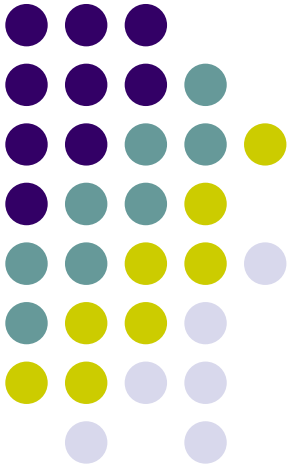
Tema 1. Informática Básica

Introducción

1.1 Informática Básica

1.2 Codificación de la Información

1.3 El ordenador. Fundamentos estructurales y de funcionamiento

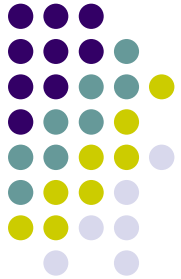


Jesús María Aransay Azofra

Informática

Universidad de La Rioja 2011/2012

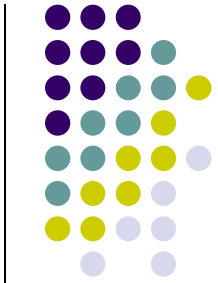
Introducción



Objetivos del tema:

- Definir las nociones de Informática y sistema informático (Sección 1.1)
- Comprender algunos de los fundamentos de la codificación digital de la información (Sección 1.2)
- Estudiar algunas de las partes fundamentales de un sistema informático (Sección 1.3)

1.1 Informática básica



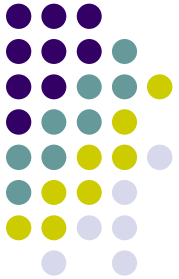
Informática:

conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento *automático* de la *información* por medio de ordenadores

Sistema Informático:

hardware + software + “factor humano”

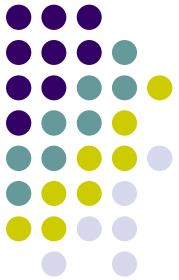
1.2 Codificación de la información



1.2.1 Señal digital

1.2.2 Codificación de caracteres

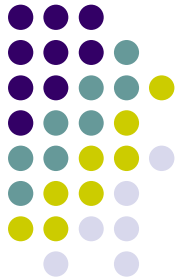
1.2.1 Señal digital



- Basada en almacenar un número finito de estados
- Bit: binary digit. Almacena 2 estados, “0” ó “1”
- Octeto o byte: 8 bits, $2^8 = 256$ estados

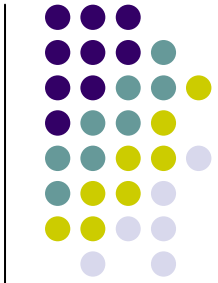
Generalmente, los sistemas informáticos siempre trabajarán como mínimo con bytes, ya que los bits no permiten almacenar cantidades de información significativas

1.2.1 Señal digital



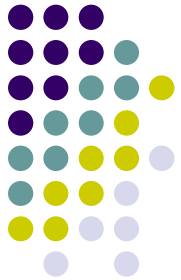
- Otras unidades útiles de almacenamiento de información:
  - KB: KiloByte = 1024 bytes (2^10 bytes)
  - MB: MegaByte = 1024 KB (2^20 bytes)
  - GB: GigaByte = 1024 MB (2^30 bytes)
  - TB: TeraByte = 1024 GB (2^40 bytes)
- Es importante distinguir las anteriores unidades de:
  - Kb: Kilobit = 1.000 bits (aprox. 125 bytes)
  - Mb: Megabit = 1.000 Kb (aprox. 125 KB)
  - Gb: Gigabit = 1.000 Mb (aprox. 125 MB)

1.2.1 Señal digital



- Lo que signifique la información contenida en un bit (o byte) depende del tipo de archivo en el que dicho bit o byte se encuentre (puede representar un color, en un archivo de imagen, o un carácter, en un fichero de “texto plano”).
- Los ficheros de “texto plano” permiten almacenar información (de tipo textual) que no contiene mucho procesamiento o información relativa a su formato (tamaños o tipos de letra, márgenes…). Se usan ampliamente para almacenar código de programas, como ficheros de configuración, para almacenar datos…
- En un fichero de texto plano, es crucial conocer la “tabla de codificación” con respecto a la cual el mismo ha sido guardado (y con la cual debe ser abierto)

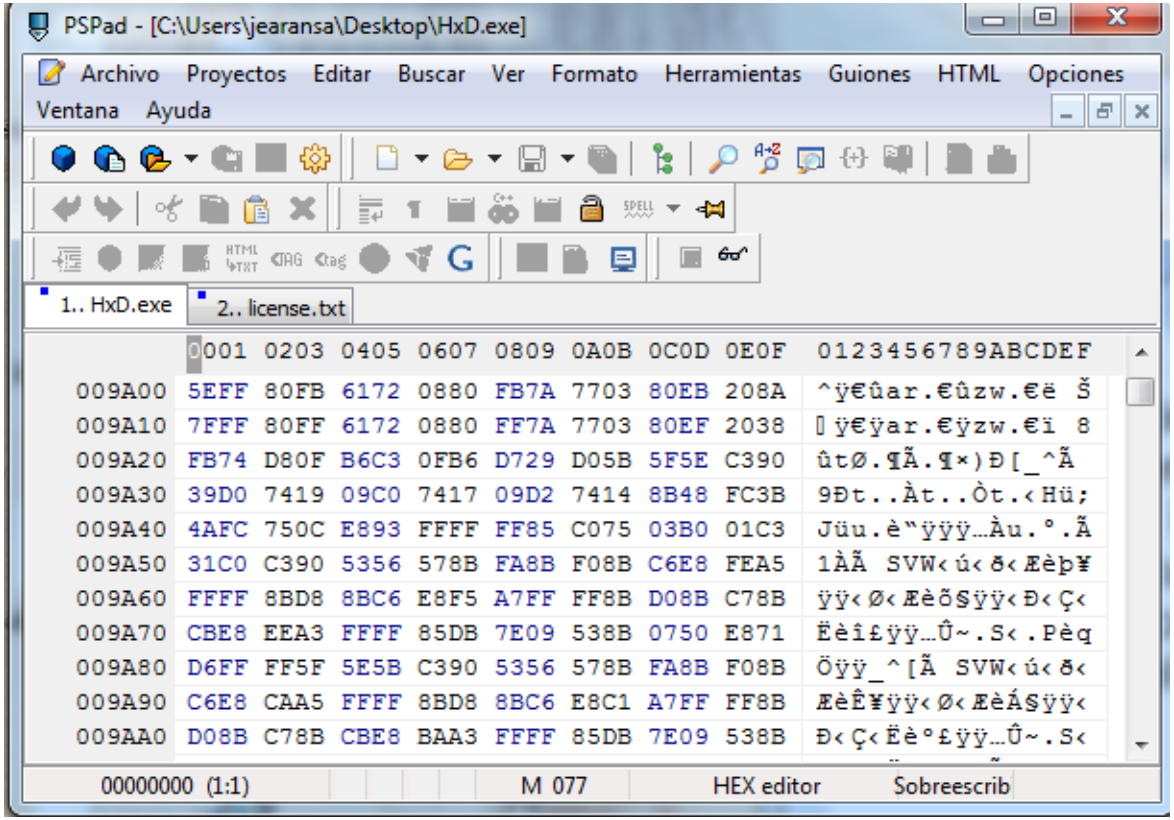
1.2.1 Señal digital



- A partir de la anterior división, solemos hablar de:
  - Archivos (o ficheros) binarios: cualquier fichero que solo contiene información codificada en binario (en forma de bits)
  - Archivos (o ficheros) de texto: un fichero binario en el que cada byte representan un carácter “textual” (el fichero solo contiene “texto plano”)

