multi-nivel, formada por dos tipos (en algunos procesadores hay hasta tres tipos) de caché: la caché primaria, más pequeña, rápida y cercana a la CPU (normalmente está integrada dentro del procesador); y la caché secundaria, más grande y lenta que la primaria (normalmente está integrada en un chip en la placa base), aunque todavía más rápida y mucho más pequeña que la memoria principal. Su funcionamiento es el siguiente: parte de la información de la memoria principal (que es más lenta) se duplica en la memoria caché; los accesos siguientes de la CPU se realizan a dicha copia, haciendo que el tiempo de acceso medio al dato sea menor; cuando se necesita un dato que no está en la caché, se copia la parte necesaria de la memoria principal, desechando los datos más antiguos almacenados en la caché.

  Existen tres tipos diferentes de memoria caché para procesadores:

- **Caché de Primer nivel o L1**: Integrada en el núcleo del procesador, trabajando a la misma velocidad que éste. Su tamaño varía de un procesador a otro y suele estar dividida en dos partes dedicadas, una para instrucciones y otra para datos.
- **Caché de Segundo nivel o L2**: Integrada también en el procesador, aunque no directamente en el núcleo, tiene las mismas ventajas que la caché L1, aunque es algo más lenta que ésta. La caché L2 suele ser mayor que la caché L1. A diferencia de la caché L1, esta no está dividida, y su utilización está más encaminada a programas que el sistema.
- **Caché de Tercer nivel o L3**: Es un tipo de memoria caché más lenta que la L2, muy poco utilizada en la actualidad, incorporada a la placa base.

- **Memoria principal**. Contiene los programas en ejecución y los datos con que operan dichos programas. La CPU puede transferir información muy rápidamente entre un registro del procesador y la memoria principal a través de la memoria caché. En las computadoras modernas se usan memorias de acceso aleatorio (RAM – Random Access Memory), conectadas a la CPU a través del Front Side Bus y el puente norte.

1.2.3.2. Memoria secundaria

El problema de la memoria primaria es que es volátil, es decir, que la información se pierde cuando se desconecta el ordenador. Por ello, son necesarios otros dispositivos de almacenamiento para guardar la información a largo plazo. Estos dispositivos de almacenamiento se engloban en la categoría de memoria secundaria. Algunos de estos dispositivos son los discos duros, CD, DVD y memorias FLASH. La memoria secundaria es de mucha mayor capacidad que la memoria primaria, pero también es mucho más lenta. En los ordenadores modernos, los discos duros suelen usarse como los principales dispositivos de almacenamiento masivo. El tiempo necesario para acceder a un byte almacenado en un disco duro es de unas pocas milésimas de segundo, mientras que el tiempo que lleva acceder al mismo byte en una memoria RAM es de algunas milmillonésimas de segundo (nanosegundos). Los discos duros son del orden de un millón de veces más lentos que la memoria.

Se puede llevar a cabo una **clasificación de este tipo de memorias** según diferentes criterios.

- Según la **tecnología empleada**:
  - **Tecnología magnética**. Emplean sustrato de plástico o aluminio cubierto de material magnetizable (óxido férrico o de cromo). La información se graba en celdas que forman líneas o pistas. Cada celda puede estar sin magnetizar o magnetizada con dos posibles valores: norte (0) y sur (1).
  - **Tecnología óptica**. Usan energía lumínica (como el rayo láser) para almacenar o leer información. Los ceros o uno se representan por la presencia o ausencia de señal luminosa.
  - **Tecnología Magneto-Óptica**. Los soportes originarios poseen una magnetización previa (norte o sur) y mediante láser es posible cambiar la magnetización de las celdas.

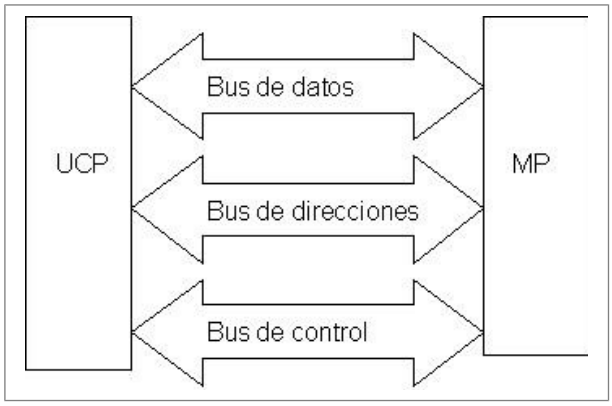
- **Tecnología Flash-USB**. Usan memorias semiconductoras de tipo flash nand, con la característica de que no necesitan refresco al usar tecnología de puerta flotante.

