

TEMA 1. Informática básica

Introducción

El Tema 1 nos servirá para dos fines. En la sección 1.1 Informática básica, realizaremos una breve presentación de la noción de "Informática". Veremos que la misma es muy amplia y que comprende muchos elementos distintos. Enunciaremos algunos de ellos, presentaremos los que se tratarán en secciones posteriores, y descartaremos otros por no pertenecer al ámbito de la asignatura en la que nos encontramos. También definiremos la noción de "Sistemas Informáticos", que da nombre a nuestra asignatura, y presentaremos una breve descripción de los contenidos de este curso.

En la parte restante del Tema 1 (secciones 1.2, 1.3, 1.4 y 1.5) nos centraremos ya en un tipo concreto de sistema informático, el ordenador, sobre el cual ilustraremos alguna de sus principales características. Hay que tener en cuenta que ésta no es una asignatura de "Estructura de Computadores", sino más bien de "Sistemas Operativos", y por lo tanto no entraremos en un nivel muy elevado de detalles. Sin embargo, sí que creemos necesario mencionar los "Fundamentos Estructurales y de Funcionamiento" (sección 1.2), donde nos centraremos en la codificación de la información en código binario o hexadecimal, destacando también los "Componentes básicos de un ordenador" (sección 1.3), "El Disco Duro" (sección 1.4) y "El sistema de arranque de un ordenador. Particiones y Volúmenes" (sección 1.5).

1.1 Informática básica

Podemos definir Informática, siguiendo el diccionario de la RAE, como el "conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento automático de la información por medio de ordenadores". De una forma más sintetizada, podemos identificar la Informática con "información automática".

Por lo tanto, un "Sistema Informático" lo podemos entender como una solución completa a un problema concreto de tratamiento automatizado de información.

Esta solución, por lo general, contempla una parte física (el "hardware" o máquinas sobre las que se trabaja) y una parte "software" (formada por los programas o utilidades de que se hace uso en la máquina). Por último, los sistemas informáticos suelen

requerir por lo general de una interacción humana, que se suele identificar con la que demanda unos ciertos servicios.

Ejemplos de sistemas informáticos, por lo tanto, serían el usuario que en su ordenador personal trata de redactar un texto o leer el correo electrónico, alguien que en su teléfono móvil apunta un número en su agenda, o un operario de una fábrica que debe controlar ciertas máquinas a través de un programa informático; o podemos también pensar en sistemas mucho más sencillos como un programador de un termostato o de un electrodoméstico cualquiera.

Nuestra asignatura no pretende entrar en el diseño y definición de sistemas informáticos, puntos que se verán más adelante durante la titulación, sino como asignatura de iniciación a los sistemas informáticos, introducirnos en el conocimiento y uso de los elementos básicos en todo sistema.

Sólo a modo de ejemplo, podemos observar que el diseño completo de un sistema informático requiere la definición de elementos de muy diferente naturaleza, como pueden ser:

- Equipos: Ordenadores, periféricos, sistemas de comunicación (a este punto dedicaremos la sección 1.2).
- Procedimientos: Sistemas Operativos (a este punto dedicaremos el tema 3), aplicaciones de gestión, ofimática, sistemas de comunicación, documentación, seguridad (copias, acceso, protección).
- Personal: necesidades de personal, especialización, procedimientos de usuario, formación (inicial y continuada).
- Otros: estudio económico (coste inicial, mantenimiento, de personal, vida del sistema).

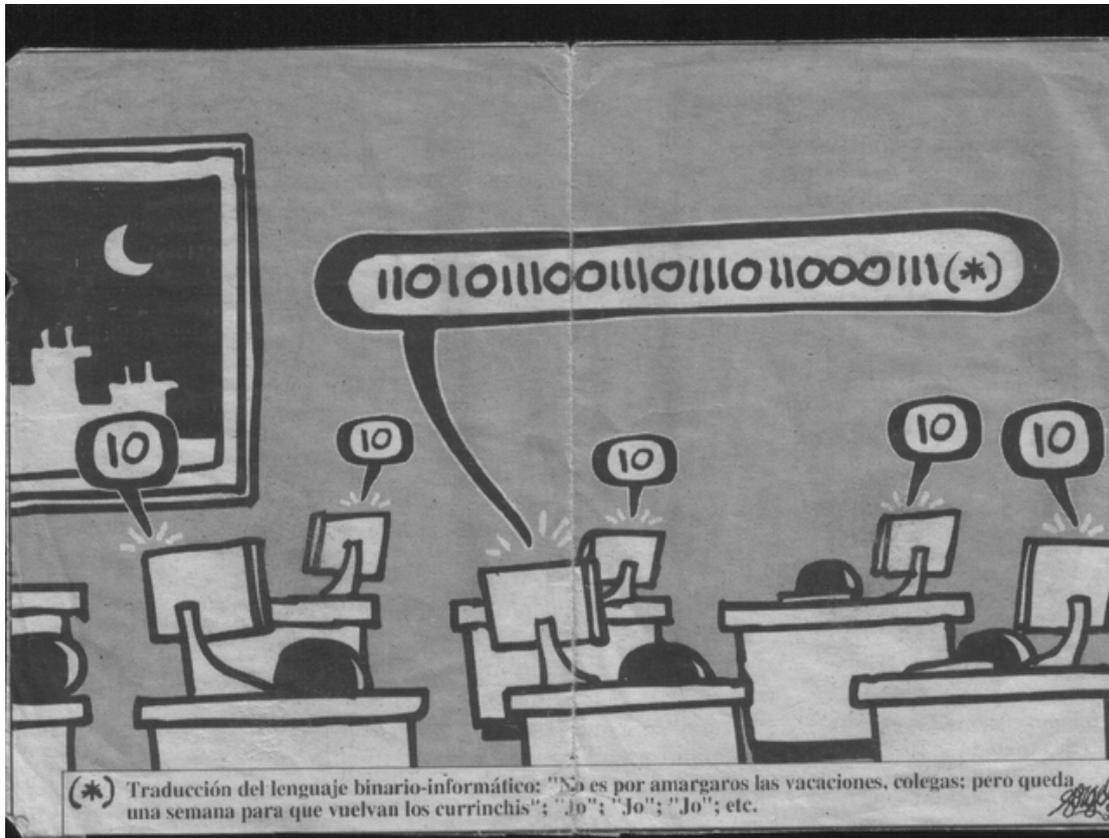
Entre estos distintos componentes se pueden encontrar cuestiones de hardware, de software, lenguajes de programación, bases de datos, sistemas de comunicación, análisis y diseño de sistemas, estrategias de formación...

La creación de sistemas complejos normalmente no se lleva a cabo por una única persona, sino por un equipo de personas, cada una de ellas especializada en una determinada área.

1.2 El ordenador. Fundamentos estructurales y de funcionamiento

A partir de la definición hecha de Informática como "información automática", nuestro primer objeto de interés va a ser conocer cómo la información es almacenada en un sistema informático. Para ello

explicaremos la diferencia entre señales analógicas y señales digitales. Posteriormente, presentaremos la estructura básica de un ordenador, con el fin de saber cómo la información almacenada en un dispositivo puede ser procesada.



1.2.1 Almacenamiento de información: señal digital.

Los ordenadores actuales se basan en una tecnología electrónica digital, en comparación a otros dispositivos que utilizan señal analógica.

Podemos identificar la idea de digital con "discreto": en un sistema digital las variables de entrada y salida son magnitudes (en general señales eléctricas) discretas o que se toman como tales, y se procesan como valores discretos. Es decir, sólo un número entero y concreto de valores son posibles.

Por el contrario, podemos identificar la idea de analógico con "continuo": en un sistema analógico sus variables de entrada y salida son magnitudes (en general, señales eléctricas) continuas y se procesan como valores continuos, es decir, pueden alcanzar cualquier valor dentro de un espectro determinado.

Un ejemplo de señal analógica sería la fotografía tradicional con película. La luz es captada e impresa sobre la película de forma continua, sin distinguir regiones o cuadros. Sin embargo, las cámaras fotográficas en la actualidad son en su mayoría digitales, ya que la señal se ha discretizado. Por ejemplo,

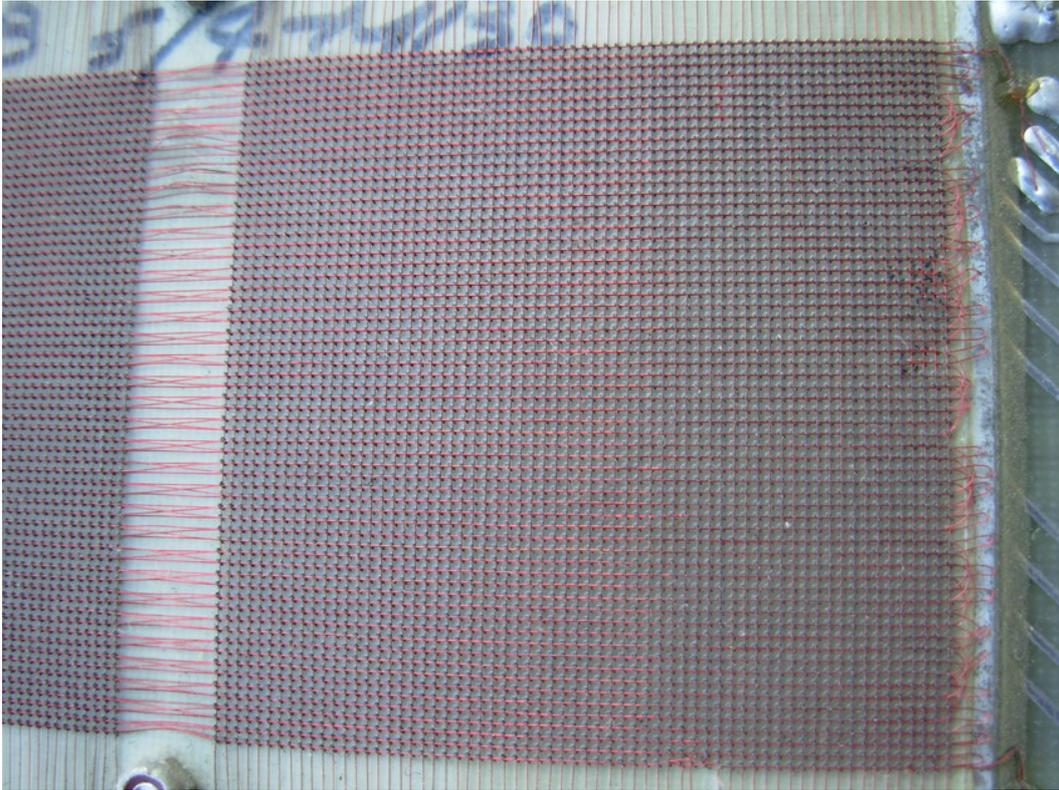
en una resolución de 14 megapíxeles la señal se ha discretizado en una matriz de 4592 * 3056 píxeles, con cada píxel capturando una intensidad de rojo, verde y azul (RGB) correspondiente. Si aumentáramos el tamaño de una foto cualquiera, veríamos como la misma está "discretizada" o "pixelada". La captura de la fotografía se basa en los impulsos electromagnéticos que produce el objeto fotografiado sobre el sensor de la cámara (que es el encargado de grabar la misma para cada píxel). Otro ejemplo similar sería la señal analógica (continua) que se graba en las cintas de audio o discos de vinilo, que en la actualidad se han convertido en discos compactos o "mp3" donde la señal de nuevo está "discretizada".

La informática por lo general se basa en la utilización de señales digitales. La tecnología actual permite construir con cierta facilidad todo tipo de dispositivos discretos que distinguen entre dos estados diferentes (los cuales etiquetaremos como "0" y "1"). Así pues, nuestra tecnología digital gestiona entradas y salidas y las procesa admitiendo únicamente dos valores diferentes (aún así, el concepto genérico de digital se refiere a un número finito que puede ser más de dos). Desde el punto de vista teórico, el hecho de admitir más de dos estados en un sistema digital haría inmensamente más eficiente y potente dicho sistema (por ejemplo, los sistemas cuánticos se basan en el control de más de dos estados de cada uno de sus elementos, pero por el momento siguen siendo dispositivos teóricos en los que a nivel práctico, el control de varios estados resulta muy costoso), pero por el momento nos vemos limitados a trabajar con bits (binary digits) que sólo aceptan 2 estados (podemos pensar en ellos como "encendido" y "apagado").

Por este motivo toda la información que se almacena y maneja en los ordenadores es información codificada en formato binario. Podemos pensar en la información que tenemos almacenada en una memoria digital. Esa memoria está compuesta por un gran conjunto de elementos (podemos pensar en casilla o celdas) que sólo son capaces de distinguir dos estados diferentes. Cualquier información que queramos almacenar en ella habrá que codificarla de alguna manera a base de cadenas de dichos dos estados. Toda la información, números, texto, colores, sonido, imagen... hay que codificarla a una secuencia de dos estados distintos que identificamos con 0 y 1.

Llamaremos bit (binary digit) a la unidad básica (y mínima) de información en un sistema digital (y también en informática). Un bit puede representar un 0 ó un 1.

En la imagen inferior podemos observar una memoria de una computadora (del año 1980) en la cual el tamaño de los bits se puede comprobar a simple vista. La memoria contiene 4096 celdas o bits, y su tamaño es aproximadamente 4 * 4 cm.



Semejante fragmento de memoria nos serviría para almacenar 2 elevado a 4096 valores. Si bien puede parecer un número muy elevado, en un sistema con bytes (explicamos el significado de "byte" en el siguiente párrafo) de 8 bits nos permitiría almacenar únicamente 512 caracteres.