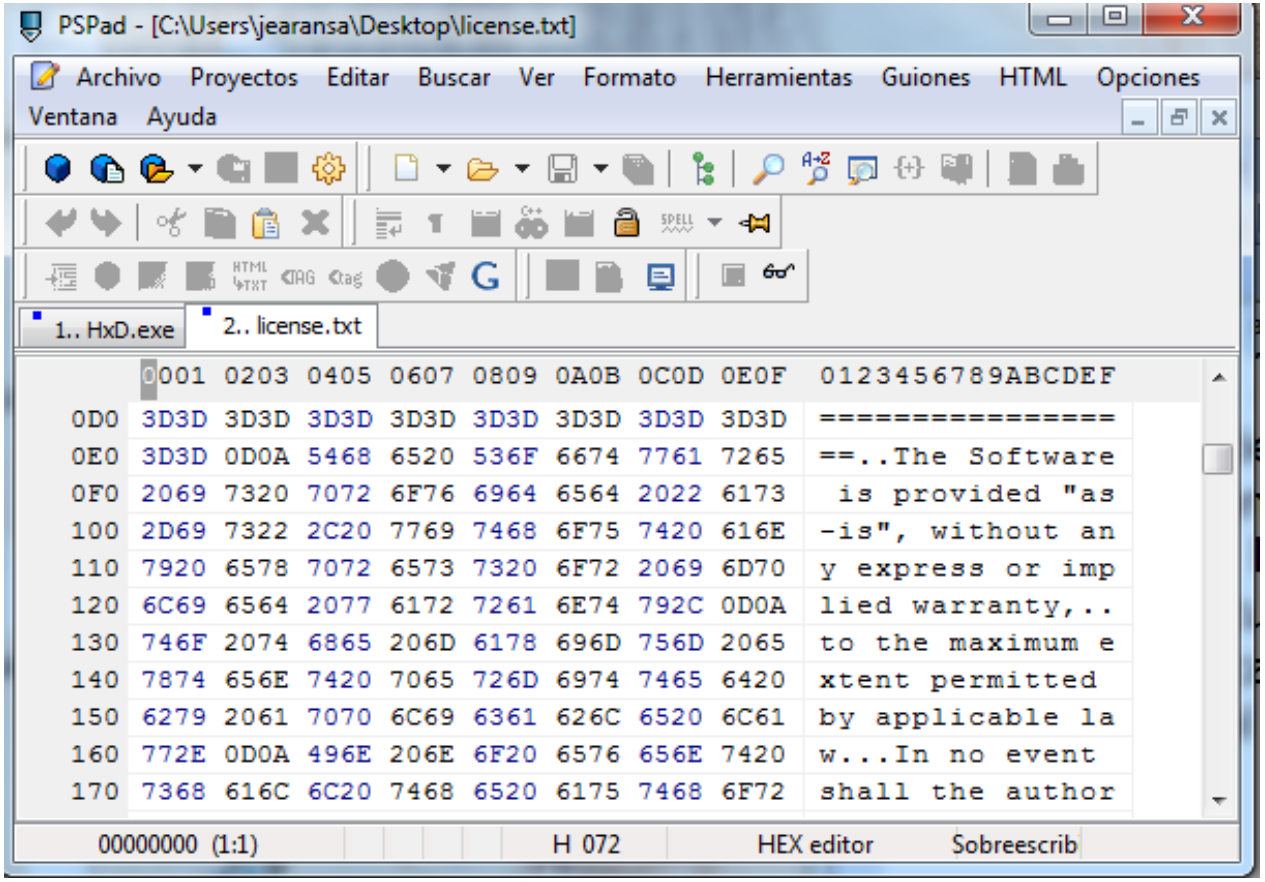
1.2.2 Codificación de caracteres

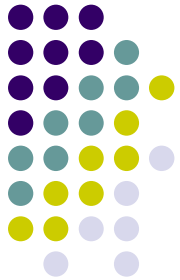
- Archivos binarios (también representados en formato binario)



1.2.2 Codificación de caracteres

- Archivos de texto plano (sin formato)





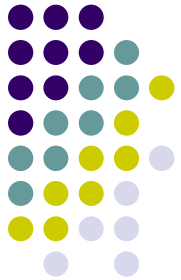
Archivos de texto

Cada carácter del fichero es almacenado en memoria por una secuencia de “0”s y “1”s (por lo general, de uno o varios octetos).

¿Cómo es el proceso de almacenamiento de un carácter en un fichero de texto plano en memoria?

1. El usuario teclea y guarda : “A”
2. El ordenador comprueba: ¿cuál es la tabla de caracteres activa? ¿Y su codificación de caracteres respectiva?
3. El ordenador calcula: ¿cuál es la codificación del carácter “A” en esa regla de codificación?
4. El ordenador guarda en memoria la codificación (los bytes) del carácter “A” con respecto a la tabla de caracteres activa

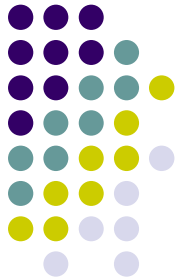
Y viceversa…



¿Cómo es el proceso de lectura de un fichero de texto desde memoria?

1. Abrimos un fichero de texto plano con un programa (bloc de notas, PsPad, Notepad++…)
2. El ordenador comprueba la tabla de caracteres que usa el programa por defecto, o la infiere del archivo
3. El ordenador lee la primera secuencia de bytes (no de bits)
4. El ordenador comprueba a qué carácter corresponde esa secuencia en la codificación de esa tabla de caracteres, y muestra ese carácter

¿Cuántas cosas pueden ir mal en el anterior proceso?

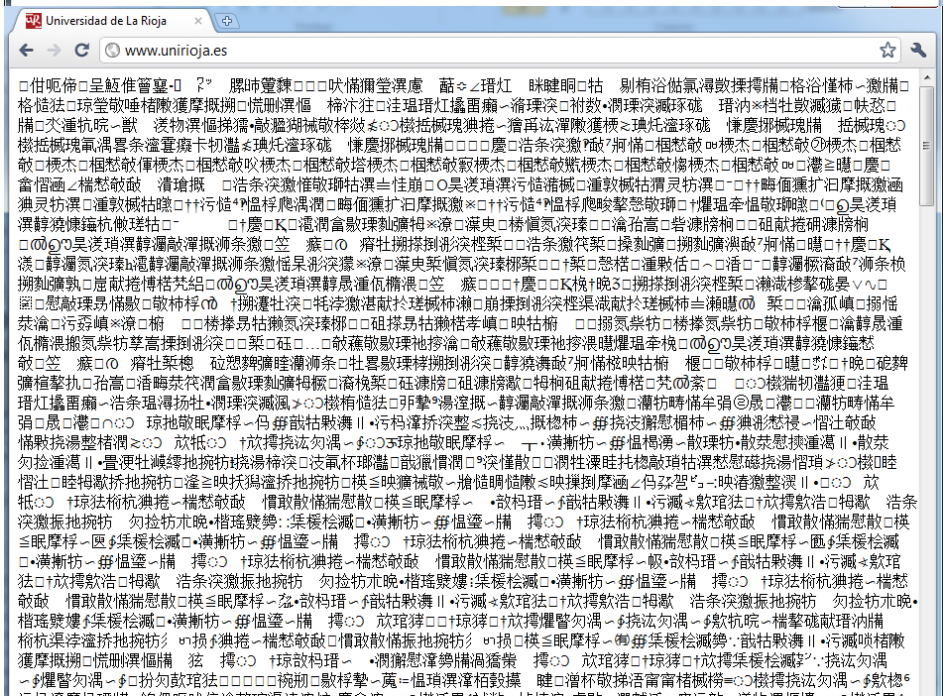


A nivel teórico unas cuantas, en la práctica generalmente una:

- que la tabla de caracteres que usamos al guardar un archivo NO sea la misma que la que usamos al abrirlo (porque hemos cambiado de programa, de ordenador…)

Otro problema, al menos a nivel teórico:

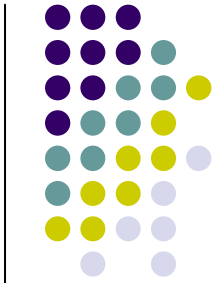
- que la regla (o algoritmo) de codificación haya cambiado (UTF-16 Big Endian vs. UTF-16 Little Endian)



www.unirioja.es abierta con la misma codificación con que se guardó (ISO-8859-1)

www.unirioja.es abierta con otra codificación distinta de la que se guardó (UTF-16LE)

Algunas definiciones (no estándar):

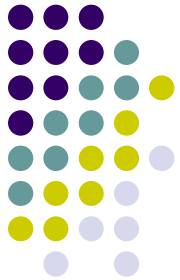


Repertorio de caracteres: caracteres que una codificación pretende almacenar
Ejemplo: {a, e, i, o, u, A, E, I, O, U, á, é, í, ó, ú}

Tabla de caracteres: una asignación entre los caracteres de un repertorio y números enteros positivos
Ejemplo: {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}

(Regla de) Codificación de caracteres: algoritmo que a cada número de la tabla de caracteres le asigna su representación en binario (y por tanto la que se almacena en memoria)
Ejemplo: {1 = 00000001, 2 = 00000010…}

Algunas definiciones (no estándar):

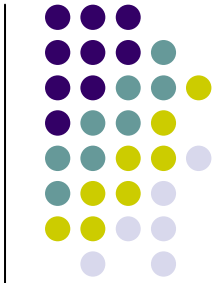


Longitud de palabra: espacio (generalmente en bytes) que ocupa cada carácter al ser codificado

Observaciones:

- si una codificación tiene una longitud de palabra de 1 byte, el número máximo de caracteres que podrá alojar será 2^8 = 256; si fuera 2 bytes, el máximo sería 2^16 = 65536
- si un carácter ocupa un byte en una codificación, no quiere decir que todos los caracteres ocupen 1 byte en esa codificación (veremos más adelante ejemplos de este caso, en UTF8 y UTF16)

Notas



Siguiendo con el ejemplo anterior, [“a” = 1 = 00000001], lo podemos leer como ‘El carácter “a” cuya posición en la tabla de caracteres es 1 se codifica (y almacena) en binario como ‘00000001’. Su longitud de palabra es 1 byte (8 bits), al menos para la “a”

La posición en la tabla de codificación y la codificación en binario **no tienen por qué coincidir** (por ejemplo, la “O” en la tabla ASCII ocupa la posición 79, y su codificación en binario es “11001111” = 207). De ahí la importancia de conocer no sólo la tabla de caracteres, sino también la codificación o regla correspondiente

Notas

Cómo convertir a binario o hexadecimal un número entero:

Base 2. Símbolos usados: {0,1}

135 = 67 * 2 + 1
67 = 33 * 2 + 1
33 = 16 * 2 + 1
16 = 8 * 2 + 0
8 = 4 * 2 + 0
4 = 2 * 2 + 0
2 = 1 * 2 + 0
1 = 0 * 2 + 1

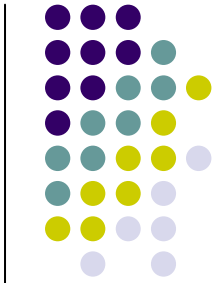
135 = 10000111 (cogemos los restos de dividir por 2 en orden inverso)

Base 16. Símbolos usados: {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}

143 = 8 * 16 + (15 = F)
8 = 0 * 16 + 8

143 = 8F (cogemos los restos de dividir por 16 en orden inverso)

Algunos sistemas de codificación



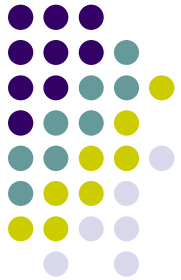
ASCII (American Standard Code for Information Interexchange)