- Según el **tipo de operaciones** que se pueda realizar sobre los mismos:
  - **Reutilizables** (cinta magnética, disco cd-rw, etc).
  - **No reutilizables (disco cd-rom, etc).**
- Según la **forma de acceder** a la información:
  - **Secuencial** (cinta magnética)
  - **Directo** (cd-rom, disco duro).
- Según la **ubicación física de dicha unidad**:
  - **Interna** (disco duro, unidad flexible).
  - **Externa** (memoria USB, discos duros externos)
- Según la relación existente entre el soporte de almacenamiento y el elemento que lleva a cabo la lectura escritura:
  - **Removibles** (discos flexibles)
  - **No removibles** (discos duros)

1.2.4. Buses: arquitecturas y funcionamiento

La interconexión de todas las unidades estudiadas se lleva a cabo a través de una serie de canales de conexión denominados **buses** que, físicamente, son un conjunto de líneas por las que se transmite la información binaria (sea de una instrucción, un dato, o una dirección, en un instante dado).

Se denomina **ancho de bus** al tamaño de ese número de hilos o bits que se transmiten simultáneamente por uno de esos canales.



Se pueden distinguir tres tipos de buses:

- **Bus de datos (bidireccional).** Transporta datos procedentes o con destino a la memoria principal y las unidades de entrada-salida. Cabe destacar cómo la velocidad de este bus en su conexión con la memoria RAM es un factor determinante en el rendimiento del sistema.

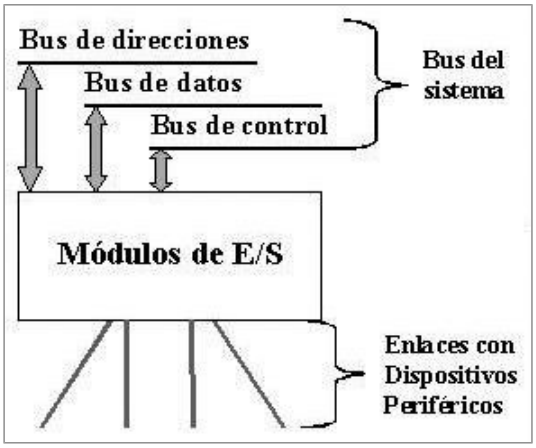
- **Bus de direcciones (unidireccional)**. Transporta las direcciones de la unidad de control a la memoria principal o a los periféricos.
- **Bus de control (bidireccional).** Transporta las señales de control (microórdenes) generadas por la unidad de control.

1.2.5. Unidad de E/S. Controladores y periféricos

Un ordenador tendría una utilidad nula sin la presencia de algún medio que permitiese realizar las entradas y salidas de datos para poder interactuar con el medio. El concepto de entrada y salida hace referencia a toda comunicación o intercambio de información entre la CPU o la memoria central con el exterior.

Estas operaciones se suelen llevar a cabo a través de una cada vez más amplia gama de dispositivos externos llamados **periféricos** que proporcionan al ordenador las vías para intercambiar datos con el exterior.

La parte del equipo que permite esta comunicación es la **unidad de entrada-salida**.



Básicamente estos módulos o canales se usan para resolver las diferencias (velocidad de transmisión, formato de datos, etc.) que pueden existir entre el procesador y dichos periféricos. Sus funciones fundamentales son direccionamiento, transferencia y sincronización.

Existen distintas arquitecturas de ordenador **según el direccionamiento** de los dispositivos de entrada-salida.

- Buses separados de memoria y entrada-salida.
- Entrada-salida mapeada en memoria o máquina de bus único. Los puertos de entrada-salida se tratan como si fuesen direcciones de memoria.

También existen distintas arquitecturas de ordenador según como se establece el **control del tránsito de datos**:

- **Entrada-salida controlada por programa**. Mediante la ejecución de unas instrucciones especiales si es de bus separado (in, out, etc.) e instrucciones de almacenamiento si es entrada-salida mapeada a memoria. Usada en periféricos con velocidades menores que la CPU.
- **Entrada-salida por acceso directo a memoria**. Todas las funciones se implementan mediante un circuito controlador llamado **controlador de DMA** (direct access memory).

Por último, también existen diferentes arquitecturas según el modo en que se produce la sincronización de la CPU con los periféricos:

- **Entrada-salida con sincronización por sondeo y selección**. La CPU hace encuesta o consulta a los dispositivos de su situación y los va atendiendo.
- **Entrada-salida con sincronización por interrupciones**. Los dispositivos son los que interrumpen la ejecución del programa en CPU cuando están en disposición de realizar una operación de entrada-salida.