En general un bit es un fragmento de memoria tan pequeño (sólo permite almacenar dos valores) que cualquier sistema informático utilizará fragmentos de memoria un poco mayores, o agrupaciones de bits, que le permitan almacenar cantidades de información relevantes. A dichos fragmentos les llamaremos bytes.

Un byte es una secuencia de bits contiguos. Su tamaño depende del código de caracteres (o el tamaño de palabra) que esté definido en nuestro sistema.

Por lo general, se considera que un byte está formado por 8 bits, aunque en algunos sistemas la longitud de palabra (y por tanto el byte) tiene otro tamaño. Debido a esta asignación de un byte como 8 bits, en castellano se suele utilizar la palabra octeto para sustituir a la palabra anglosajona byte.

Siguiendo con esa asignación, un byte (de 8 bits) puede contener $2^8 = 256$ valores, y por tanto puede representar 256 informaciones diferentes, que serían las siguientes:

0000000	0000001	0000010	0000011	0000100	0000101	0000110	0000111
0001000	0001001	0001010	0001011	0001100	0001101	0001110	0001111
0010000	0010001	0010010	0010011	0010100	0010101	0010110	0010111
0011000	0011001	0011010	0011011	0011100	0011101	0011110	0011111
0010000	0010001	0010010	0010011	00100100	00100101	00100110	00100111

00101000	00101001	00101010	00101011	00101100	00101101	00101110	00101111
00110000	00110001	00110010	00110011	00110100	00110101	00110110	00110111
00111000	00111001	00101010	00111011	00111100	00111101	00111110	00111111
01000000	01000001	01000010	01000011	01000100	01000101	01000110	01000111
01001000	01001001	01001010	01001011	01001100	01001101	01001110	01001111
01010000	01010001	01010010	01010011	01010100	01010101	01100110	01010111
01011000	01011001	01001010	01011011	01011100	01011101	01011110	01011111
01100000	01100001	01100010	01100011	01100100	01100101	01100110	01100111
01101000	01101001	01101010	01101011	01101100	01101101	01101110	01101111
01110000	01110001	01110010	01110011	01110100	01110101	01100110	01110111
01111000	01111001	01101010	01111011	01111100	01111101	01111110	01111111
10000000	10000001	10000010	10000011	10000100	10000101	10000110	10000111
10001000	10001001	10001010	10001011	10001100	10001101	10001110	10001111
10010000	10010001	10010010	10010011	10010100	10010101	10100110	10010111
10011000	10011001	10001010	10011011	10011100	10011101	10011110	10011111
10100000	10100001	10100010	10100011	10100100	10100101	10100110	10100111
10101000	10101001	10101010	10101011	10101100	10101101	10101110	10101111
10110000	10110001	10110010	10110011	10110100	10110101	10100110	10110111
10111000	10111001	10101010	10111011	10111100	10111101	10111110	10111111
11000000	11000001	11000010	11000011	11000100	11000101	11000110	11000111
11001000	11001001	11001010	11001011	11001100	11001101	11001110	11001111
11010000	11010001	11010010	11010011	11010100	11010101	11100110	11010111
11011000	11011001	11001010	11011011	11011100	11011101	11011110	11011111
11100000	11100001	11100010	11100011	11100100	11100101	11100110	11100111
11101000	11101001	11101010	11101011	11101100	11101101	11101110	11101111
11110000	11110001	11110010	11110011	11110100	11110101	11100110	11110111
11111000	11111001	11101010	11111011	11111100	11111101	11111110	11111111

Lo que un conjunto de ceros y unos signifique depende de la manera en que nosotros hayamos decidido codificar nuestra información (por ejemplo, de la codificación de caracteres que utilice nuestro sistema, o de la tabla de caracteres vigente), pero es importante tener presente que la información guardada en un sistema digital (en una memoria) siempre lo estará, en última instancia, en el formato anterior.

1.2.2. Codificación de caracteres.

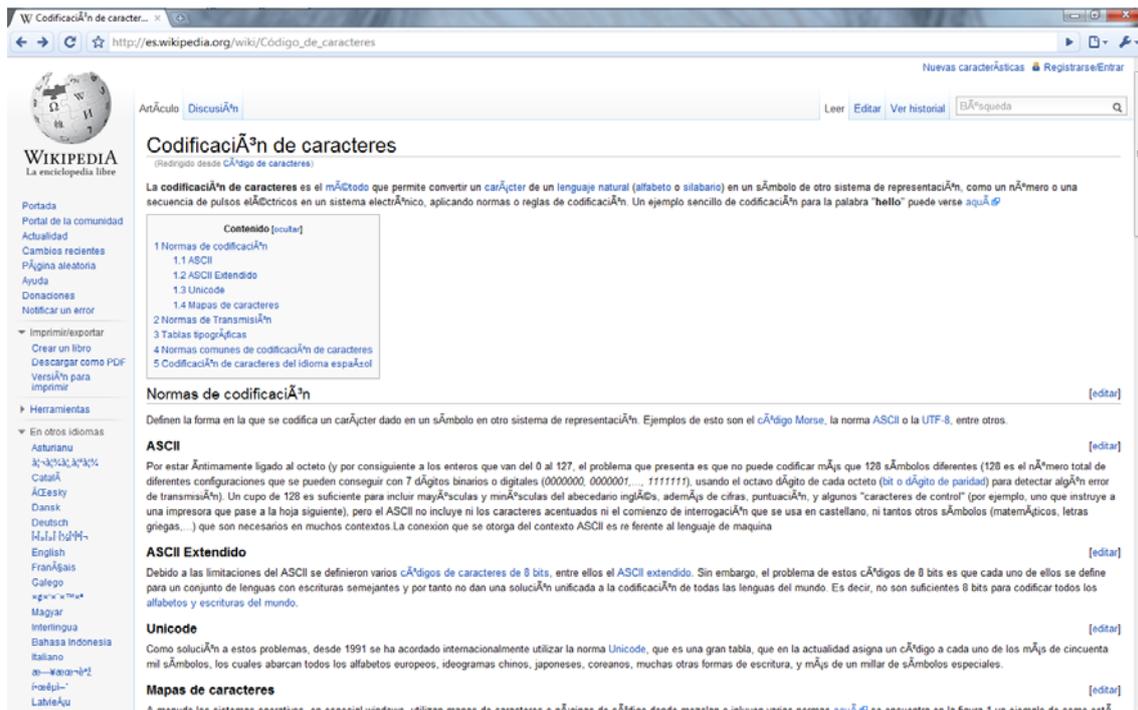
Generalmente, en el disco duro de un ordenador, la información se almacena en archivos. Un archivo es un conjunto de bits (y también de bytes) que tiene un nombre único, y que para el sistema operativo de nuestro ordenador se entiende como una unidad lógica. Es importante resaltar aquí la diferencia entre la representación del archivo en el disco duro, que ocupará un número determinado de bytes (o de unidades de memoria física) y la representación del mismo para el sistema operativo, para el cual el archivo es una unidad.

Estos archivos pueden ser de muy diversos tipos (imágenes, sonidos, vídeos, ejecutables, archivos de configuración, archivos propios del sistema operativo...). Lo que pueda significar un bit o un byte en un disco duro depende directamente del tipo de archivo en que nos encontremos. De este modo, un byte en un fichero de imagen puede hacer referencia al color de un píxel, mientras que ese mismo byte en una base de datos puede representar un número cualquiera, o en un fichero de un procesador de textos puede contener información sobre la tipografía utilizada (Arial, Times, Verdana...).

Existe un tipo de archivos particulares que se conocen como "ficheros de texto" que se caracterizan porque contienen exclusivamente información textual (caracteres) y no llevan información sobre el formato del mismo (tamaño de la fuente, tipo, color, márgenes...). Estos archivos son de amplio uso, ya que permiten guardar códigos fuente de programas, archivos de configuración de equipos, ficheros con grandes cantidades de datos...

Más adelante profundizaremos en el tratamiento de archivos de cualquier tipo en el disco duro. Por el momento, sólo nos vamos a interesar por cómo la información (en este caso texto sin formato, o texto plano) es almacenada en los citados archivos de texto.

Cuando escribimos el carácter "A" y lo guardamos en nuestro ordenador en un fichero de texto plano (por ejemplo, en el bloc de notas, pero no en "Microsoft Word", que le aportaría formato al mismo), "alguien" (nosotros, el programa en uso o el sistema operativo) habrá determinado un sistema de codificación para guardar "A" como un conjunto de ceros y unos, que representan la codificación del carácter en dicho sistema de codificación. Por tanto, es imprescindible que cuando leamos lo que hemos guardado lo hagamos con un programa que utilice el mismo sistema de codificación de caracteres que se usó al guardarlo. En caso contrario, podría suceder lo que observamos en la siguiente captura de pantalla de un navegador. ¿Qué ha sucedido con todos los caracteres acentuados o no estándar?



Antes de que entremos en más detalle sobre las codificaciones de caracteres es importante fijar una serie de conceptos que definiremos a continuación. La diferencia entre ellos puede parecer sutil, pero es necesario comprenderla para poder entender los procesos de codificación y decodificación de caracteres que tienen lugar sobre ficheros de texto plano o sin formato (las siguientes definiciones no son estándar, y quizá en algunos textos las encuentres de forma distinta, pero es importante que distingas los tres conceptos que se presentan, ya que es bastante común confundirlos, haciendo la conversión de caracteres mucho más complicada de entender de lo que en realidad es).

- Repertorio de caracteres: es un conjunto de caracteres sin ninguna estructura. Lo que hacemos por medio de un repertorio de caracteres es definir los caracteres que van a ser importantes para nosotros en nuestra codificación. Sí que es relevante notar cómo un repertorio de caracteres puede depender de la región en que nos encontremos (no será lo mismo, posiblemente, un repertorio de caracteres para Asia que un repertorio de caracteres centroeuropeos o de Europa Occidental). En un sencillo ejemplo, podríamos considerar como un repertorio de caracteres el conjunto $\{a, e, i, o, u, A, E, I, O, U, á, é, í, ó, ú\}$.
- Tabla de caracteres: una tabla de caracteres puede entenderse como una aplicación que a cada carácter (de un repertorio de caracteres) le asigna un número entero (no negativo). El número entero suele ser conocido como "código de posición" (en la tabla de caracteres) o "número de código". El anterior repertorio de caracteres podría ser representado ahora por la siguiente tabla de caracteres: $\{a=1, e=2, i=3, o=4, u=5, A=6, E=7, I=8, O=9, U=10, á=11, é=12, í=13, ó=14, ú=15\}$, pero también por otras tablas de caracteres distintas (por ejemplo, en orden inverso, o sólo por números pares).

- Codificación de caracteres: una codificación de caracteres es un algoritmo o regla que permite convertir los códigos de posición de una tabla de caracteres en octetos (o secuencias de octetos). La codificación más sencilla es convertir cada código a su representación binaria {a = 00000001, b = 00000010, c = 00000011...}. Esto funciona bien para tablas de menos de 256 caracteres. Sin embargo, es muy fácil encontrar otras codificaciones diferentes; por ejemplo, invirtiendo el orden de los bits: {a = 10000000, b = 01000000, c = 11000000...}. Para tablas con más de 256 elementos los algoritmos son más complejos, ya que se hace necesario más de un octeto.

Los computadores de los años 80 utilizaban procesadores de 8 bits (un octeto), y eso hizo que la mayor parte de las codificaciones de caracteres se ajustaran a esas dimensiones (es decir, que cada palabra ocupe un octeto). Es importante notar que toda codificación que ocupe un octeto va a permitir representar un máximo de 256 caracteres. En este contexto, una de las tablas o normas de codificación que se hicieron más populares fue el ASCII (American Standard Code for Information Interexchange). Siendo un estándar de origen occidental (más concretamente norteamericano) sirve para codificar caracteres latinos (por supuesto, no contiene caracteres cirílicos, hebreos, árabes...).

Sus principales características serían las siguientes. Su repertorio de caracteres está compuesto sólo por 128 caracteres y símbolos de control. De éstos, 32 corresponden a caracteres de control, algunos de los cuales se usaban en la época para enviar señales a dispositivos periféricos tales como impresoras, y el resto se corresponden con caracteres estándar (mayúsculas, minúsculas, signos de puntuación...). Se puede encontrar la definición concreta de los operadores de control en <http://es.wikipedia.org/wiki/ASCII>.