- Origen occidental
- Tiene su origen en los años 60
- Repertorio de caracteres con sólo 128 (2^7) elementos
- Los 32 primeros caracteres son “no imprimibles” (caracteres de control)
- Del 33 hasta el 128 son imprimibles
- Su longitud de palabra suele ser de 8 bits (1 byte), por lo que deja un bit libre

Tabla ASCII

Bits b_4 b_3 b_2 b_1	Row ↓ / Column →	b_7 b_6 b_5 = 000	001	010	011	100	101	110	111
		0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FC	,	<	L	\	l	/
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

Codificación ASCII



1. Tomamos la posición en la tabla ASCII del carácter (g = 103)
2. Conversión a binario (en 7 dígitos)

   103 = 51 * 2 + 1; 51 = 25 * 2 + 1; 25 = 12 * 2 + 1

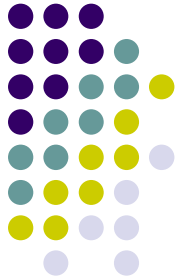
   12 = 6 * 2 + 0; 6 = 3 * 2 + 0; 3 = 1 * 2 + 1

   1 = 0 * 2 + 1

   Representación en binario: _1100111
3. Añadimos el **bit de paridad** (1 si hay un número impar de unos, 0 en otro caso) en el octavo dígito:

g = **1**1100111

ISO – 8859 - …



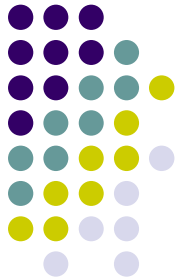
- Repertorio de 256 caracteres (8 bits)
- Los caracteres 1 a 128 coinciden con los caracteres imprimibles de ASCII
- Contiene variantes para zonas geográficas:
  - ISO-8859-1: ISOLatin1 (Europa Occidental)
  - ISO-8859-2: ISOLatin2 (Europa Oriental)
  - …
  - ISO-8859-15: ISOLatin9 (Europa, incluye €)
- Los caracteres 128 a 256 de ISO-8859-1 coinciden con los de Unicode (no así sus codificaciones, como veremos)

Tabla de codificación ISO-8859-1

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F
80																
90																
A0	NBSP 00A0	¡ 00A1	¢ 00A2	£ 00A3	¤ 00A4	¥ 00A5	¦ 00A6	§ 00A7	¨ 00A8	© 00A9	ª 00AA	« 00AB	¬ 00AC	- 00AD	® 00AE	¯ 00AF
B0	° 00B0	± 00B1	² 00B2	³ 00B3	´ 00B4	µ 00B5	¶ 00B6	· 00B7	¸ 00B8	¹ 00B9	º 00BA	» 00BB	¼ 00BC	½ 00BD	¾ 00BE	¿ 00BF
C0	À 00C0	Á 00C1	Â 00C2	Ã 00C3	Ä 00C4	Å 00C5	Æ 00C6	Ç 00C7	È 00C8	É 00C9	Ê 00CA	Ë 00CB	Ì 00CC	Í 00CD	Î 00CE	Ï 00CF
D0	Ð 00D0	Ñ 00D1	Ò 00D2	Ó 00D3	Ô 00D4	Õ 00D5	Ö 00D6	× 00D7	Ø 00D8	Ù 00D9	Ú 00DA	Û 00DB	Ü 00DC	Ý 00DD	Þ 00DE	ß 00DF
E0	à 00E0	á 00E1	â 00E2	ã 00E3	ä 00E4	å 00E5	æ 00E6	ç 00E7	è 00E8	é 00E9	ê 00EA	ë 00EB	ì 00EC	í 00ED	î 00EE	ï 00EF
F0	ð 00F0	ñ 00F1	ò 00F2	ó 00F3	ô 00F4	õ 00F5	ö 00F6	÷ 00F7	ø 00F8	ù 00F9	ú 00FA	û 00FB	ü 00FC	ý 00FD	þ 00FE	ÿ 00FF

Los caracteres 128 a 160 también son caracteres de control

DOS - ...

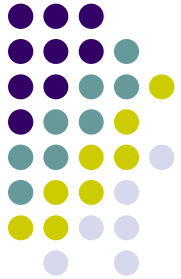


- Repertorios de 256 caracteres (8 bits)
- Los caracteres 33 a 128 coinciden con los caracteres imprimibles de ASCII
- Todos los caracteres (incluidos los de control) tienen una equivalencia gráfica en la consola de MSDOS
- Generalmente usada en la consola de MSDOS (chcp)
- Contiene variantes para zonas geográficas:
  - DOS-437: DOSUS (Estados Unidos)
  - DOS-850: DOSLatin1 (Europa Occidental)
  - … (http://en.wikipedia.org/wiki/Category:DOS_code_pages)

DOS - ...

Tabla de caracteres DOS-850, como se mostraría en una consola de MSDOS convencional:

Windows - …



- Repertorio de 256 caracteres (8 bits)
- Los 128 primeros caracteres coinciden con los caracteres imprimibles de ASCII, suelen ser compatibles con las ISO correspondientes
- Generalmente usada en aplicaciones Windows (80s y 90s)
- Contiene variantes para zonas geográficas:
  - Windows-1252: WinLatin1 (Europa Occidental)
  - Windows-1250, Windows -1251 (Europa Oriental)
  - …(http://msdn.microsoft.com/en-us/goglobal/bb964654)

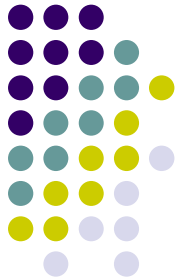
Tabla de caracteres Win-1252

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F
80	€ 20AC		‚ 201A	ƒ 0192	„ 201E	… 2026	† 2020	‡ 2021	ˆ 02C6	‰ 2030	Š 0160	‹ 2039	Œ 0152		Ž 017D	
90		‘ 2018	’ 2019	“ 201C	” 201D	• 2022	– 2013	— 2014	˜ 02DC	™ 2122	š 0161	› 203A	œ 0153		ž 017E	Ÿ 0178
A0	NBSP 00A0	¡ 00A1	¢ 00A2	£ 00A3	¤ 00A4	¥ 00A5	¦ 00A6	§ 00A7	¨ 00A8	© 00A9	ª 00AA	« 00AB	¬ 00AC	- 00AD	® 00AE	¯ 00AF
B0	° 00B0	± 00B1	² 00B2	³ 00B3	´ 00B4	µ 00B5	¶ 00B6	· 00B7	¸ 00B8	¹ 00B9	º 00BA	» 00BB	¼ 00BC	½ 00BD	¾ 00BE	¿ 00BF
C0	À 00C0	Á 00C1	Â 00C2	Ã 00C3	Ä 00C4	Å 00C5	Æ 00C6	Ç 00C7	È 00C8	É 00C9	Ê 00CA	Ë 00CB	Ì 00CC	Í 00CD	Î 00CE	Ï 00CF
D0	Ð 00D0	Ñ 00D1	Ò 00D2	Ó 00D3	Ô 00D4	Õ 00D5	Ö 00D6	× 00D7	Ø 00D8	Ù 00D9	Ú 00DA	Û 00DB	Ü 00DC	Ý 00DD	Þ 00DE	ß 00DF
E0	à 00E0	á 00E1	â 00E2	ã 00E3	ä 00E4	å 00E5	æ 00E6	ç 00E7	è 00E8	é 00E9	ê 00EA	ë 00EB	ì 00EC	í 00ED	î 00EE	ï 00EF
F0	ð 00F0	ñ 00F1	ò 00F2	ó 00F3	ô 00F4	õ 00F5	ö 00F6	÷ 00F7	ø 00F8	ù 00F9	ú 00FA	û 00FB	ü 00FC	ý 00FD	þ 00FE	ÿ 00FF

Los caracteres 128 a 160 ya no son de control

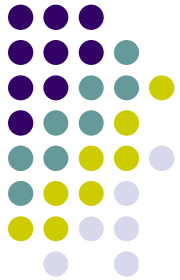
El resto de la tabla es idéntica a ISO – 8859 – 1

Unicode



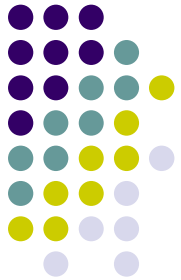
- Repertorio y tablas de más de 100.000 caracteres (diversas codificaciones!!)
- Compatible con ASCII (del 1 al 128)
- Compatible con ISO – 8859 – 1 (del 129 al 256)
- Estándar de facto: usado internamente en Windows (desde NT), Java, .NET, Unix, Mac OS X, KDE…
- No dispone de variantes, la idea es incluir cualquier carácter de cualquier idioma
- Dispone de varias codificaciones (UTF – 8, UTF – 16, UTF – 32)

Unicode



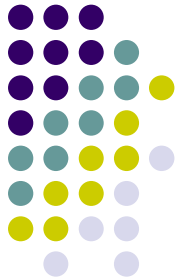
- Los caracteres del repertorio se organizan en “planos” que admiten 65.536 (2^16) caracteres
- Existen 17 planos, algunos de los cuales están todavía sin usar
- El primero plano es el BMP (Basic Multilingual Panel), con 65.536 caracteres, siendo los 256 primeros los de ISO – 8859 – 1
- Lista de planos: http://en.wikipedia.org/wiki/Unicode_plane
- Caracteres en BMP: http://en.wikipedia.org/wiki/Basic_Multilingual_Plane#Basic_Multilingual_Plane

UTF-8



- Codificación de Unicode de longitud variable (uno o más bytes)
- Los caracteres ASCII los codifica en un byte
- Para cualquier otro carácter usa 2 o más bytes (mirar los apuntes)
- A menudo añade al principio de los archivos el BOM (Byte Order Mark), marca que avisa de que el archivo está en formato UTF – 8

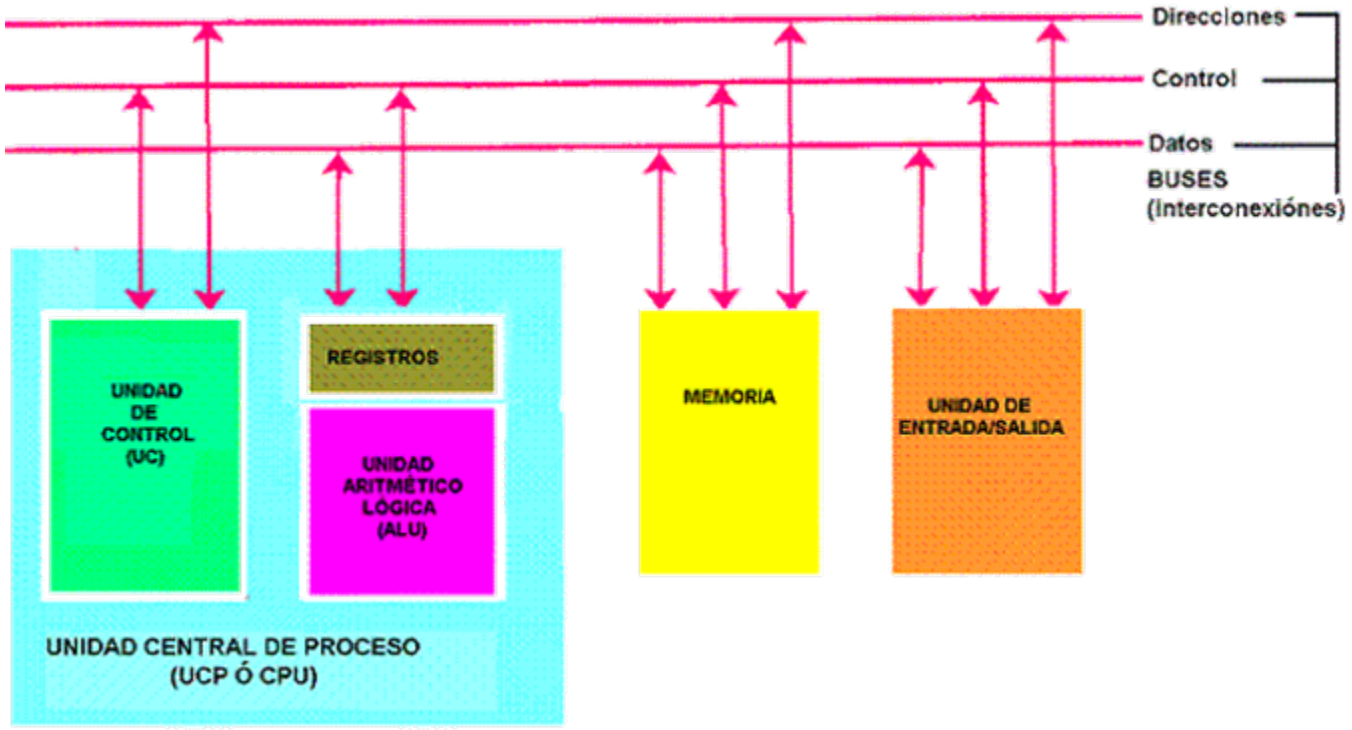
UTF-16



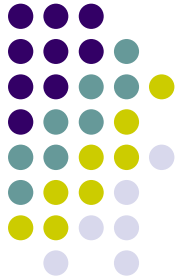
- Codificación de Unicode de longitud variable (2 o más bytes)
- Incompatible con archivos ASCII, UTF – 8, ISO – 8859 -, … (recordar el ejemplo de www.unirioja.es)
- Los caracteres en el BMP (los 4096 primeros) los codifica en 2 bytes (los demás en 4 bytes)

1.3 El ordenador. Fundamentos estructurales y de funcionamiento

Estructura de programa almacenado



UCP (unidad central de proceso)



- 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. Se utilizan para guardar direcciones o datos de memoria (MAR, MDR), de las instrucciones llevadas a cabo (IR) o de las que se han de ejecutar (PC)
- UAL (Unidad aritmético-lógica): se encarga de realizar las operaciones aritméticas o lógicas con los datos que se encuentren en los registros correspondientes
- UC (Unidad de control): dirige todo el proceso, leyendo del registro correspondiente la instrucción a ejecutar y generando las señales de control para que se realicen las operaciones pertinentes

UCP: componentes

Ejemplo de UCP con bus único

IR: Instruction register

PC: Program counter

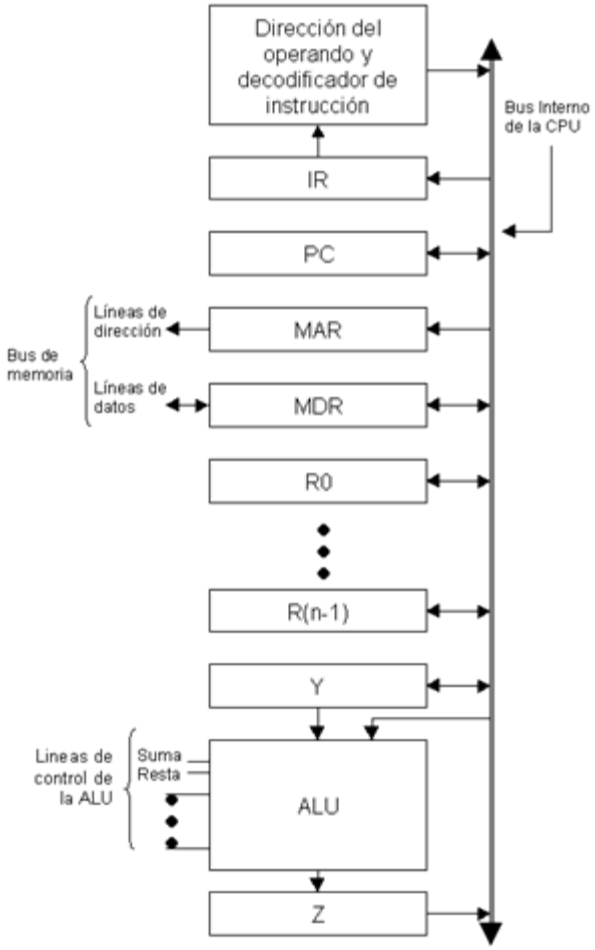
MAR: memory address register

MDR: memory data register

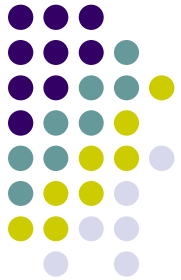
ALU: Unidad aritmético-lógica

R0…R(n-1): registros de aplicación especial

Y, Z: registros de uso interno y almacenamiento temporal

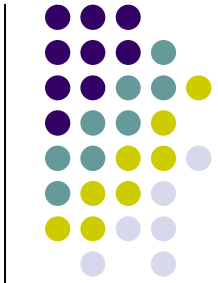


UCP: lenguaje máquina



- La UCP tiene un lenguaje propio de instrucciones que permiten llevar a cabo operaciones sobre memoria, datos… (se le suele llamar “lenguaje procesador”)
- Estas instrucciones “consumen” ciclos del procesador. La cantidad de ciclos (es decir, de instrucciones) que un procesador puede completar por segundo se mide en hercios (1,6 GHz = 1,6 * 10^9 ciclos por segundo)
- Las instrucciones de un procesador son específicas del mismo (x86, tarjetas gráficas, ARM…); por eso, mayor velocidad no siempre es sinónimo de mayor rendimiento (si el lenguaje máquina estás más optimizado…)

UCP: lenguaje máquina

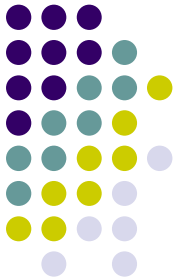


- http://en.wikipedia.org/wiki/X86_instruction_listings
- http://oopweb.com/Assembly/Documents/ArtOfAssembly/Volume/Chapter_6/CH06-1.html
- http://en.wikipedia.org/wiki/Intel_Core_i7

Uno de los lenguajes máquina, o de instrucciones, más extendidos es lenguaje ensamblador (es lo más legible a la vez que cercano al lenguaje máquina propio de la UCP). El mismo es específico del procesador sobre el que lo ejecutemos

- http://es.wikipedia.org/wiki/Lenguaje_ensamblador#Ejemplos_de_lenguaje_ensamblador

UCP: ejemplo de uso

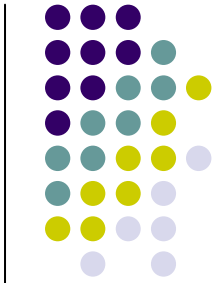


http://www.course.com/downloads/computerscience/aeonline/applets/cpu-sim/CPUSim.html

El orden “habitual” de ejecución (guiado por la UC) de un mandato es:

1. El contador de programa (PC) especifica qué orden debe ejecutarse (a través de un bus de direcciones)
2. La instrucción se carga de memoria (en este caso, de RAM) en el registro de instrucciones (IR) (a través de un bus de datos)
3. El IR “traduce” la instrucción
4. Ejecución de la instrucción:
   - Lectura de dato de memoria
   - Escritura de resultado a memoria
   - Operación aritmético – lógica (ALU)
5. Retorno al contador de programa (PC)