La tabla de caracteres ASCII con sus códigos de posición sería la siguiente:

					0	0	0	0	1	1	1	1	1
					0	0	1	0	1	0	1	0	1
Bits	b ₄	b ₃	b ₂	b ₁	Column	0	1	2	3	4	5	6	7
	↓	↓	↓	↓	Row	↓	↓	↓	↓	↓	↓	↓	↓
0	0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	4	EOT	DC4	\$	4	D	T	d	t
0	1	0	1	5	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	8	BS	CAN	(8	H	X	h	x
1	0	0	1	9	9	HT	EM)	9	I	Y	i	y
1	0	1	0	10	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	11	VT	ESC	+	;	K	[k	{
1	1	0	0	12	12	FF	FC	,	<	L	\	l	/
1	1	0	1	13	13	CR	GS	-	=	M]	m	}
1	1	1	0	14	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	15	SI	US	/	?	O	_	o	DEL

El repertorio de caracteres ASCII consta sólo de 128 caracteres. Si observamos que $2^7 = 128$, con 7 bits sería suficiente para poder codificar la tabla de caracteres ASCII. Sin embargo, internamente, y ya que la longitud de palabra del ordenador es el octeto, cada carácter se codifica con 8 bits. Para ello el proceso (o algoritmo) de codificación sería el siguiente. A cada carácter se le asignan los 7 bits correspondientes a la codificación en binario de su posición en la tabla de caracteres. Por ejemplo, al carácter "O", cuya posición en la tabla es 79 (puedes contar las columnas, o convertir 4 - 15 de hexadecimal a digital para conocerlo), le asignamos los 7 bits de su representación binaria ($79 = 1 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 + 1 \cdot 2^3 + 0 \cdot 2^4 + 0 \cdot 2^5 + 1 \cdot 2^6$), es decir, 1001111. Para determinar el bit de paridad, que colocaremos en primera posición, contamos el número de "1"s en la representación binaria de "O" (en este caso 5), y la convertimos en par (de ahí el nombre de bit de paridad), añadiendo un nuevo 1. Finalmente, la codificación del carácter "O" queda como "11001111". Si el carácter a representar fuese "P", cuya posición en la tabla es $80 = 0 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 0 \cdot 2^3 + 1 \cdot 2^4 + 0 \cdot 2^5 + 1 \cdot 2^6$ y cuyo bit de paridad será 0 (ya que la codificación anterior contiene dos "1" y ya es par), sería 01010000.

El repertorio (y la tabla) de caracteres ASCII se convirtió en un estándar de facto, pero pronto se vieron sus limitaciones, sobre todo por el corto número de caracteres que se pueden codificar en la misma (se puede observar, por ejemplo, que no se pueden encontrar ni vocales acentuadas "á, é, í...", ni la letra "ñ", ni la "u" con diéresis "ü", propios del castellano, ni muchos otros caracteres de otros idiomas).

Se hizo necesaria la definición de nuevos repertorios de caracteres, que utilizaran al menos toda la capacidad del octeto (256 valores) para codificar caracteres. Debido al papel preponderante que había adquirido ASCII en el mercado de los ordenadores, las tablas de caracteres que surgieron después coincidieron en incluir los caracteres "representables" ASCII (es decir, desde

el 33 hasta el 128) codificados de la misma forma que en la norma ASCII (así nos aseguramos que cualquier texto que sólo contenga caracteres de la norma ASCII se podrá guardar y leer en cualquier norma de codificación de la misma forma).

Uno de los intentos de aumentar la norma ASCII fue la conocida como ASCII extendida. Existen múltiples códigos ASCII extendidos. Su principal característica es que contienen los caracteres del 33 al 128 idénticos a los de ASCII (los caracteres de control en las posiciones 1 a 32 pueden variar). Los caracteres 129 a 256 se utilizan para codificar caracteres propios de distintas regiones o lenguas. Algunas de estas codificaciones son la ISO-8859-1 (también conocida como ISOLatin1), cuyos caracteres imprimibles puedes encontrar en <http://es.wikipedia.org/wiki/ISO-8859-1> (que coincide en todo el rango 1 – 128 con la tabla ASCII), la 850 (DOSLatin1), página de códigos que se utiliza en la consola de MSDOS para Europa Occidental, cuya codificación se puede encontrar en <http://aspell.net/charsets/codepages.html>, o Windows 1252 (WinLatin1), que también se puede encontrar en <http://aspell.net/charsets/codepages.html>. Es interesante comprobar algunos caracteres especiales (entre las posiciones 129 y 256) en cada una de ellas y ver si coinciden o no, eso podría explicar algunos fallos de visualización que podemos encontrar en algunos ficheros de texto o páginas web.

La falta de un estándar globalmente aceptado y la diversificación de normas de codificación llevaron a la creación del repertorio de caracteres Unicode. Unicode es un estándar de representación de caracteres que pretende facilitar el trabajo con caracteres de múltiples idiomas (incluyendo lenguas muertas o lenguas basadas en ideogramas como chino, coreano, japonés) o disciplinas técnicas (como matemáticas, física, química, ingeniería).

Veamos ahora con un poco más de detalle la tabla de caracteres para una parte de Unicode. En primer lugar vamos a presentar el repertorio de caracteres y su posición (es decir, la tabla de caracteres) y más adelante presentaremos los algoritmos y reglas de codificación de dichos caracteres. No incluimos los repertorios y tablas en este texto por ser demasiado extensa. Se pueden encontrar en http://en.wikibooks.org/wiki/Unicode/Character_reference/0000-0FFF. Si navegas por esta página podrás observar que hay al menos otras 15 tablas como la anterior (cada una con 4096 caracteres), dando lugar a lo que se conoce como BMP (Basic Multilingual Plane), y algunas otras más. La tabla anterior sólo contiene $16^3 = 4096$ caracteres (a día de hoy se calcula que Unicode cuenta con más de 100.000 símbolos ya definidos, y dispone de espacio para al menos 2^{20} símbolos).

Unicode, tal y como la hemos introducido, es una tabla de caracteres (o, en cierto modo, un conjunto de tablas de caracteres). Pero todavía no hemos dicho nada de la codificación de estos caracteres en sistema binario.

El número de caracteres que permite Unicode es tan elevado que el mismo ha dado lugar a distintas formas de codificación. Las tres más extendidas son:

1. UTF - 8. Codificación orientada a byte (de 8 bits u octeto) con símbolos de longitud variable (cada símbolo ocupará en memoria de 1 a 4 octetos). La codificación UTF - 8 permite representar no sólo 256 caracteres (un octeto) sino cualquier carácter que esté en Unicode. Esto es posible debido a la idea de codificación de longitud variable. Si queremos representar un carácter que esté entre los símbolos 0 y 127 de Unicode (que coinciden con los ASCII), la codificación utilizará en memoria un único octeto. Si queremos representar un carácter que esté fuera de dicho rango, su codificación utilizará de 2 a 4 octetos, 16 a 32 bits.
2. UTF - 16. Codificación de 16 bits de longitud variable (cada símbolo ocupará en memoria 1 ó 2 segmentos de 16 bits). Por tanto, desde el punto de vista de memoria, es bastante peor que UTF - 8 (por lo general, un texto que no contenga caracteres especiales, ocupará el doble que en UTF - 8). Sin embargo, permite codificar con dicha longitud de palabra $2^{16} = 65536$ caracteres, lo cual nos asegura que prácticamente cualquier carácter que queramos almacenar va a estar disponible. Gracias a que es de longitud variable, si pretendemos codificar algún carácter que no esté entre los 65536 disponibles pero sí en las tablas Unicode (hasta la posición 10FFFFFF), aumentará la longitud de palabra a 2 bytes.
3. UTF - 32. Codificación de longitud fija. Dispone de 2^{32} (más de $4 * 10^9$) símbolos. Cada carácter ocupa en memoria 32 bits (4 octetos o bytes).

Abre el enlace correspondiente a la primera tabla del BMP (http://en.wikibooks.org/wiki/Unicode/Character_reference/0000-0FFF) y observemos algunos detalles que nos permitirán repasar algunos de los conceptos vistos hasta ahora.

- En primer lugar, si compruebas la filas que van desde la "0000" hasta la "0070" (¿qué número representa 0070 en formato hexadecimal?) y las columnas de la "0" hasta la "F", podrás ver que los 128 primeros caracteres de Unicode, tal y como ya habíamos anunciado, son idénticos a los de ASCII (y a los de todas las codificaciones que hemos visto hasta ahora). Por ejemplo, el símbolo "e", que en la tabla de caracteres ocupa la posición "0065" en hexadecimal, diríamos que ocupa la posición 101 (en sistema decimal) de la tabla de codificación Unicode o la posición "1100101" en binario. Como ya anunciamos anteriormente, la posición en la tabla no tiene por qué coincidir con la codificación en binario. Su codificación binaria en UTF - 16 sería "00000000 01100101", y en UTF - 8 sería "01100101" (el último octeto de la representación UTF - 16).

- En segundo lugar, puedes comprobar los caracteres que van desde la fila "0800" hasta la "0F00". Son los caracteres que en la tabla Unicode ocupan la posición desde la 129 hasta la 256. A priori, son aquellos cuya codificación debería ocupar sólo un octeto (un byte) en codificación UTF-8. En realidad la codificación UTF-8, al ser de longitud variable, debe incluir bits de paridad en cada octeto, y ello hace que, por ejemplo, en la práctica, el carácter "ñ" no se codifique como "F1" en hexadecimal, o "11110001", sino como "C3B1", o "11000011 10110001" (lo puedes comprobar, por

ejemplo, con el "Pspad"). La razón por la que "11110001" pasa a convertirse en "11000011 10110001" se puede encontrar en <http://es.wikipedia.org/wiki/UTF-8> (en realidad no nos preocupa tanto el cómo se codifican, sino que el motivo para hacerlo así es para poder definir un código de longitud variable). Sería interesante comparar los símbolos que ocupan en la tabla Unicode las posiciones del 129 al 256 con los equivalentes de ISO-8859-1 o Windows-1252.

- Las filas siguientes, de la "0100" en adelante, contienen los símbolos Unicode a partir del 256. Si los observas verás que algunos de ellos son de uso muy extendido. Su codificación requiere más de un octeto. Sin embargo UTF-8 permite codificar los mismos gracias a su longitud variable. Por ejemplo, si queremos codificar la cadena "dě" en UTF-8 (en UTF-16 no supondría ningún problema, al encontrarnos dentro de su rango de símbolos) nos encontraremos con la siguiente secuencia de bits (d = 100, ě = 283): 01100100 11000100 10010101.

Como en todas las codificaciones de longitud variable, el problema ahora es intentar decodificar la anterior cadena "dě". ¿Cómo sabe el decodificador de la misma que debe considerar "01100100" como un primer carácter y "11000100 10010101" como un segundo carácter? ¿Por qué no decodificarlo como una cadena de tres caracteres? ¿O considerar los dos primeros octetos como un símbolo y el tercero como otro? Éste es el gran problema de las codificaciones de longitud variable. Sin información adicional, es imposible saber cuáles son los símbolos que debemos producir al decodificar.

Para eso hemos hecho antes el "truco" de convertir la "ñ", "F1" en hexadecimal, o "11110001", en "C3B1" o "11000011 10110001". En codificación UTF-8 sabemos que siempre que un octeto empieza por "110xxxxx" (como la "ñ" o la "ě" anterior), estamos ante un símbolo de 2 octetos (y por lo tanto debemos leerlo y decodificarlo junto al siguiente símbolo). Si aún necesitáramos usar tres octetos para algún carácter, el primero de ellos comenzaría por "1110xxxx". El segundo (y en su caso el tercer) octeto siempre comenzarían por "10xxxxxx" (de nuevo, como es el caso en los segundos octetos de la "ñ" o la "ě"). Así se explica que los caracteres del 129 al 256 (cuya representación binaria en UTF-8 sería "1xxxxxxx") no puedan ser representados con un único octeto, sino con dos octetos.

Si bien la forma de codificación en UTF-8 puede resultar un poco complicada de entender, es importante retener las ideas de que existen codificaciones de longitud variable, de que Unicode en sus distintas codificaciones se está convirtiendo en un estándar ampliamente aceptado en la informática, y por supuesto de la importancia de codificar y decodificar con respecto a la misma norma de codificación para recuperar los textos originales.

Sólo como curiosidad, en la página web <http://www.sslug.dk/~chlor/utf-8/> se pueden encontrar los símbolos Unicode entre las posiciones 129 y 2047, con su codificación correspondiente en hexadecimal y en binario (por ejemplo, puedes buscar la "ñ" o la "ě" y compararlas con las codificaciones que hemos propuesto antes).

El hecho de que los ordenadores se basen en una "tecnología binaria" ha motivado que, al hablar de unidades de información (por analogía de memoria), trabajemos siempre en potencias de 2 y, por ello, los típicos prefijos del Sistema Métrico Decimal, Kilo, Mega, Giga... tienen una pequeña diferencia cuando hablamos de memoria:

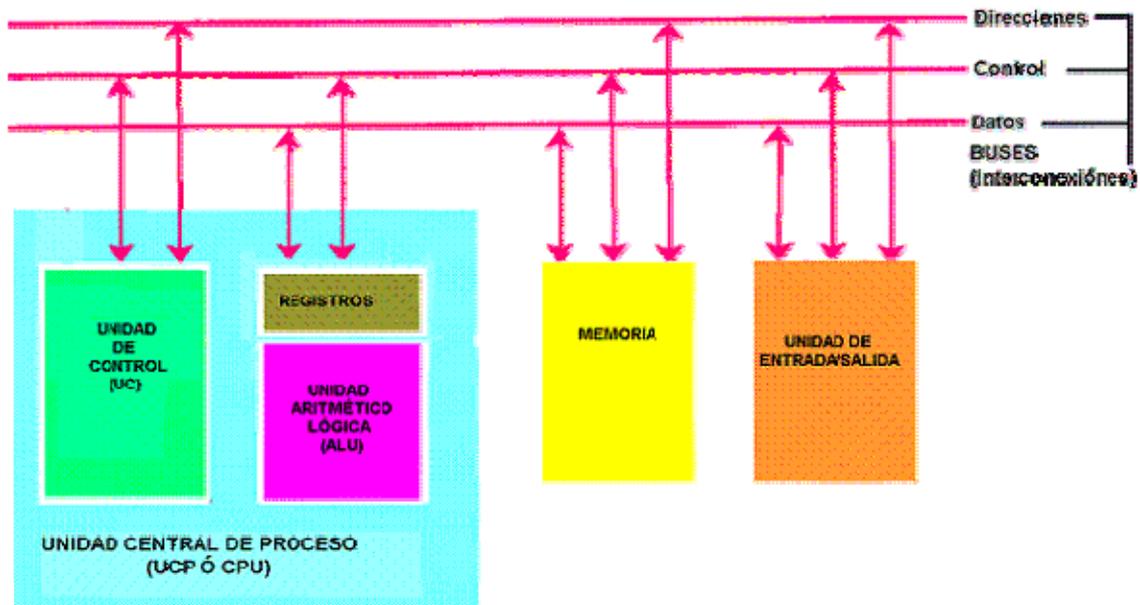
1 Kb (Kilobyte)	2^{10} bytes = 1024 bytes
1 Mb (Megabyte)	2^{20} bytes = 1024 Kb
1 Gb (Gigabyte)	2^{30} bytes = 1024 Mb
1 Tb (Terabyte)	2^{40} bytes = 1024 Gb

¿Serías capaz de calcular cuantos caracteres en codificación ISO-8859-1 podemos alojar en un Mb, o en un Gb?