UCP: ejemplo de uso

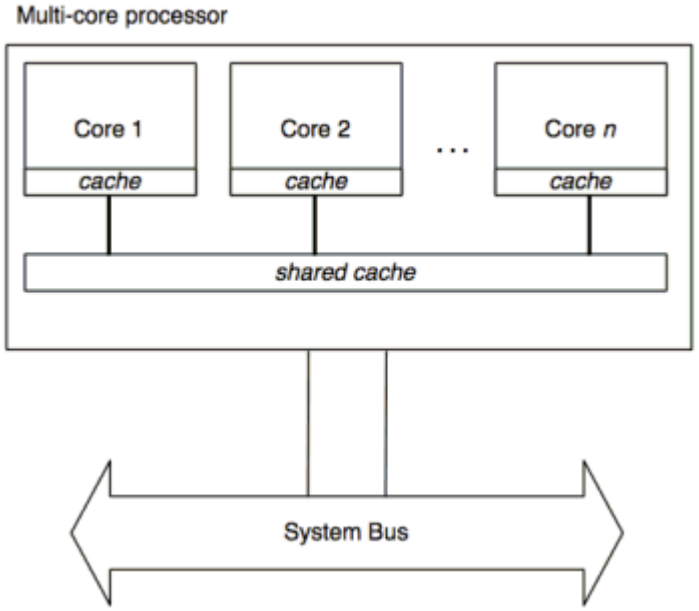


- Lectura de dato de memoria: si el dato está disponible en los registros W…Z, T1…T4 se captura de ellos; si no, se leerá (se envía una señal de lectura por el bus de control) desde su origen (a través del bus de direcciones) y se trasladará (por el bus de datos)
- Escritura de un dato a memoria: se envía (por el bus de direcciones) la dirección de la celda de memoria que se va a escribir, y por el bus de control la señal de escritura; el dato es enviado por el bus de datos

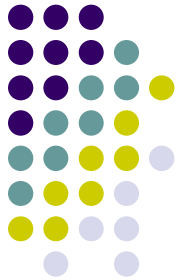
UCP: arquitecturas “multicore”

Las arquitecturas actuales suelen disponer de al menos dos núcleos y en muchos casos de 4 (o incluso 6, 8…)

Los procesadores con múltiples núcleos comparten algunas de sus partes (en particular, los buses, la memoria interna…)



UCP: rendimiento

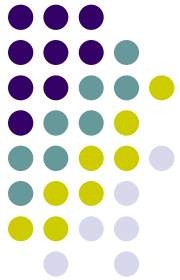


Por tanto, el rendimiento de la UCP no se puede medir como “nº de núcleos * nº de ciclos de cada uno”, puede haber “cuellos de botella” en acceso a memoria RAM, uso de buses…

Para ello se desarrollan tests específicos (benchmarks) basados en operaciones en coma flotante o sobre enteros que miden la cantidad de estas operaciones (FLOPS, floating point operations per second, MIPS, millions of instructions per second) que puede ejecutar el procesador (todos sus núcleos)

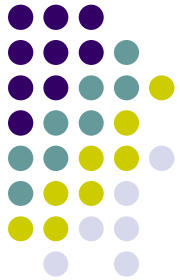
Uno de los benchmarks clásicos es Linpack. Un programa que ejecuta los tests de rendimiento, QwikMark

UCP: Arquitecturas convencionales



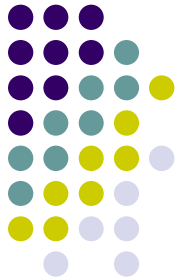
- Los registros de la CPU, los buses de datos y los de direcciones suelen tener un tamaño determinado, que se suele expresar en bits
- Una “arquitectura de 32 bits” implica que se pueden procesar datos (y direcciones) de 32 bits;
- Si cada registro almacena una posición de la memoria (que es la arquitectura más convencional), por medio de 32 bits podemos “indexar” 2^32 bits = 4 GB de memoria principal (o RAM)

UCP: Arquitecturas convencionales



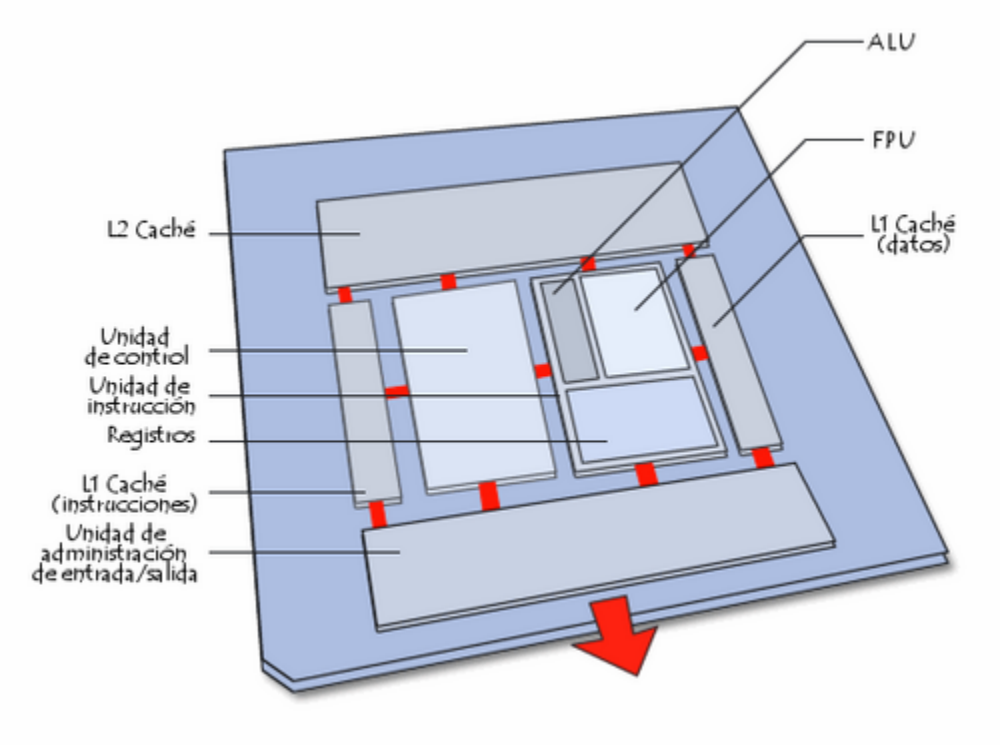
- Para superar esa limitación empezaron a surgir las arquitecturas de 64 bits, al principio (años 60) en supercomputadores, en la actualidad en ordenadores de gama media – alta
- Esto nos permite encontrar ordenadores con 16 ó 32GB de memoria principal, y a nivel teórico hasta 2^64 bits de memoria principal
- En las arquitecturas de procesadores x86, la mayor parte del software se compila a código de 32 bits, no de 64, por lo que no toma ventaja del mayor espacio de memoria o de datos
- Incluso algunos drivers de dispositivos antiguos, diseñados originariamente para 32 bits, podrían no ejecutarse en arquitecturas de 64 bits

Memoria interna



- Almacén de datos y programas en ejecución (o procesos); en particular, contiene las órdenes (en lenguaje “ensamblador”) que debe ir ejecutando el procesador
- Es volátil (desaparece al apagar el equipo)
- Se compone de los registros de la UCP, varias cachés (L1, L2, L3) y de la memoria interna (o principal)
- Todas ellas son de tipo RAM (random access memory, el tiempo de acceso a cualquier casilla es el mismo) aunque generalmente llamamos RAM a la memoria principal

Memoria interna



Memoria interna

Módulos de memoria RAM

Memoria interna

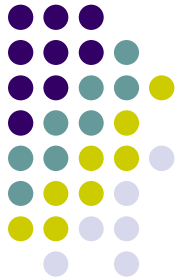
Se estructura en varios niveles. Cuanto más próxima a la UCP, resulta más cara, más pequeña y más rápida

Niveles:

1. Registros de memoria (MDR, MAR, IR, PC)
2. Memoria caché (“level” 1, 2, 3…)
3. Memoria principal (o RAM)

Memoria secundaria o externa (disco duro…)

Memoria interna

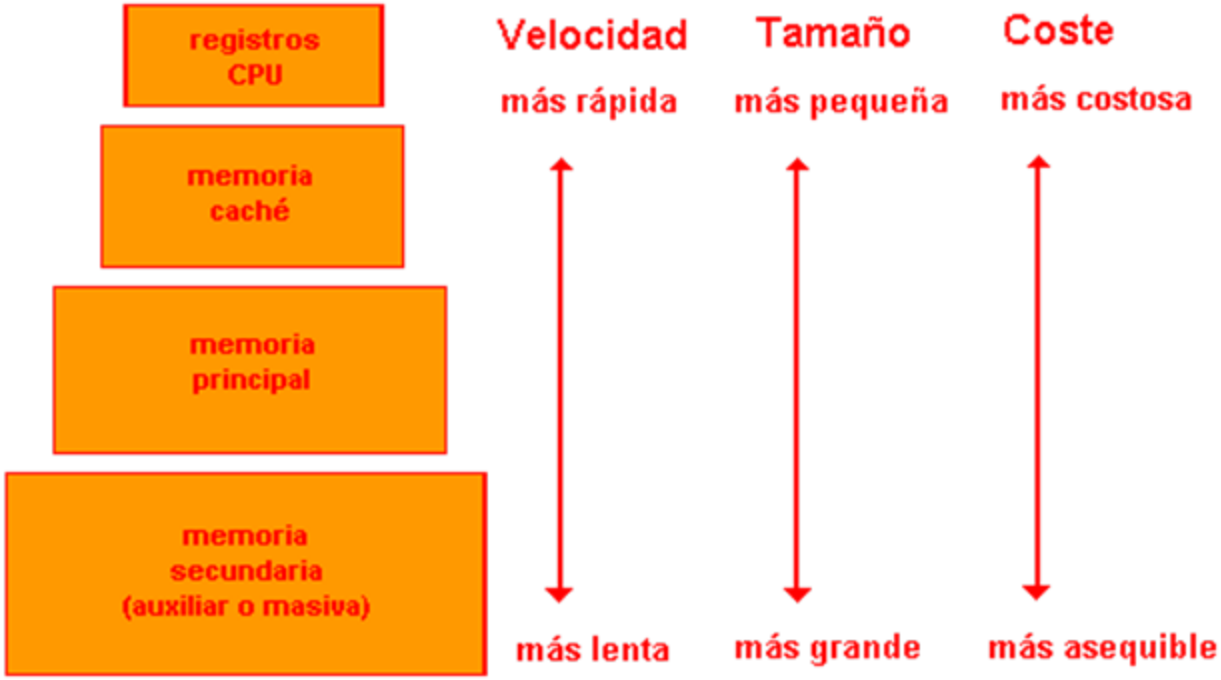


El disco duro no lo hemos incluido como memoria interna por dos motivos:

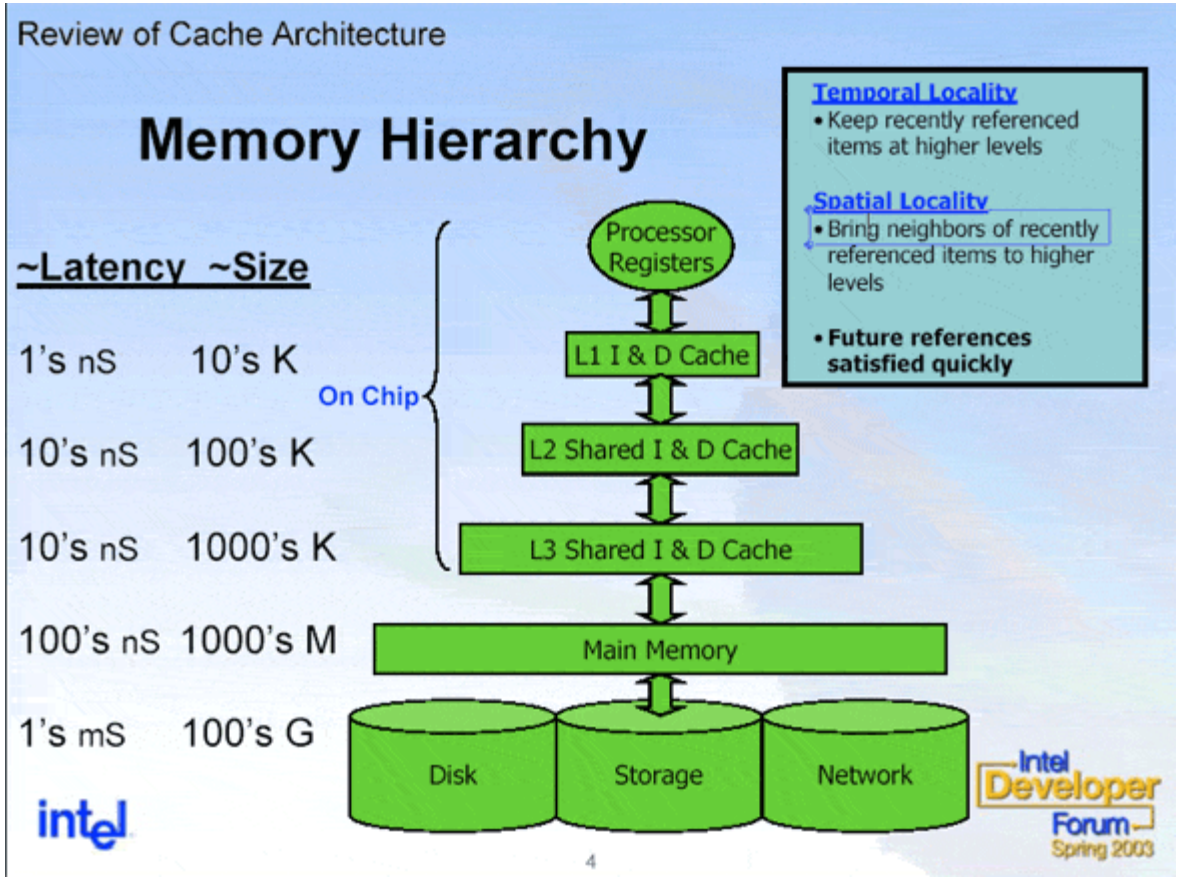
- La UC no puede acceder directamente al disco duro, primero copiará los datos o programas a los niveles 1, 2 ó 3 anteriores y luego los usará
- La memoria del disco duro no es “volátil”, ya que permanece en el mismo entre sesiones

Aún así, sí que cumple los principios de “ser de mayor tamaño” que los niveles 1, 2 y 3 y de “acceso más lento”

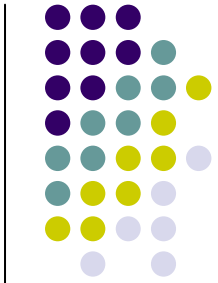
Memoria interna



Memoria interna



Memoria interna

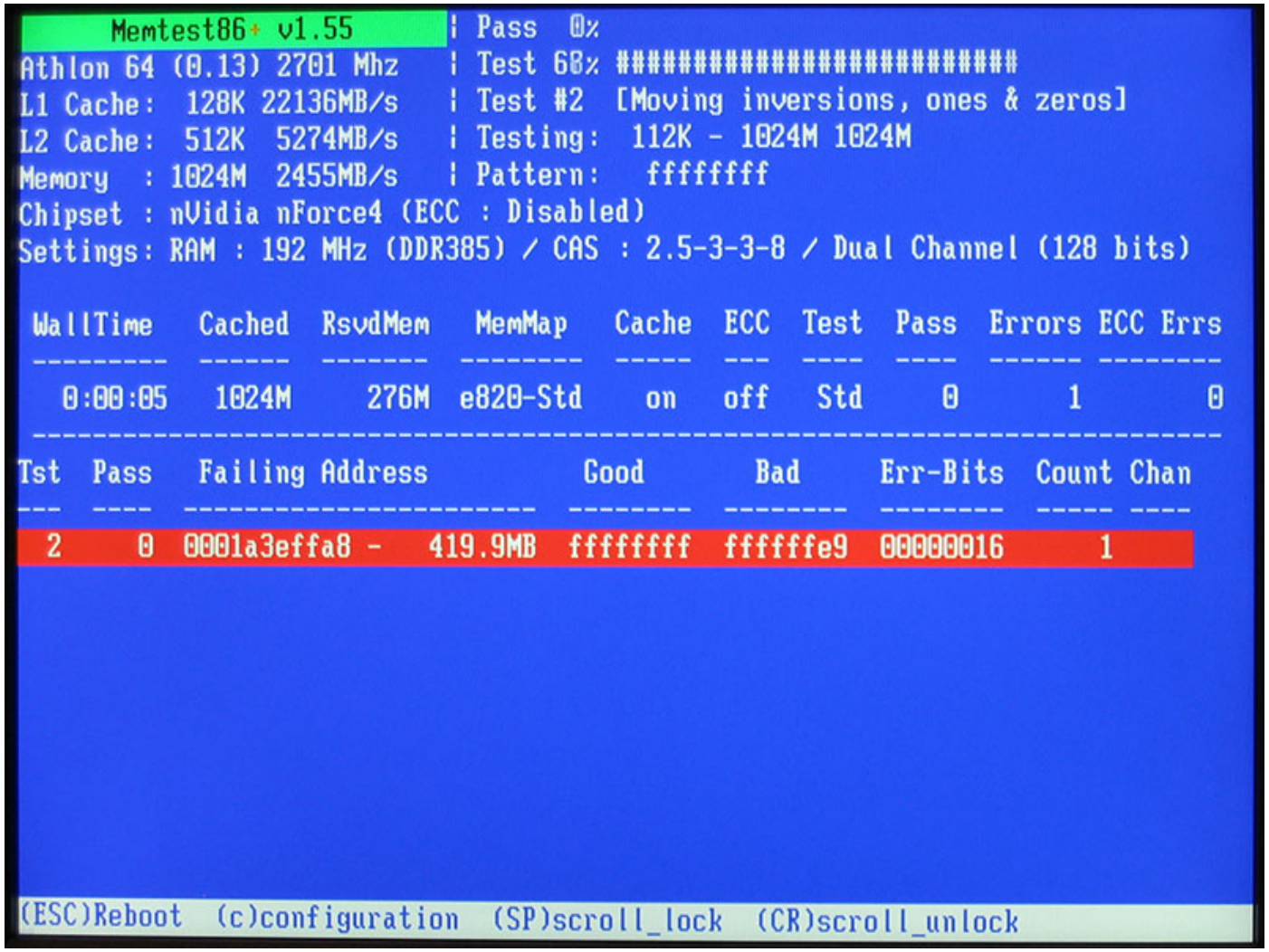


El uso de la memoria caché, o memoria interna se basa en los dos siguientes principios:

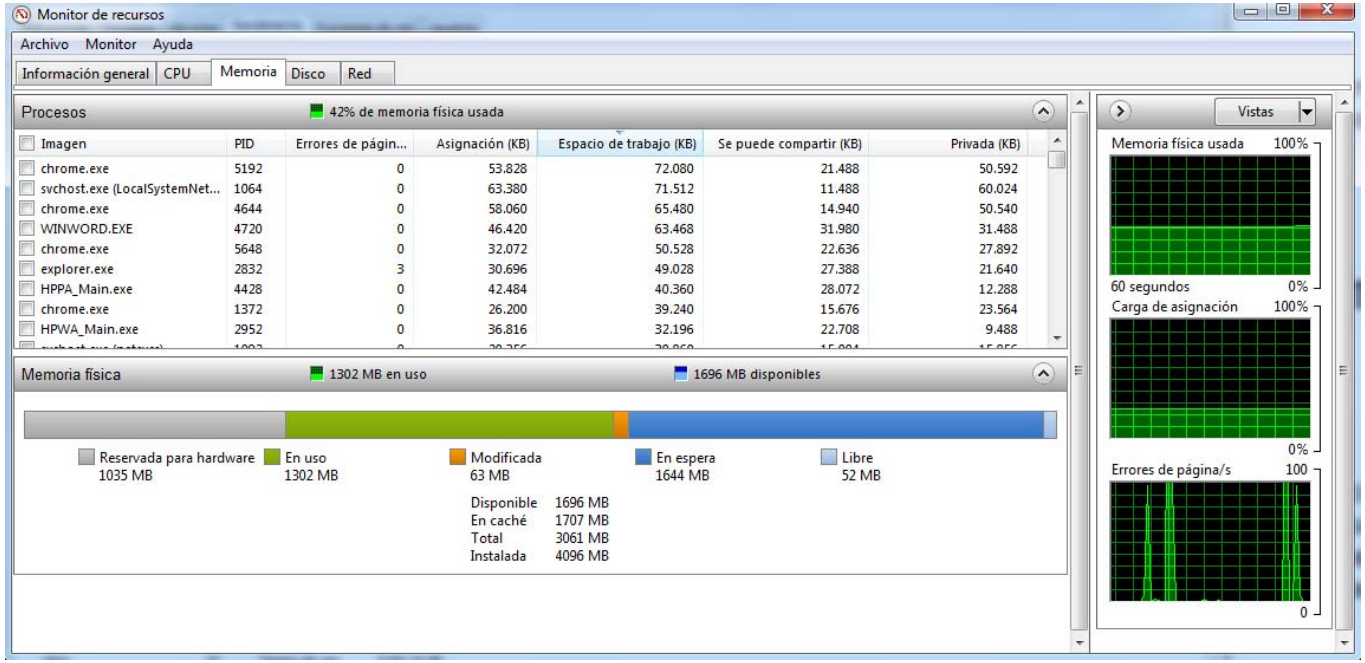
- Principio de localidad temporal: cuando un dato se acaba de usar, es posible que se use próximamente
- Principio de localidad espacial: cuando un dato se acaba de usar, es probable que se usen datos en direcciones de memoria próximas

Existen decenas de metodologías para optimizar el uso de los distintos niveles de caché, y así mejorar el rendimiento de los equipos

Memoria interna



Memoria interna

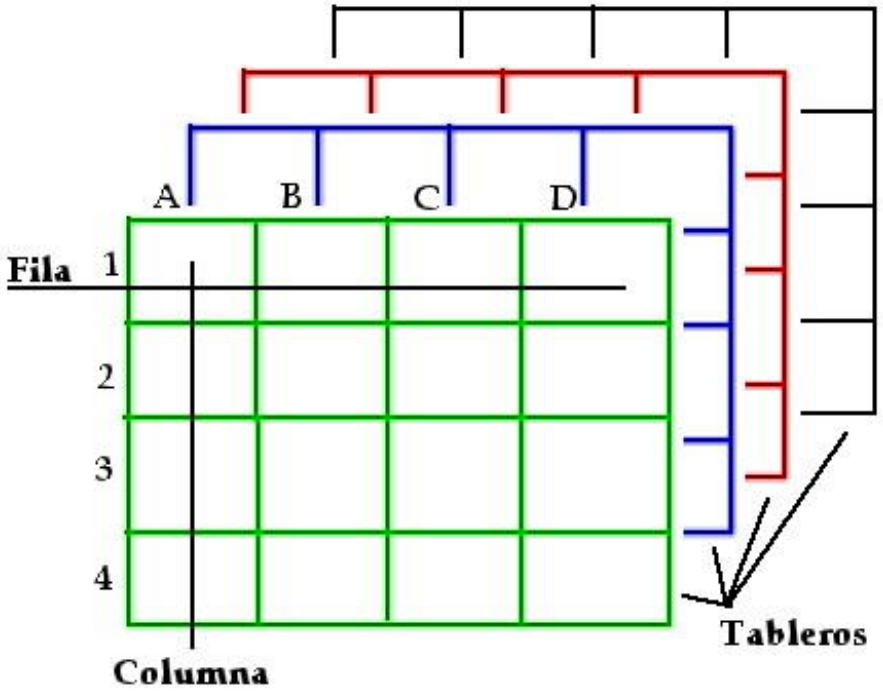


Otra forma de observar la memoria interna del equipo y su uso.

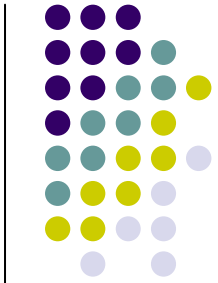
La memoria interna aloja los procesos (programas en ejecución) y los datos que se encuentran en uso en nuestro equipo en un momento concreto. La memoria interna se divide en páginas que son ocupadas por distintos procesos

Memoria interna

Aspecto (esquemático) de la memoria principal



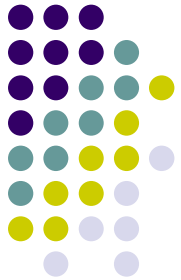
Memoria interna



Propiedades de la memoria interna:

- Suele ser de tipo RAM (o SDRAM, Synchronous Dynamic RAM)
- Su capacidad se mide en GB (tamaños habituales hoy en día son 2, 4 ó 8 GB)
- Su velocidad (número de ciclos de lectura o escritura por segundo) se mide en Hz (velocidades habituales, 1,6GHz por segundo, 1,6*10^9 ciclos por segundo)
- Dispone también de tiempos de latencia, que representan el tiempo (en ciclos) que consumen ciertas operaciones básicas (tiempo de acceso a una columna, a una fila y columna, de una fila a la siguiente y para activar un tablero completo). Generalmente se expresan como 6-6-6-18. A menores latencias, mayo rendimiento

Tecnologías de memoria interna

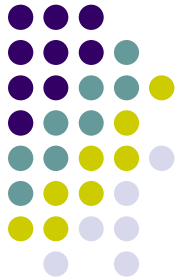


En la actualidad, la mayor parte de memorias principales son de tecnología SDRAM (Synchronous Dynamic RAM).

La noción de dinámica proviene de que los datos, cada cierto tiempo, deben ser revisados y recargados (consumiendo un ciclo de refresco)

La sincronía hace referencia a que las celdas de la RAM se “refrescan” coordinadas con el bus de sistema del ordenador

Tecnologías de memoria interna



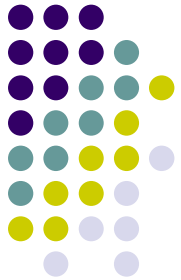
Las memorias SDRAM actuales suelen ser de tecnología DDR3 (Double Data Rate 3)

Su velocidad se calcula como sigue:

Un módulo DDR3 con frecuencia 100MHz transferirá (como máximo):

100.000.000 * 4 (multiplicador del reloj de bus)
* 2 (Data Rate)
* 64 (núm. de bits transf. en cada operación)
/ 8 (pasar de bits a bytes)
= 6.400.000 bytes = 6.400 MB / s

Tecnologías de memoria interna



Velocidades habituales de memoria SDRAM DDR3 serían:

- DDR3 – 800: 100MHz (Máx. transf. 6,4 GB / s)

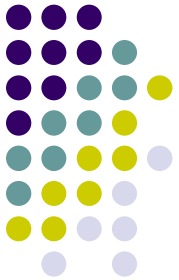
…

- DDR3 – 1333: 166MHz (Máx. transf. 10,66 GB / s)

…

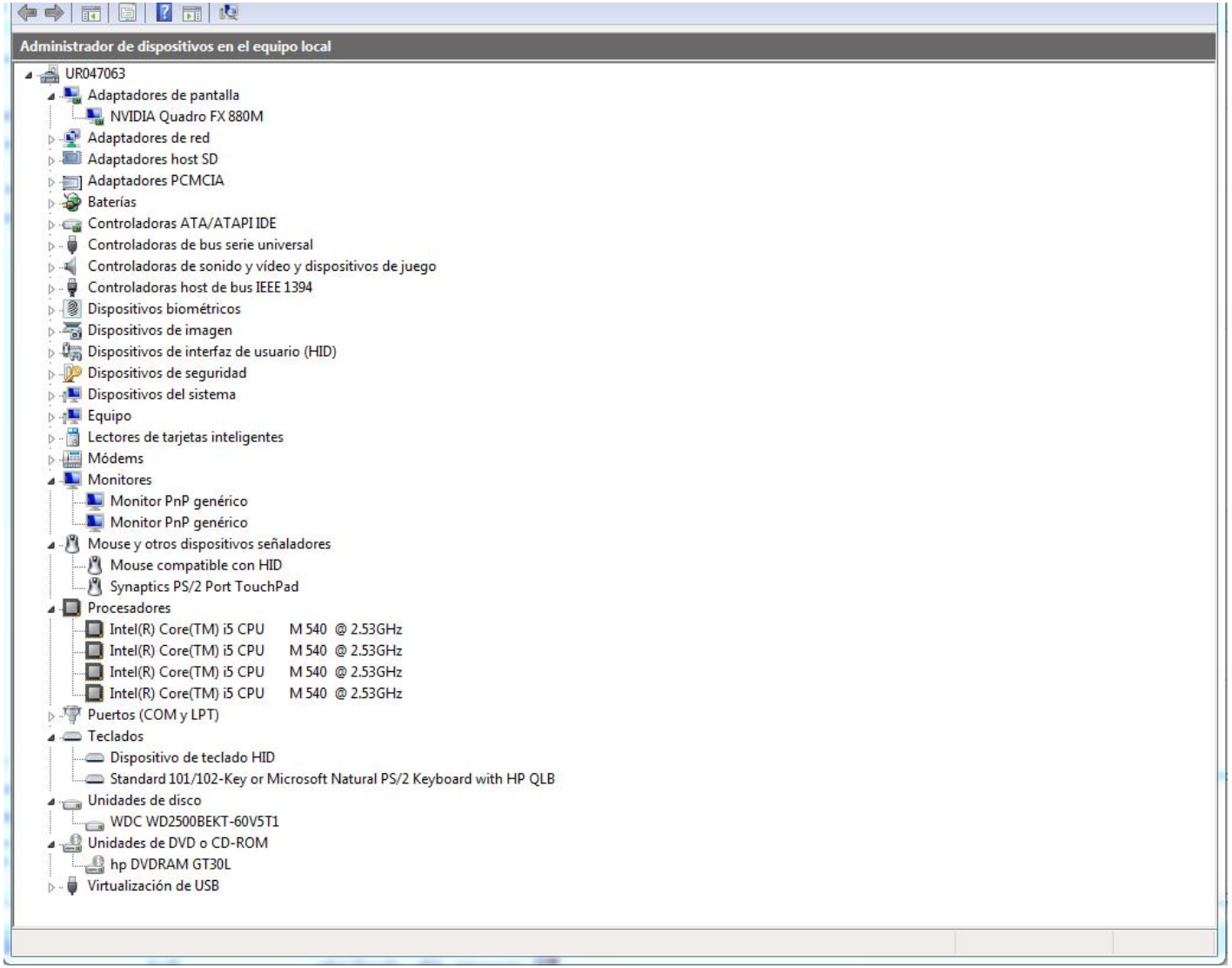
- DDR3 – 2133: 266MHz (Max. transf. 17,066 GB / s)