¡ATENCIÓN!: La diferencia (de considerar 1024 en lugar de 1000) es sólo al hablar de unidades de información o memoria. Kilo, Mega, Giga... son prefijos usados en el Sistema Métrico Decimal y alrededor de un sistema informático hay muchas cosas que se miden que no son cantidades de memoria. Sirva como ejemplo la velocidad de un procesador (2'4 Ghz, por ejemplo).

1.3 Componentes básicos de un ordenador

La estructura funcional básica en la que se fundamentan los ordenadores actuales tiene su origen en la década de los años 40 del siglo XX. A esta estructura se le llama en ocasiones la estructura de "programa almacenado" o de Von Neumann. El matemático alemán John Von Neumann propuso una estructura de ordenador con 4 elementos funcionales en la que destacaba notablemente el hecho de que la memoria almacenaba tanto datos (información) como instrucciones (programas):



El aspecto novedoso de la propuesta de Von Neumann reside en que las propuestas previas siempre habían considerado el uso de una memoria para los datos y otra para los programas.

En la estructura de Von Neumann destacan los siguientes elementos:

- UCP (unidad central de proceso o procesador), que, a su vez, está constituida por:
 - a. Registros: son un número reducido de áreas de almacenamiento en las que se tiene la información con la que en cada momento trabaja la UCP.
 - b. UAL (Unidad aritmético-lógica): se encarga de realizar las operaciones aritméticas o lógicas con los datos que se encuentren en los registros correspondientes.
 - c. UC (Unidad de control): dirige todo el proceso, leyendo de memoria la instrucción a ejecutar y generando las señales de control para que se realicen las operaciones pertinentes.
- Memoria: en ella se pueden almacenar datos y programas (se corresponde con lo que solemos denominar la memoria RAM, aunque como veremos más adelante la memoria RAM puede tener más utilidades).
- Unidades de Entrada/Salida: para conectar el ordenador con el mundo exterior (por ejemplo, con monitores, impresoras, discos duros o cualquier otro tipo de periféricos).
- Elementos de interconexión (o buses): son los caminos a través de los que fluye la información entre las demás unidades funcionales dentro del ordenador.

Pasemos a ver de forma un poco más detallada algunos de estos componentes.

UCP. Funcionamiento básico.

Una UCP (o CPU en sus siglas en inglés) se construye con un "juego de instrucciones" que la misma es capaz de entender y ejecutar y que se conoce como lenguaje máquina del computador. Comentamos antes que el gran avance de la arquitectura de Von Neumann fue la idea de que en la memoria se podían guardar no sólo datos sino también programas. Un programa será una secuencia de "instrucciones" (en lenguaje máquina) y, teniendo en cuenta que la memoria sólo se compone de bits, cada instrucción ha de tener su codificación en código binario.

El lenguaje máquina depende de cada arquitectura o UCP sobre la que lo estemos ejecutando. El ejemplo más típico de lenguaje máquina es el lenguaje ensamblador. El lenguaje ensamblador es un lenguaje de bajo nivel dependiente de la plataforma en la que lo estemos utilizando y en cuyo lenguaje las órdenes simbolizan las instrucciones de la UCP, los registros del procesador, las posiciones de memoria en las que se almacenan datos o programas, etc. El lenguaje ensamblador fue de uso muy extendido en los primeros tiempos de la informática, cuando los lenguajes de programación de alto nivel (FORTRAN, COBOL, C...) no estaban muy desarrollados y el hardware era mucho más limitado. El desarrollo de lenguajes de programación de alto nivel (como los antes citados) que permiten en una sola orden llevar a cabo varias instrucciones propias de la máquina, las mejoras en el hardware y la poca portabilidad de los programas en ensamblador lo han hecho caer en desuso, aunque siguen existiendo usuarios y situaciones que lo hacen ser útil y conveniente. Puedes encontrar más información sobre lenguaje ensamblador y algunos ejemplos del mismo para diversas arquitecturas en http://es.wikipedia.org/wiki/Lenguaje_ensamblador.

Dentro de la UCP hemos nombrado antes la existencia de diversos tipos de registros. Los más relevantes serían el registro MAR ("memory address register" o registro de direcciones de memoria), donde se puede encontrar la dirección de memoria de las siguientes lectura o escritura. El MDR ("memory data register", o registro de datos de memoria) que contiene los datos que se van a escribir en la memoria o que recibe los datos leídos de la misma. También podemos encontrar el registro de instrucción (IR), que contiene la última instrucción leída, así como el contador de programa (PC), que contiene la dirección de la próxima instrucción que se leerá de memoria.

Como se puede observar, la información en algunos de estos registros es redundante (datos o direcciones de memoria...). A priori podría parecer que los mismos no son de gran utilidad. Su uso consigue aumentar el rendimiento del sistema, ya que permite acceder a informaciones (datos, memoria) con mucha mayor velocidad (al estar integrados en la propia UCP) que si los mismos debieran ser sacados de la memoria del ordenador. Su utilidad se puede identificar con la de una memoria caché (dentro de la propia UCP), que dispone de poca capacidad pero de gran velocidad de acceso.

Sin embargo, es importante recordar que la principal misión de la UCP es la ejecución de las instrucciones propias de la máquina. La forma de ejecutar una instrucción cualquiera será la siguiente, siempre bajo la supervisión de la unidad de control (UCP). El procesador comprueba el valor que contiene el contador del programa (el registro PC antes citado). Se dirige a la instrucción que encuentra en dicha dirección y la almacena en el registro de instrucción (IR). Ahora el procesador lee los bits correspondientes a dicha instrucción, y lleva a cabo la acción requerida. Los tipos de instrucciones que podemos encontrar se pueden dividir en cuatro categorías:

- Comunicación entre el procesador y la memoria: se transfieren datos entre ambas partes del sistema. Para ello entran en funcionamiento los registros MAR y MDR. También se pueden transferir datos entre los propios registros.
- Comunicación entre el procesador y los dispositivos de E/S: operaciones de transferencia de datos entre el procesador y alguno de los módulos de E/S.
- Procesamiento de datos: el procesador (a través de la unidad aritmético-lógica) puede realizar algunas operaciones aritméticas o lógicas sobre los datos.
- Operaciones de control: saltos o bifurcaciones que rompen la secuencia de ejecución de instrucciones. Por ejemplo, una instrucción puede especificar que se va a alterar la secuencia de ejecución, saltando las siguientes "n" instrucciones.

En la página web <http://arity.ca/2010-11/CSE101/resources/RSCPU/> puedes encontrar un simulador de una UCP en el que puedes observar y modificar los distintos registros de que consta la misma.

Las órdenes que es capaz de ejecutar el procesador dependen del modelo o arquitectura del mismo. Por ejemplo, los procesadores de última generación de Intel para ordenadores de sobremesa (conocidos como Core i7) soportan un conjunto de instrucciones resultado de añadir las propias de x86, x86-64, MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1 y SSE4.2 (como puedes observar, x86 corresponde al nombre de una familia de procesadores ampliamente usada en los ordenadores de sobremesa <http://es.wikipedia.org/wiki/X86> y también nos sirve para identificar el lenguaje propio de esos procesadores http://es.wikipedia.org/wiki/Lenguaje_ensamblador_x86). Las últimas adendas a estas instrucciones, por lo general, tienen que ver con operaciones que permiten la generación de gráficos o la descompresión de formatos de vídeo más rápidamente. En la página http://en.wikipedia.org/wiki/X86_instruction_listings puedes encontrar el conjunto de instrucciones propio del conjunto x86. Como se puede ver, contiene las operaciones básicas que hemos ido nombrando antes, como escribir datos del registro a memoria y viceversa, realizar operaciones aritméticas (ADD,...), operadores lógicos (AND, OR, XOR...).

Para que el procesador sea capaz de "procesar" una serie de mandatos necesita que tanto las instrucciones como los operandos necesarios se encuentren en sus registros internos. Esta es otra de las razones por las cuales son necesarios los registros MDR y MAR.

Cada instrucción puede tener uno, dos o tres operandos:

- Si queremos sumar dos cantidades necesitamos los dos datos a sumar y el lugar donde vamos a colocar el resultado.
- Si queremos colocar un valor en un registro necesitamos el valor y el registro.
- Si queremos que se ejecute un salto en la ejecución de nuestro programa, necesitaremos la nueva dirección de memoria en la que se encuentra la instrucción a la que queremos saltar.

En función del tamaño de los operandos con los que puede trabajar la UCP (8, 16, 32... bits) diremos que nuestro procesador es de 8, 16, 32... bits. Como ya mencionamos en la parte referente a las codificaciones, en los años 80 muchos de los procesadores eran de 8 bits, de ahí que las codificaciones tuvieran también esa longitud de palabra. Hoy en día la mayor parte de las arquitecturas empiezan a ser ya de 64 bits, aunque muchos programas, controladores o utilidades siguen trabajando con longitudes de palabra de 32. Para que toda la arquitectura del ordenador sea coherente y efectiva, debe haber una relación coherente entre el tamaño de los registros de la UCP, el ancho de los distintos buses y las palabras de memoria. La mayor parte de los lenguajes de procesado (en particular x86 y sus sucesores) admiten distintos "modos" que se identifican con la longitud de las instrucciones que el procesador puede soportar (16 bits, 32, 64...).

Las instrucciones de la UCP se suelen realizar en una serie de pasos elementales que, en última instancia consisten en activaciones y desactivaciones de ciertas señales eléctricas. El conjunto de instrucciones depende de la plataforma, UCP o procesador de que dispongamos. En general, se puede afirmar que cualquier procesador dispone de los siguientes conjuntos de instrucciones:

Operaciones con enteros

Este es el conjunto de operaciones que generalmente lleva a cabo la UAL. Dependiendo de la arquitectura de nuestro procesador, los enteros ocuparán en memoria 8, 16, 32 ó 64 bits.

Las operaciones con enteros pueden ser:

- Operaciones aritméticas (sumas, restas, multiplicaciones...).
- Operaciones booleanas, operaciones lógicas bit a bit (AND, OR, NOT...).
- Operaciones con bits (desplazamientos, rotaciones).
- Comparaciones.

Operaciones con números reales

Están definidas en un estándar de la IEEE (siglas de Institute of Electrical and Electronical Engineers), en concreto en el IEEE 754. La mayor parte de los procesadores hoy en día ya incluyen un coprocesador numérico optimizado para este tipo de operaciones. Entre ellas se incluye:

- Operaciones aritméticas básicas (sumas, restas, divisiones...).
- Operaciones trigonométricas (senos, cosenos...).
- Operaciones con raíces, logaritmos y potencias.

Aparte del conjunto de operaciones lógicas y matemáticas, otra de las grandes funciones de los procesadores es la de "transportar" información entre la memoria de nuestro dispositivo y los registros propios de la UCP. Es importante recordar que la UCP trabaja siempre con información que se encuentra "dentro" de la misma (en sus registros) y por tanto las operaciones de lectura y escritura a dispositivos de memoria son muy habituales.

Operaciones de transferencia de datos

- Desde los registros a la memoria y viceversa: son operaciones de copia de datos (los datos permanecerán en el origen). Por ejemplo, podemos copiar datos de un registro a otro, de un registro a la memoria, de memoria a un registro...
- Operaciones de la pila: múltiples procesadores incluyen un registro especial conocido como la pila. En la misma se pueden alojar valores temporales de funciones o direcciones de retorno de estas funciones. Esto nos permite, por ejemplo, saber desde dónde ha sido llamada una función o subrutina y devolver el control del programa a ese punto, cuando dicha función ha completado su ejecución. Para manejar los datos y valores en la pila, se dispone de dos instrucciones "PUSH" y "POP". PUSH tiene un parámetro, "origen", de tal modo que la operación "PUSH origen" nos permite copiar a la dirección del primer elemento de la pila el dato disponible en "origen", y aumenta el tamaño de la pila por el orden del dato "empujado". Por el contrario, "POP", que también tiene un parámetro "destino", lo que hace es sacar un elemento de la pila y actualizar el valor de la pila para que apunte al siguiente elemento (el funcionamiento concreto de PUSH y POP es dependiente de la implementación de la pila que tengamos, lo importante es recordar que las dos instrucciones que permite la misma son PUSH y POP, y que permite albergar valores o direcciones de memoria en la pila).
- Operaciones de entrada/salida: operaciones que mueven datos de un registro o de la memoria desde o hacia los puertos. La operación INPUT permite hacer lecturas desde un puerto de entrada, la operación OUTPUT permite hacer operaciones de escritura hacia un puerto de salida.

Operaciones de control del flujo del programa

Estas operaciones incluyen todo lo relacionado con las llamadas y retornos de subrutinas o funciones, con la gestión de las interrupciones (que ahora presentaremos) y con saltos en la ejecución de la secuencia de las instrucciones de la UCP.

Cada familia de procesadores puede añadir instrucciones u operaciones adicionales que permitan optimizar ciertas tareas que son muy frecuentes

en el mismo. Por ejemplo, los procesadores de la familia x86 incluyen el siguiente conjunto de instrucciones específicas de su arquitectura:

Instrucciones SIMD

Conjunto de instrucciones para la realización de una misma operación sobre distintos valores en paralelo. Generalmente permite operaciones sobre enteros o valores en coma flotante.

Todas las anteriores instrucciones consumen ciclos o periodos propios del reloj del procesador. Este reloj marca el periodo mínimo que cualquiera de estas activaciones/desactivaciones (instrucciones) deben durar para que los componentes del ordenador sean capaces de procesarlas.

Para medir estos impulsos se utiliza una unidad de medida conocida como hercio (Hz). Los hercios miden la frecuencia de un suceso. Por ejemplo, si un suceso tiene una frecuencia de un hercio, quiere decir que se repite una vez por segundo. Si tiene una frecuencia de 1.000 Hz, este suceso se repetirá 1.000 veces en un segundo.

Por tanto, cuando decimos que un procesador trabaja a 200 MHz (MegaHertzios) estamos queriendo decir que el reloj genera 200 millones de pulsos por segundo ("Mega" es la abreviatura para 10^6). Si hablamos de 1 GHz (GigaHertzio), el reloj generará 1.000 millones de pulsos por segundo y, por tanto, trabaja 5 veces más rápido. Cada instrucción que es capaz de realizar una UCP tardará uno o varios ciclos de reloj.

Es importante entender que la velocidad de un ordenador no depende exclusivamente de la velocidad de su procesador (de la cantidad de Hercios que soporte su procesador), sino de la complejidad de las instrucciones que soporte su lenguaje máquina. En este sentido, las tarjetas gráficas, tienen un lenguaje máquina especialmente diseñado para soportar operaciones propias de cálculo y renderización de gráficos, que las pueden hacer más aptas para determinadas tareas que un procesador "genérico" muy potente (con un elevado número de hercios).

Casi todos los computadores proporcionan un mecanismo por el cual otros módulos (memoria y E/S) pueden interrumpir el secuenciamiento normal del procesador. Este mecanismo, conocido como "interrupciones", permite acompañar la velocidad del procesador con la velocidad de las operaciones de lectura o escritura de memoria o de dispositivos de E/S (por lo general, mucho más lentas que la velocidad que puede alcanzar el procesador).

No entra en los objetivos de este curso aprender lenguaje ensamblador o programar en el mismo, pero siempre puede ser de utilidad conocer algo de su sintaxis o su aspecto por si alguna vez tienes que enfrentarte al mismo. En la página http://es.wikipedia.org/wiki/Lenguaje_ensamblador#Ejemplos_de_lenguaje_ensamblador se pueden encontrar varios ejemplos del lenguaje para diversas arquitecturas de procesadores.

Memoria

Dentro de la arquitectura clásica de Von Neumann hemos visto la utilidad y forma de trabajo de la UCP, así como de los registros y de la UAL. Existen otras dos componentes propias de la arquitectura que si bien han aparecido por su interrelación con la UCP, todavía no han sido descritas. Las mismas son la memoria y las unidades de entrada/salida. En realidad, la palabra "memoria" está sobrecargada en relación a la estructura de los ordenadores, ya que la misma puede hacer referencia a muy diversos tipos de memoria que vamos a presentar en las siguientes secciones.

El primer tipo de memoria al que accede la UCP son los registros propios del microprocesador, algunos de los cuales ya hemos presentado (como el MAR o el MDR). Existe un segundo tipo de memoria, que todavía no hemos presentado, que es la memoria caché, que más tarde introduciremos (que también forma parte del propio procesador). Después, nos encontramos con lo que se conoce (erróneamente, pero de forma muy extendida) como memoria RAM, o más correctamente memoria principal, que se encuentra fuera del procesador. Por último, podemos agrupar todos los dispositivos de almacenamiento externo propios del dispositivo, que comprenden discos duros, dispositivos USB, tarjetas de memoria, etc, a los cuales el procesador no accede directamente pero que sirven para almacenar datos de forma permanente.

Debemos ser capaces de diferenciar estos cuatro tipos de memoria y conocer las características básicas de cada uno de ellos. Los registros de la UCP ya han sido presentados en la anterior sección.

Tipos de Memoria

Desde un punto de vista amplio, memoria es cualquier dispositivo capaz de almacenar información (datos e instrucciones). En función del criterio que se siga para su clasificación, se puede hablar de muchos tipos diferentes de memoria. En la siguiente clasificación no vamos a prestar atención a su capacidad o a su velocidad, dos de los parámetros más relevantes a la hora de elegir o de seleccionar una memoria (que contemplaremos en la siguiente sección), sino que nos centraremos en sus propiedades de construcción o fabricación. Después veremos cómo se relacionan estos tipos de memoria con los que presentamos antes (registros, caché, memoria principal y dispositivos externos).

En función de la forma de acceso:

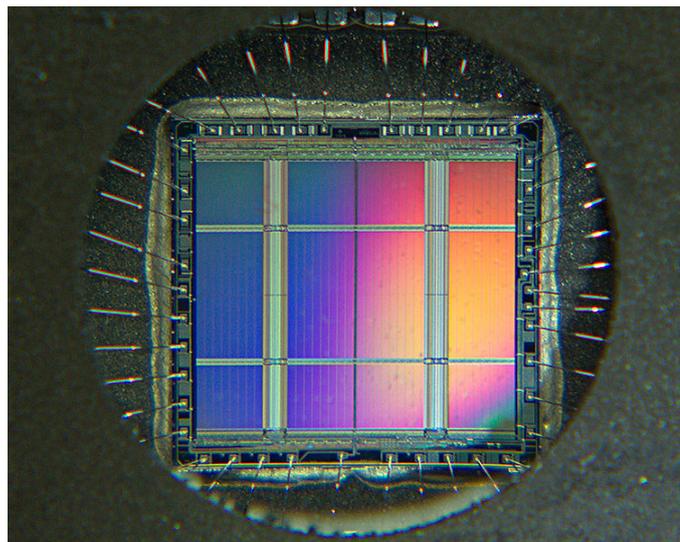
- RAM (Random Access Memory ó Memoria de acceso aleatorio). Aquella memoria a la que se puede acceder con igual costo de tiempo a cualquier dirección la misma (la memoria principal del ordenador responde a este modo de acceso, y por eso generalmente se la conoce como RAM; sin embargo, los registros del procesador también son un tipo de memoria RAM, y por tanto usar memoria RAM para

describir la memoria principal no es erróneo, tampoco es suficientemente preciso).

- Memoria de acceso secuencial. Aquella que no puede direccionarse y hay que seguir un orden secuencial para llegar hasta la información buscada. Un ejemplo puede ser una memoria en cinta, un disco o CD.

En función de las capacidades para leer y escribir en la misma:

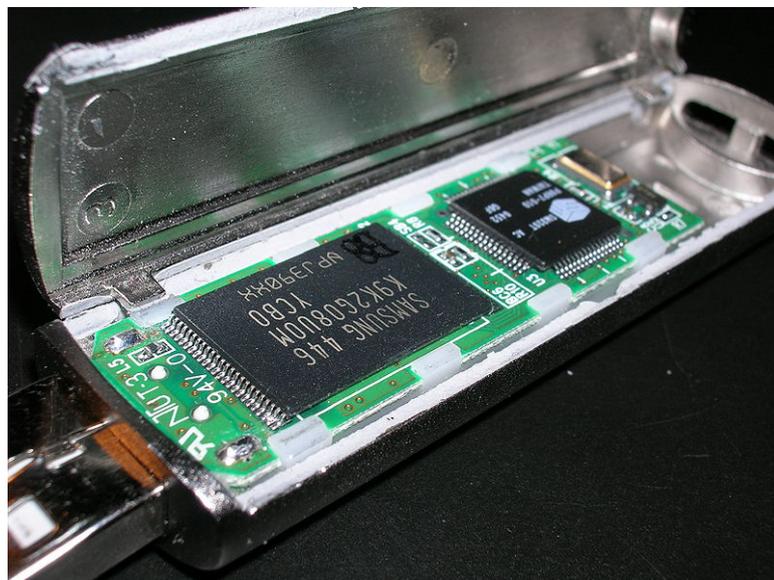
- De sólo lectura (ROM: Read Only Memory). Todas las memorias de este tipo son no volátiles, es decir, mantienen la información de manera indefinida. Se suele utilizar para almacenar los programas que ponen en marcha el ordenador o realizan los diagnósticos. En los primeros ordenadores, se usaban incluso para introducir en ellas el sistema operativo. Hoy también son de uso en algunas videoconsolas que las utilizan como almacén de juegos (también conocidos como cartuchos). Sólo permiten lectura de su información, pero no su destrucción.
 - ROM (Memoria de sólo lectura no programable): En el proceso de fabricación queda cargada con la información correspondiente.
 - PROM (Memoria de sólo lectura programable): Se adquiere el dispositivo de memoria con una arquitectura general y es programable (se carga la información) mediante un sistema no reversible. Se puede escribir una única vez.
- De lectura / escritura (RWM: Read Write Memory).
 - No Volátiles (mantienen la información de manera indefinida):
 - EPROM (Erasable PROM): Memorias que se pueden borrar usando luz ultravioleta y se reprograman eléctricamente.



- EEPROM (Electrically Erasable PROM): Memorias que se borran y graban eléctricamente. Tanto las memorias EPROM como las EEPROM fueron usadas y se siguen

usando en la actualidad con fines militares, pero también (en particular las EEPROM) sirvieron de inspiración para las hoy ampliamente extendidas memorias FLASH.

- FLASH: Se borran y graban eléctricamente pero, por la tecnología de fabricación, poseen un alto nivel de integración, lo que permite almacenar mucha memoria en dispositivos de reducido tamaño. Además, permiten la lectura y escritura de múltiples posiciones de memoria en la misma operación. Esto ha hecho que se hayan difundido de manera extensa y que, poco a poco, estén quitando el sitio a dispositivos magnéticos clásicos como los Discos Duros (HD), aunque todavía su precio es bastante más elevado.



- Volátiles (Necesitan una fuente de alimentación para mantener la información):
 - SRAM (RAM estáticas): Los procesos de lectura son rápidos pero admiten menos densidad de integración.
 - DRAM (RAM dinámicas): Los procesos de lectura son destructivos por lo que se necesita reescribir cada vez que se lee y esto las hace más lentas. Su capacidad de integración es mayor y es más barata de construir.
 - SDRAM (RAM dinámicas síncronas): generalmente las memorias DRAM son asíncronas, lo que quiere decir que los cambios en el estado de la memoria se producen tan rápido como se puede, pero sin sincronizarse con el resto de elementos de la placa (en particular, con el bus de transferencia de datos). Sin embargo, las memorias síncronas (SDRAM en particular), se sincronizan con el reloj del sistema, y por tanto espera a las señales del reloj para responder a las peticiones de lectura y escritura de datos. Su comportamiento es más complejo, pero a la vez permite mayor velocidad. Las memorias SDRAM se han extendido con celeridad en el

marcado. Disponen de tecnología DDR (double data rate) que permite dos lecturas o escrituras por ciclo. En la actualidad la tecnología DDR ha anunciado su cuarta generación (DDR4) y se prevén velocidades de la misma de hasta 2,133 GHz (es decir, completarán un ciclo cada $1 / (2,133 * 10^9)$ segundos, y en ese ciclo realizarán dos operaciones).

La extendida acepción de memoria RAM para la memoria interna del ordenador está motivada porque, en efecto, es un tipo de memoria de acceso aleatorio, pero ésta no es una identificación muy precisa. En realidad, la memoria caché del ordenador también es un dispositivo de acceso aleatorio. Sí es importante retener la idea de que la memoria interna del dispositivo será de tipo volátil, y por tanto será borrada cada vez que apaguemos el dispositivo.

Jerarquía de Memoria y memoria caché

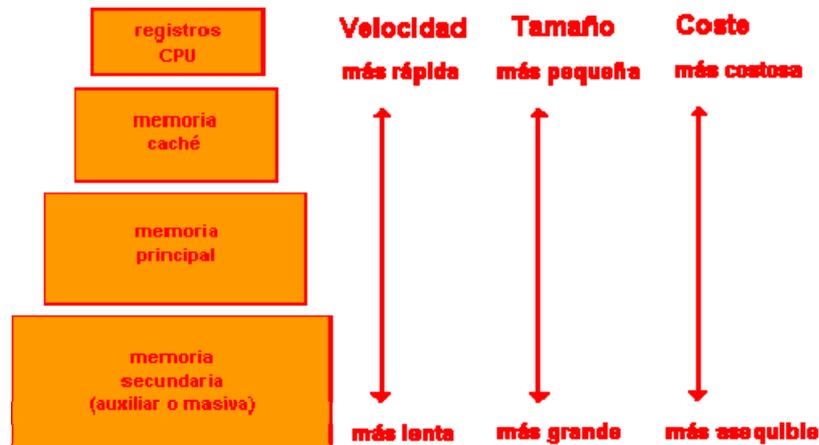
En la sección anterior hemos clasificado los tipos de memoria de dispositivos por su estructura o propiedades. Existe otra forma de clasificación de los tipos de memoria basada en su relación (o en su "distancia") con la UCP del dispositivo, y en el precio de la misma.

El problema fundamental de la memoria es la relación velocidad/precio. Para leer o escribir un dato en memoria se tarda un tiempo que, generalmente, es muy grande comparado con la velocidad a la que trabaja la UCP (para leer o escribir en la memoria, salvo en los registros de la UCP, se necesitan muchos ciclos de reloj, o muchos hercios, ya que debemos realizar las operaciones de transferencia de estos datos a través de los buses de la arquitectura del procesador). De este modo, la UCP debe estar mucho tiempo "esperando" a que lleguen los datos hasta sus registros desde la memoria principal o viceversa (como ya dijimos antes, de coordinar estas esperas se encarga el mecanismo conocido como "interrupciones"). Conseguir que la memoria principal vaya a velocidades cercanas en magnitud a las de la UCP es extremadamente caro.

Con respecto a este criterio de "distancia" o tiempo de lectura/escritura para la UCP, y precio, podemos distinguir al menos cuatro tipos de memoria. Así, podemos hablar de una jerarquía de memoria que comienza en los niveles más próximos a la UCP (en los que hay poca memoria pero muy rápida). Es importante notar que la jerarquía es válida, por lo general, tanto para la velocidad de acceso a la memoria, como para el tamaño de la misma:

- Registros de la UCP: ya los hemos mencionado (RDIM o RDAM). Son pocos y se puede trabajar con ellos a la velocidad de la UCP. De hecho, forman parte de ella.
- Memoria caché. La misma sirve de almacén intermedio entre la memoria interna del dispositivo y los registros propios del procesador.

- Memoria principal. Es externa al procesador y como hemos dicho, es volátil.
- Memoria secundaria o memoria externa (Dispositivos externos, E/S). Incluyen dispositivos de almacenamiento masivo como discos duros, memorias externas de tipo FLASH, o incluso repositorios en red (o en la nube).



Registros de la UCP

Los registros propios de la UCP (que también son un tipo de memoria interna del procesador) se pueden considerar como una memoria de muy poca capacidad (no podemos alojar gran cantidad de datos en los mismos) pero de muy rápido acceso, ya que se encuentran integrados en el propio microprocesador. Los registros se suelen medir por la capacidad en número de bits que almacenan (cada una de sus celdas) y por tanto se suele hablar de registros de 8 bits, de 32, 64... Los mismos pueden almacenar datos, direcciones de memoria, etc. Su tamaño es realmente pequeño, en http://en.wikipedia.org/wiki/Processor_register#Some_examples puedes observar el número de registros propios de algunos modelos clásicos de microprocesadores. Una forma de optimizar código para aplicaciones consiste en hacer el uso más conveniente de dicho número de registros, ya que el procesador accede a ellos a una muy alta velocidad. Esta memoria, en los procesadores actuales, suele ser de tipo SRAM o DRAM.

Memoria caché

Hay dos principios en informática que se han demostrado muy eficaces experimentalmente que son:

- Principio de localidad temporal: cuando un dato se acaba de usar, es muy probable que se vuelva a usar próximamente.
- Principio de localidad espacial: cuando un dato se acaba de usar, es muy probable que se usen datos que se encuentran en direcciones de memoria próximas.

En estos principios se basa la utilización de memoria caché. Uno de los mayores retos en ciencia de computadores consiste en decidir las estrategias que deciden qué datos van a ser introducidos en la memoria

caché. La memoria caché es una memoria más rápida y más pequeña que la memoria principal, que se coloca entre los registros de la UCP y ésta. Generalmente, los microprocesadores disponen de una memoria caché propia, integrada en la UCP, y suele existir también lo que se conoce como una caché de disco, que va en la memoria interna o principal (más lenta que la primera).

La memoria caché de la UCP es a su vez de dos tipos. Existe lo que se conoce como caché primario (L1), que va directamente integrada en el procesador. Su velocidad de acceso es similar a la de los registros propios de la UCP. El caché secundario (L2) suele ser de tipo SRAM (RAM estática), más rápida pero más cara que la memoria principal, generalmente de tipo DRAM (RAM dinámica). Es volátil, así que cada vez que apagamos un dispositivo la misma se vacía.

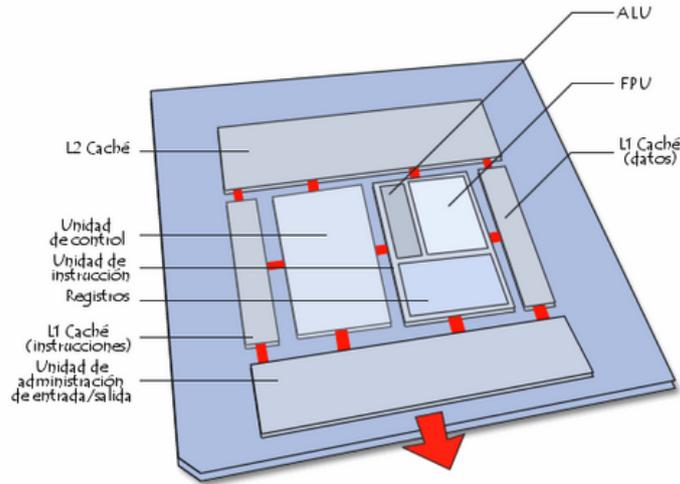
Cuando se necesita el contenido de una dirección de memoria, primero se busca en la memoria caché y si está allí se utiliza; si no está, se lee de memoria principal pero dejando el dato también en la memoria caché.

En el siguiente vídeo (<http://www.intel.com/technology/product/demos/cache/demo.htm>) puedes observar, entre otras cosas, cuál es el tamaño habitual de la memoria caché (L2) en algunos de los últimos procesadores que están saliendo al mercado (del orden de 4 a 8 MB), cómo la misma se sitúa junto a la UCP (dentro de la placa, pero fuera del procesador), en una posición intermedia entre la memoria RAM y la UCP, y cómo hay muy diversas estrategias de "poblarla" para optimizar su uso de cara a los datos solicitados por la UCP.

A la memoria caché de disco se la suele conocer también como L3.

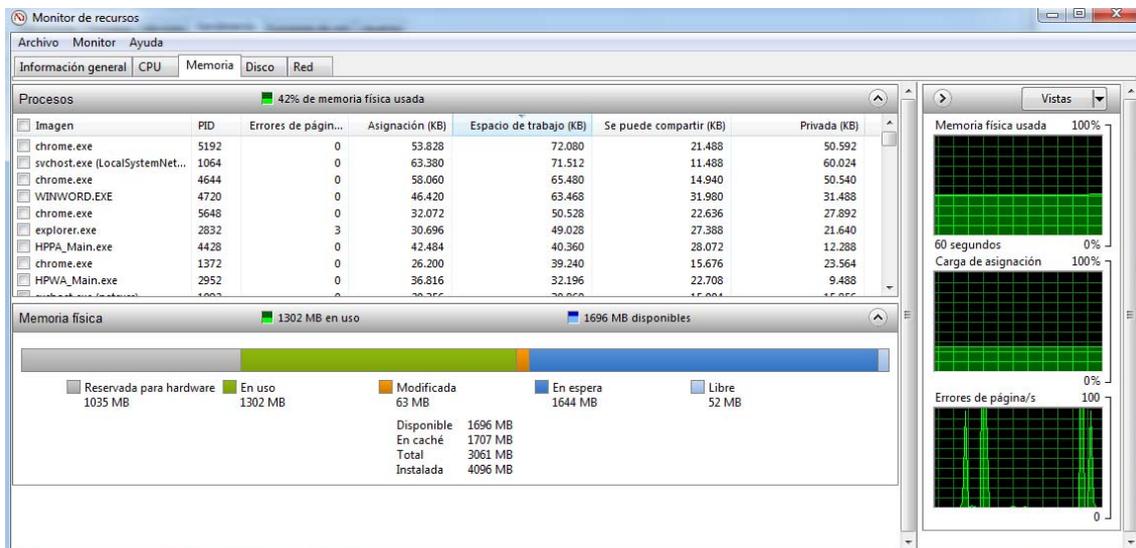
Los ordenadores actuales suelen trabajar con varios niveles de memoria caché (si tienes instalado en tu ordenador GRUB como gestor de arranque y utilizas el programa "memtest" del menú de inicio del mismo, puedes comprobar cuántos niveles de caché tiene tu ordenador y de qué tamaño): caché de nivel 1 que se encuentra dentro de la propia UCP y, caché de nivel 2 que se encuentra en la placa del ordenador.

En la siguiente imagen puedes observar la distribución de las distintas memorias caché en una placa:



Memoria principal, interna o RAM

La memoria principal de un dispositivo, también conocida como memoria RAM, y de tipo volátil, tiene una utilidad similar a la de la memoria caché. Se utiliza como memoria de trabajo para el sistema operativo, los programas y el software. En ella se cargan las instrucciones que se van a ejecutar en el procesador, en la UAL, etc. En realidad, por medio del software, es bastante fácil de comprobar la utilización y capacidad de la memoria RAM de un equipo.



Los principales parámetros que miden la eficacia de una memoria interna son: su tamaño y su velocidad de lectura y escritura. La velocidad de lectura y escritura, o velocidad de acceso, depende de la velocidad de transferencia de los datos y de la latencia (el tiempo que le lleva comunicarse con la UCP).

Dos de las principales características de la memoria principal es que la misma es de acceso aleatorio (de ahí el nombre de RAM), y que es volátiles. Que sea de acceso aleatorio quiere decir que la velocidad de lectura y escritura de datos en la memoria es la misma para cualquier posición (a

diferencia, por ejemplo, de un disco, en el que la lectura de ciertos datos implica el recorrido del mismo hasta la posición en la que dichos datos se encuentran). Como ya vimos en los tipos de memoria, hoy en día muchos tipos de memoria cumplen esta premisa, como por ejemplo las memorias FLASH. La segunda característica de las mismas es que son volátiles. Esto quiere decir que cada vez que un dispositivo es apagado, su memoria RAM es borrada y la información en la misma desaparece (a diferencia, por ejemplo, de las memorias externas). De ahí que su utilidad sea almacenar las instrucciones que la UCP ha de ejecutar, o los datos que la misma ha de usar, pero no incluir información que vaya a ser permanente.

La memoria principal, en los ordenadores actuales, está formada por módulos de memoria de tipo DRAM (RAM dinámica, que debe ser reescrita cuando un dato es leído de la misma, y por tanto es un poco más lenta que la SRAM, pero a la vez es mucho más económica).

La conexión de la memoria principal con el resto de componentes del dispositivo se realiza por una serie de pines en uno de los extremos del circuito. La necesidad de estandarización para los distintos fabricantes de módulos y procesadores llevó a la definición de una serie de modelos de conexión, de los cuales hoy en día el que se utiliza es el DIMM. Su principal característica es que su bus de datos es de 64 bits (8 bytes). También debemos tener en cuenta la frecuencia a la que es capaz de transferir los datos (tal y como sucede con los procesadores).

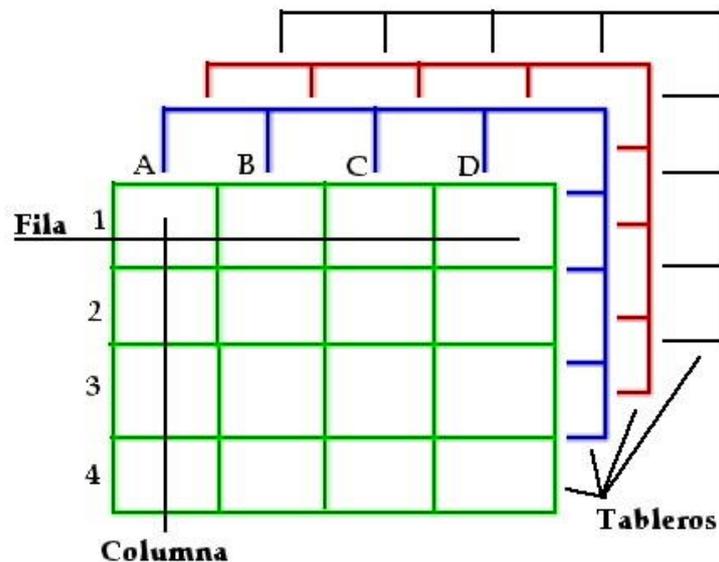
Siendo que la memoria principal sirve para almacenar las instrucciones y los datos que está manejando la UCP en un momento concreto, es claro que el hecho de tener una mayor memoria principal ("más Gigas de RAM", en jerga) permitirá tener más aplicaciones (o de mayor uso de memoria) ejecutándose en el mismo dispositivo. En general, a día de hoy, la capacidad más común de la memoria principal de un ordenador convencional de sobremesa (o un portátil) está entre los 2 y los 8 GB. Es bien conocida la limitación de algunos sistemas operativos (de 32 bits) todavía en uso para utilizar más de 4GB de memoria principal (puedes encontrar una explicación de esta limitación en <http://www.brianmadden.com/blogs/brianmadden/archive/2004/02/19/the-4gb-windows-memory-limit-what-does-it-really-mean.aspx>).

En general, la memoria principal es una pieza bastante modular dentro de las placas de los ordenadores, y reemplazarlas (o aumentarlas) es una operación sencilla. El aspecto de una memoria principal podría ser el siguiente:



Aparte de saber cómo es el aspecto externo de una memoria principal, es importante conocer cómo podemos medir su rendimiento. Si miras la especificación de una memoria principal, verás que se destacan de la misma su frecuencia y sus latencias. Las frecuencias actuales son del orden de entre 0,5 GHz y 2 GHz. Si tenemos en cuenta que el bus de comunicación es de 64 bits (8 bytes), en cada segundo la cantidad de información transferida podrá llegar a ser del orden de entre $8 * 0,5 \text{ GB}$ y $8 * 2 \text{ GB}$.

Antes de explicar en qué consisten los tiempos de latencia, es importante tener una idea concreta de cómo es la estructura interna de una memoria principal. En la siguiente imagen puedes observar su aspecto:



La memoria principal (no así los discos duros, como veremos más adelante) está formada por una serie de tableros puestos en paralelo que tienen una estructura matricial, es decir, están compuestos de miles de bits cuya posición viene dada por una celda y una memoria. Cada uno de estos bits puede estar encendido (un 1) o apagado (un 0), almacenando así la información en código binario. Por tanto, dependiendo de a qué tablero queramos acceder, y a qué posición del mismo, nuestro ordenador sufrirá unos ciertos tiempos de retardo o latencia.

La latencia viene dada por el tiempo que consume el acceso a cada uno de los bits de la memoria. En la realidad, la situación es un poco más complicada, y el tiempo de latencia viene definido por 4 valores distintos (observa por ejemplo en http://www.kingston.com/hyperx/products/khx_ddr3.asp los valores de los tiempos de latencia para algunas de sus últimas memorias principales, del tipo DDR3) y se miden en ciclos, en lugar de en segundos (o nanosegundos).

Los cuatro tiempos de latencia se pueden definir como:

- CAS: tiempo que tarda la memoria en colocarse sobre una celda o memoria.
- RAS: tiempo que tarda la memoria en colocarse sobre una fila.
- ACTIVE: tiempo que tarda la memoria en activar un tablero cualquiera.
- PRECHARGE: tiempo que tarda la memoria en desactivar un tablero.

Estos tiempos se miden en ciclos de la frecuencia de la memoria. Si la memoria principal de un ordenador posee una frecuencia de 1'6 GHz, es decir, es capaz de completar "1,6 * 10⁹" operaciones o ciclos en un segundo, y el tiempo de latencia para una de estas operaciones es de "9", entonces la memoria requerirá 9 ciclos para completar tal operación. Pro tanto, en términos generales, cuanto menores sean estos valores, más rápido será el acceso a la información. En http://en.wikipedia.org/wiki/SDRAM_latency puedes encontrar (en inglés) una explicación a los distintos tiempos de latencia propios de una memoria RAM.

En general, conociendo los valores de latencia de la memoria principal, puedes modificar la BIOS de tu ordenador para que se adapte a esos tiempos y así optimizar el rendimiento del uso de la memoria. Como los tiempos de latencia miden tiempo de acceso a distintas celdas (filas, columnas) de la memoria, cuanto menores sean los mismos mayor será la velocidad y rendimiento de la misma.

Es importante retener que la velocidad de acceso a la memoria principal es de varios órdenes de magnitud menor que la velocidad de operación de la UCP, y por tanto entre ambas existe una falta de sincronización que requiere de tiempos de espera (o interrupciones) en el funcionamiento de la UCP para poder acompasarse a la velocidad de trabajo de la UCP. El acceso a la memoria RAM es uno de los cuellos de botella en el rendimiento de un ordenador (incluso si su velocidad ha aumentado a gran velocidad a lo largo de la mejora de la tecnología).

El siguiente tipo de memoria que deberíamos abordar en esta sección es las unidades externas (o discos duros). Las mismas pueden ser consideradas desde un punto de vista formal como unidades de entrada/salida, y por tanto empezaremos por introducir la noción de unidades de entrada/salida, para luego estudiar los dispositivos de almacenamiento masivo como un tipo de estas últimas.

Unidades de Entrada/Salida:

Un ordenador hace uso de dispositivos que lo comunican con el mundo exterior. Estos dispositivos permiten la entrada de información (como por ejemplo un teclado, un ratón o un micrófono), la salida (caso de una impresora o altavoces) o ambos procesos simultáneamente (como un disco, una unidad de memoria externa o una pantalla táctil). Normalmente, a todos ellos se les llama dispositivos periféricos o de entrada/salida. El objetivo de la Unidad de E/S es la conexión y adaptación de la unidad central de proceso o de la memoria principal con cualquier tipo de dispositivo periférico para que la comunicación se realice de una manera estandarizada, independientemente del dispositivo de que se trate. De este modo, el procesador (o UCP) pueda interactuar con los dispositivos de E/S de modo similar a como lo hace con la memoria propia, tanto para operaciones de lectura como de escritura (aunque, posiblemente, mucho más despacio, desde el punto de vista de la velocidad de la UCP).

La comunicación de los dispositivos de E/S no se realiza única y exclusivamente con el procesador. Puede haber situaciones en las que los periféricos deban interactuar con la memoria.

Existen distintas formas de clasificar los dispositivos de entrada/salida que interactúan con un ordenador. Uno de ellos es por la forma de almacenar la información de los dispositivos periféricos. Con respecto a este criterio, los dispositivos de entrada/salida pueden ser de bloque o de carácter.

- Dispositivos de bloque: son aquellos con los cuales la comunicación con la UCP o con la memoria principal se realiza en forma de bloques de datos indivisibles (de tamaño mayor de 1 byte). Por ejemplo, como veremos más adelante, la comunicación con un disco duro se hace siempre por medio de sectores que ocupan (generalmente) 512 bytes.
- Dispositivos de carácter: se comunican con la UCP por medio de bytes sueltos. Los teclados o impresoras permiten este tipo de comunicación.

Dependiendo del tipo de comunicación que haya entre el procesador y el dispositivo correspondiente de E/S se distinguen las siguientes técnicas para llevar a cabo operaciones de E/S:

- E/S programada: el procesador ejecuta un programa y encuentra una orden que indica una operación de E/S. Ejecuta esa orden y genera un mandato al módulo de E/S apropiado. El módulo de E/S lleva a cabo la operación y fija los bits correspondientes en el registro de estado de E/S, pero no avisa directamente al procesador. El procesador, de forma periódica, comprueba el estado del módulo de E/S hasta que encuentra que se ha completado la operación. Si debemos repetir las operaciones de entrada o salida múltiples veces, al final estaremos perdiendo una gran cantidad de tiempo a la espera de que el procesador realice las comprobaciones pertinentes.

- E/S dirigida por interrupciones: A diferencia de la E/S programada, donde la comunicación siempre tenía lugar desde el procesador hacia el módulo de E/S, en la E/S dirigida por interrupciones la comunicación tiene lugar en ambas direcciones. Si el procesador quiere realizar una operación de E/S sobre un periférico, manda la solicitud correspondiente. A continuación sigue ejecutando las órdenes que tenía programadas. Cuando el periférico está listo, envía una interrupción al procesador para indicarle al mismo que ya está preparado. De este modo, por medio de la comunicación bidireccional, se evita que el procesador tenga que realizar comprobaciones periódicas de las operaciones de E/S y además se consigue que no se demore más de lo necesario cuando una operación de E/S ha sido completada.

- Acceso directo a memoria: la E/S dirigida por interrupciones, aunque más eficiente que la E/S programada, todavía requiere de la intervención activa del procesador para completar sus operaciones. Por lo tanto, tanto la E/S programada como la dirigida por interrupciones sufren de los siguientes dos inconvenientes. En primer lugar, la tasa de transferencia de E/S estará limitada por la velocidad con la que el procesador puede comprobar el estado de un dispositivo. En segundo lugar por cada operación de E/S se deben ejecutar varias instrucciones en el procesador. Una forma de evitar estos inconvenientes es por medio de la técnica de E/S conocida como acceso directo a memoria. La forma de llevar a cabo estas operaciones de E/S es la siguiente. El procesador genera un mandato al módulo DMA (direct memory access) con información sobre si es una operación de lectura o escritura, la dirección del dispositivo de E/S involucrado, la posición inicial de memoria en la que se desea leer o escribir los datos, y el número de palabras que se pretende leer o escribir. A continuación, el módulo DMA se ocupará de la operación tomando el control del bus para transferir los datos hacia la memoria o desde ella.

La forma de interactuar de los dispositivos de E/S con el procesador (con la UCP) o con la memoria es por medio de los "controladores de dispositivos". Podríamos decir, de forma genérica, que los controladores de dispositivos actúan de traductores entre el lenguaje propio del dispositivo de E/S (una Webcam, unos altavoces...) y el lenguaje máquina que pueda entender el conjunto "UCP + memoria principal". Esto da lugar a otra forma de definir los dispositivos de E/S: la memoria principal junto a la UCP forman el "núcleo" del un dispositivo, cualquier otro dispositivo que interactúe con los mismos se considera de E/S (y necesitará de un controlador que le posibilite esa comunicación o interacción).

Elementos de interconexión:

La forma más usual de interconectar los elementos en un computador es a través de buses o a través de puertos. Podemos entender un bus como un camino por el que fluye la información y al que se encuentran "conectados" distintos dispositivos que lo utilizan. La información se transmite por

señales eléctricas con dos estados diferentes (0 / 1). Dependiendo de su método de envío de información se distinguen dos tipos de buses:

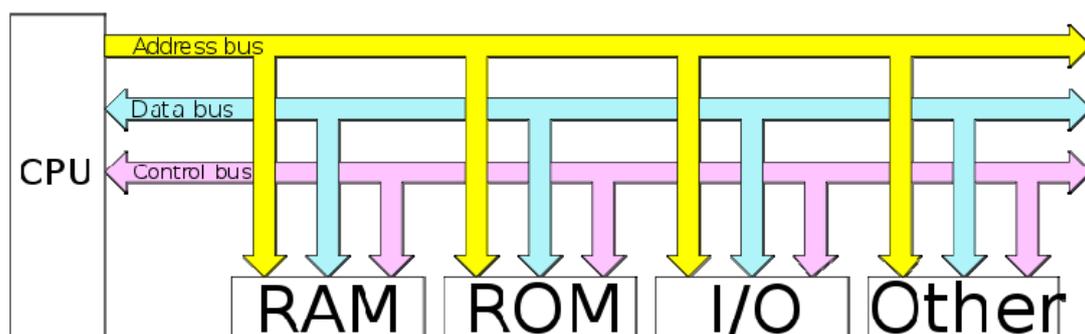
- Si el bus consta de varias líneas paralelas, cada una de las cuales es capaz de transportar un bit, hablaremos de un bus paralelo. En función de su número de líneas hablaremos de un ancho de bus de 8, 16, 32... bits. Esta anchura determina la cantidad de bits que se pueden transportar simultáneamente por él. Su funcionamiento es bastante simple y fueron los más comunes en los primeros tiempos de la informática para comunicar dispositivos de E/S y el ordenador. Su funcionamiento es bastante simple y fueron los más comunes en los primeros tiempos de la informática para comunicar dispositivos de E/S y el ordenador.
- Si el bus consta de una única línea de comunicación por la cual los datos son enviados bit a bit lo denominaremos bus serie. En este caso, el mismo está formado por pocos conductores que trabajan a frecuencias muy altas. Además, pueden tener una lógica muy compleja que requiere de los mismos gran poder de cómputo. Hoy en día su funcionamiento se está imponiendo, y son de uso habitual para conexiones USB o Firewire, así como para incluso las conexiones dentro de la propia placa base.

Otra posible división de los buses de un ordenador es dependiendo de los elementos que interconecta, entre buses locales y buses externos. Entre los buses locales están aquellos que interconectan los dispositivos locales (por ejemplo, dentro de la propia placa base, la memoria interna con el procesador, las distintas partes del procesador, la tarjeta gráfica). Los buses externos controlan las conexiones de la placa base con dispositivos de E/S (monitor, impresoras...).

El rendimiento de un bus se mide por dos parámetros, la cantidad de información que es capaz de transferir y la cantidad de energía que consume.

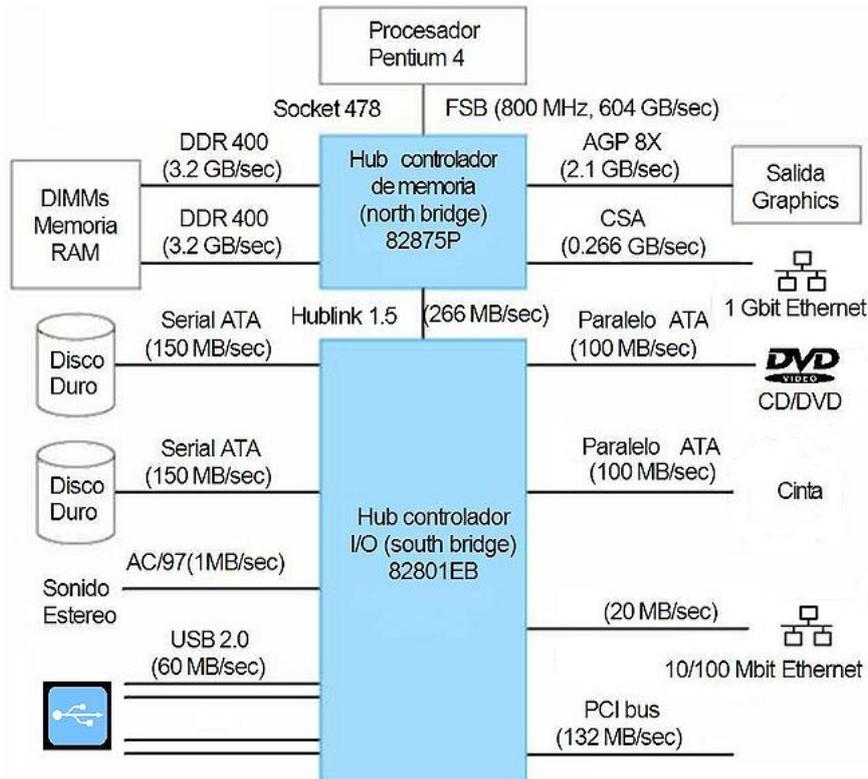
Durante mucho tiempo, los procesadores se diseñaban con un bus principal, conocido como front-side bus, que era de tipo paralelo y que estaba dividido en 3 tipos de líneas diferenciadas en función de la información que se transmitía por ellas:

- BUS de Direcciones: se envían por él direcciones de memoria.
- BUS de Control: se transmiten por él instrucciones de control para los distintos elementos.
- BUS de Datos: por él circulan datos o cualquier contenido susceptible de almacenarse en memoria (también instrucciones).



Los buses front-side han caído en desuso y sido reemplazados por buses que ofrecen mayor rendimiento (mayor velocidad de transferencia de datos), como la tecnología HyperTransport de AMD (que permite funcionamiento en serie y en paralelo), o la tecnología QuickPath de Intel, presente ya en sus últimos procesadores (I7, I9...).

En la siguiente imagen podemos observar la estructura básica de un procesador (en este caso un Pentium 4) junto con los distintos buses, sus frecuencias de comunicación, sus tipos (serial, paralelo) y periféricos que lo componen:



Si hemos dicho que los buses son las "líneas de comunicación" que permiten la transferencia de datos entre los distintos dispositivos que forman un ordenador, los puertos (o quizá, mejor traducidos como puertas) son los elementos físicos (a nivel hardware) o de tipo software que facilitan la conexión de los dispositivos a los buses. Algunos de los puertos propios de un ordenador son aquellos que permiten la conexión de ratones, monitores externos, impresoras...y dependiendo del rendimiento o velocidad que deban ofrecer, de nuevo serán divididos en paralelos o seriales (de serie).

Veremos en lo que queda del tema algunos otros elementos propios de la arquitectura de los computadores que no aparecían en el diagrama inicial que presentamos de la arquitectura de Von Neumann, pero que sin embargo es importante destacar por su relevancia. En realidad, y dentro del diagrama anterior, el disco duro lo podríamos identificar con un dispositivo externo de E/S, que nos permitirá almacenar información de manera no volátil (no así como la memoria principal).

El disco duro (HD - Hard disk)

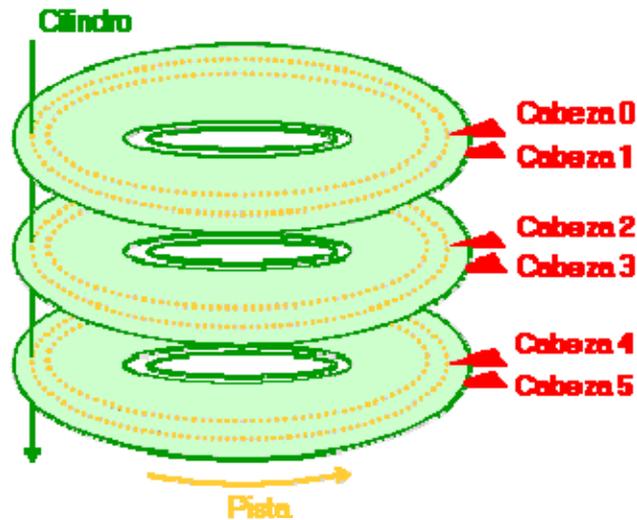
La información contenida en esta sección es relativa a los discos duros "mecánicos", formados por una serie de platos metálicos apilados que giran a gran velocidad para poder leer la información contenida en las distintas pistas. Hoy en día se están empezando a extender y popularizar (debido a su abaratamiento y mejor rendimiento) las unidades de estado sólido (SSD), como por ejemplo las memorias de tipo Flash para almacenar la información. Sin embargo, todavía la diferencia de precio es lo suficientemente grande entre unas y otras como para que muchos dispositivos que ofrecen una capacidad de almacenamiento masivo se decanten por los discos duros convencionales. En la página http://en.wikipedia.org/wiki/Solid-state_drive#Comparison_of_SSD_with_hard_disk_drives puedes encontrar una lista detallada de las diferencias entre los discos duros tradicionales y las memorias SSD, y en particular la diferencia de precio (por GB) entre unas y otras.

Desde un punto de vista formal un disco duro mecánico está compuesto de tres elementos básicos:

- Un conjunto de platos de superficie magnética sujetos a un eje que gira a miles de rpm (revoluciones por minuto).
- Un conjunto de cabezas de lectura/escritura sujetas sobre un brazo móvil, que es capaz de moverlas desde el centro hasta el extremo del plato. Cada cabeza lee sobre una superficie distinta.
- Un bus de datos que conecta las cabezas con la electrónica de la unidad. Como sólo una cabeza puede leer o escribir a la vez, todas ellas están conectadas al mismo bus, que será el que transfiera la información hacia y desde el procesador.

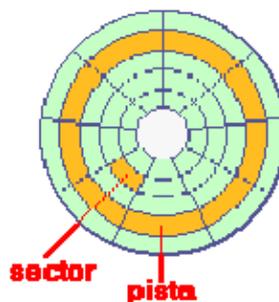
Por su construcción, la información se va guardando en "líneas concéntricas" llamadas pistas. La pista es pues la "línea" que puede leer una cabeza sin moverse.

Como todas las cabezas van unidas al mismo brazo móvil, todas se posicionan en la misma pista pero cada una en una superficie diferente. A cada conjunto de pistas que se pueden leer sin mover el brazo se le llama cilindro.



En un disco magnético podemos distinguir 3 niveles en el manejo de la información. Los dos primeros corresponden con lo que se conoce como "estructura física" del disco, mientras que el tercero es la "estructura lógica" del mismo:

- El primer nivel determinaría cómo cada bit se graba o lee mediante un cambio de magnetismo. Sólo vamos a mencionar que, a diferencia de lo visto en la memoria interna, la superficie magnética del disco no tiene "celdas individuales" y además las superficies están en movimiento muy rápido lo que hace el proceso más complejo.
- El segundo nivel: por el funcionamiento del disco (elementos que rotan a gran velocidad y piezas mecánicas para mover las cabezas), sería casi imposible ir leyendo la información a base de "palabras de memoria". Por ello, los discos son "dispositivos de bloques". Se lee un bloque grande de bits cada vez y se transfieren a la memoria principal. En la mayoría de los discos estos bloques son de tamaño fijo y se determinan en el momento de la fabricación mediante la grabación de marcas de comienzo, fin... Estos bloques se llaman sectores. Físicamente responden a porciones angulares constantes y, desde el punto de vista de la información, suelen contener 512 Bytes. Notemos que, a pesar de que las pistas interiores son mucho más pequeñas que las exteriores, todas tienen el mismo número de sectores y, por tanto, la misma cantidad de información (lo puedes comprobar con el programa "part240" que usamos en la práctica 2 sobre "Windows 98").



El número de cilindros y de sectores por pista varía de unos discos a otros. La información que se graba en el disco como marca para la determinación de las pistas, el comienzo, el fin o la numeración de los sectores... se llama formateo a bajo nivel.

El tamaño (cantidad de información) de un disco, vendrá determinado por 4 factores que hay que conocer: Número de cilindros (o pistas), número de cabezas, número de sectores por pista y tamaño de cada sector. Por ejemplo, un disco con 80 pistas, 18 sectores/pista, 512 bytes/sector y un plato de 2 superficies, tiene una capacidad de $80 \times 18 \times 512 \times 2 = 1440 \text{ KB}$ (se corresponde a un disquete de 3"1/2 de alta densidad).

La localización de cualquier información en el disco se hace mediante 3 coordenadas: número de cabeza, número de cilindro y número de sector.

- El tercer nivel: tiene que ver con la forma en que los archivos o ficheros se estructuran y se almacenan y nos referimos a él como "el sistema de archivos". El sistema de archivos se define "formateando" y hablaremos de él más adelante como parte de los Sistemas Operativos. A diferencia de los dos niveles anteriores, este tercer nivel viene determinado por el SO a utilizar y, en consecuencia, es definible por el usuario. Conviene incidir en la idea de que el sistema operativo, por medio del formateo, nos permite "olvidarnos" de la estructura física del disco duro (¿imaginas la tarea de tener que decirle a un programa qué sectores del disco duro debe usar para guardar un archivo, o recuperar un fichero alojado en determinados sectores?).

De lo descrito arriba hay que tener presentes algunas cuestiones fundamentales:

1. La arquitectura física descrita responde a una realidad "cuasi" histórica. La mayor parte de los discos duros actuales disponen exclusivamente de un plato y 2 cabezas porque la tecnología actual permite la integración necesaria y la precisión de lectura/escritura para ello. También se ha vencido, por ejemplo, la limitación de que todas las pistas deban tener el mismo número de sectores, encontrando así discos duros con número variable de sectores por pista (mayor número en las pistas exteriores, menor en las interiores, pero observa la complicación tecnológica que ello conlleva a la hora de leer cada uno de esos sectores, ya que se corresponden a sectores circulares también de distinto tamaño).
2. La idea de que un disco es un dispositivo de bloques porque tiene muchos componentes "mecánicos" y velocidades de rotación que hicieron y siguen haciendo que el acceso no se pueda hacer de una manera más precisa de forma eficiente, determinó que el tercer nivel del que hemos hablado (los sistemas de ficheros) haya estado ligeramente influido por esta circunstancia.


```

0130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
01A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
01B0 00 00 00 00 00 00 00 00 00 00 00 00 80 01 .....
01C0 01 00 0B 7F BF FD 3F 00 00 00 C1 40 5E 00 00 .....?....@^...
01D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
01E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
01F0 00 00 00 00 00 00 00 00 00 00 00 00 55 AA .....U.
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

```

En la imagen superior podemos observar un ejemplo de fichero MBR con su codificación en hexadecimal. Dentro del mismo, en la parte en verde nos encontramos con código ejecutable (los primeros 139 bytes en este caso). A continuación, en color morado, los siguientes 80 bytes corresponden a mensajes de error que se pueden producir al cargar el MBR. Hasta el byte 446 nos encontramos con "0's". En realidad los primeros 446 bytes del MBR se reservan para código ejecutable, los 64 siguientes para la tabla de particiones, y los dos últimos son la firma de unidad. En este caso, como el código máquina y los errores sólo ocupan 219 bytes, los siguientes 227 han quedado vacíos. Los 64 bytes de color rosa contienen información referente a las particiones (primarias) del disco. Como de cada partición se guardan 16 bytes de información, y sólo disponemos de 64 bytes para almacenar particiones, nos encontramos con la citada limitación de 4 particiones primarias en cada disco duro.

Un único disco físico puede ser dividido en varios "discos lógicos" llamados particiones. Cada partición será siempre una zona contigua de disco.

Las particiones nos permiten manejar discos grandes como si se tratasen de varios discos más pequeños y disponer de varios sistemas operativos con distintos "sistemas de ficheros" en un mismo equipo. Sus beneficios son múltiples ya que nos permiten estructurar la información (separar sistema, programas, datos, archivos de usuarios...), hacer mejor el sistema de copias de seguridad, la restauración, establecer cuotas de ocupación,...

Las particiones pueden ser de dos tipos: primarias o lógicas. Particiones primarias podemos encontrar un máximo de 4. Las particiones lógicas se definen dentro de una partición primaria especial denominada partición extendida.

De las cuatro entradas en la tabla de particiones del MBR puede que no esté utilizada ninguna (disco duro sin particionar, tal y como viene de fábrica) o que estén utilizadas una, dos, tres o las cuatro entradas. En cualquiera de estos últimos casos (incluso cuando sólo hay una partición), es necesario que en la tabla de particiones figure una de ellas como partición activa. La partición activa es aquella a la que el programa de inicialización (situado en el Master Boot Record) cede el control al arrancar; en ella normalmente se encontrará el sistema operativo con el que queremos trabajar o tal vez un pequeño programa gestor de arranque (por ejemplo, LILO o GRUB) que nos permitirán hacer una elección de qué sistema operativo queremos arrancar y redirigirán el arranque a otro lugar.

Estructura del MBR

Como hemos indicado, los ordenadores, al arrancar y leer el MBR esperan un conjunto con 512 bytes con la siguiente estructura:

		Primer sector físico del disco (pista cero)
512	446	Código máquina (gestor de arranque)
Bytes	Bytes	
	64 Bytes	Tabla de particiones
	2 Bytes	Firma de unidad arrancable ("55h AAh" en hexadecimal)

Puedes encontrar más información sobre el mismo en <http://es.wikipedia.org/wiki/MBR>.

Utilidades:

- En el siguiente enlace puedes descargar un pequeño programa para Windows que te permite leer cualquier sector de disco, incluido el MBR, y hacer de él una copia de seguridad. Es sumamente interesante que, si lo utilizas, guardes el contenido en un fichero y lo edites en hexadecimal, por ejemplo con Pspad: [hdhacker.zip](#)
- En este otro enlace tienes otra pequeña utilidad para Windows que te permite observar el contenido del disco y de la memoria con gran cantidad de posibilidades: [HxD.es.zip](#)