Unidades de E/S (Entrada/Salida)



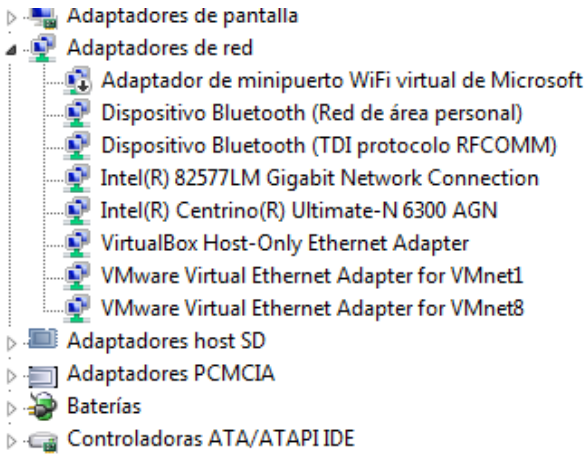
- Comunican periféricos (teclado, ratón, pantalla, impresoras, disco duro) con la UCP
- Posibilidades de comunicación:
  - E/S programada (UCP “visita” el módulo de E/S periódicamente)
  - E/S dirigida por interrupciones (bidireccional)
  - Acceso directo a memoria (independiente de la UCP)

Dispositivos (algunos de ellos de E/S)



Unidades de E/S

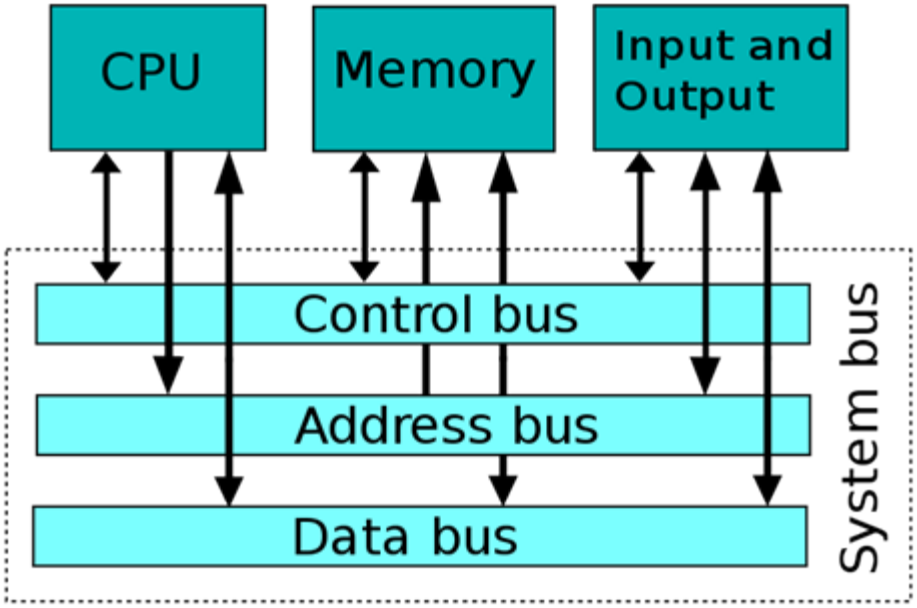
- Todos ellos requieren de un controlador (o driver) que traduce las órdenes enviadas y recibidas por la UCP al lenguaje que “entiende” el periférico o dispositivo de E/S
- La instalación de un periférico debería ir acompañada de la instalación del controlador correspondiente (o de una genérico) que permita la comunicación bidireccional
- Puedes comprobar algunos de ellos en el “administrador de dispositivos” de tu Sistema Operativo



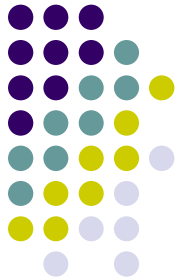
Elementos de interconexión (BUS)

Usados para transferir información entre dos puntos de la computadora

Hasta ahora hemos visto ya el bus interno de la CPU, que se suele dividir en tres partes, capaces de transmitir datos, direcciones y señales



Elementos de interconexión (BUS)



Por la estructura interna, se distinguen:

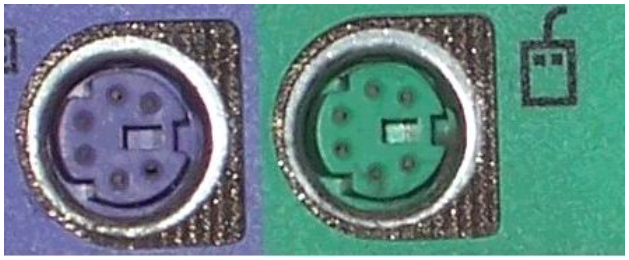
- Bus serie (datos transferidos bit a bit). Usado en puertos USB o conexiones Firewire y cada vez más en conexiones entre los elementos de la placa base
- Bus paralelo (conjuntos de bits). Dispone de varias líneas que realizan la comunicación de forma simultánea. Usado, por ejemplo, en el Front Side Bus (FSB) de procesadores Intel

Elementos de interconexión (puertos)

Son interfaces para el envío y recepción de datos entre dispositivos

Se dividen también, por su tipo de comunicación, en puertos serie y puertos paralelos

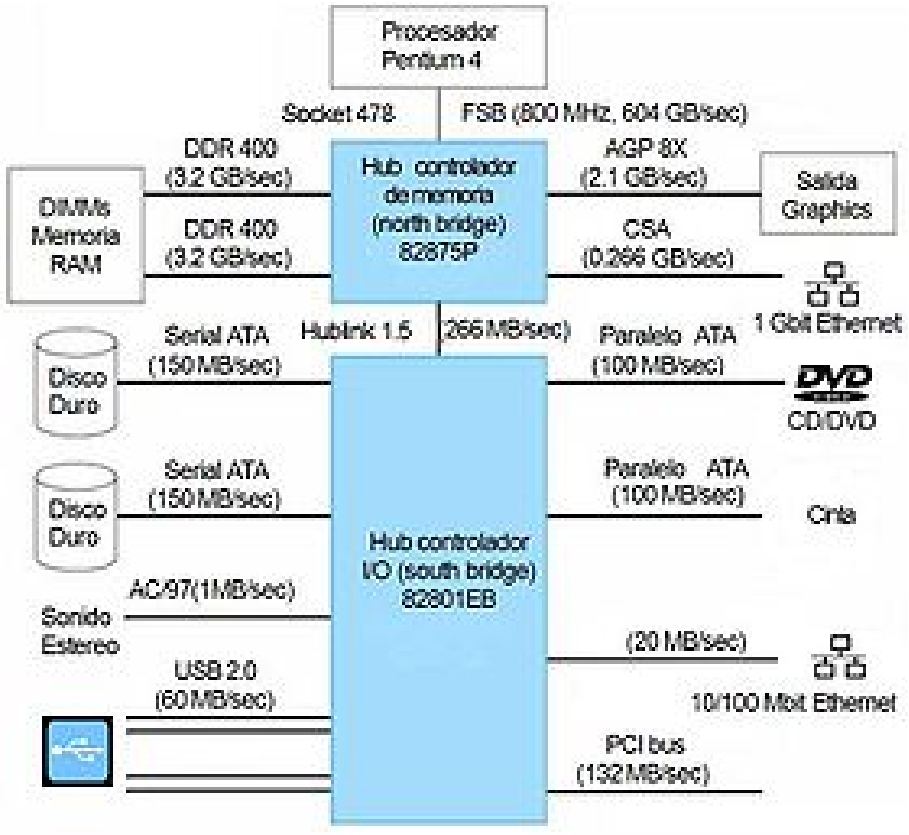
Puertos PS/2



Puerto VGA

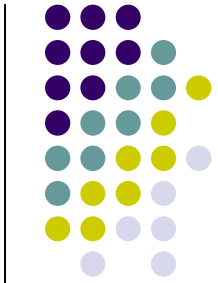
Puerto Ethernet

Elementos de interconexión



Ejemplo de dispositivo con distintos puertos y buses de comunicación con sus dispositivos externos y propios de la placa

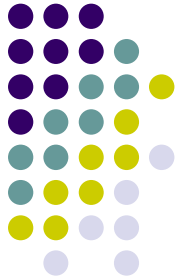
El disco duro (dispositivo de E/S)



Es importante diferenciar entre su **estructura física** (de naturaleza mecánica) y su **estructura lógica** (dividido en unidades de asignación indexadas…)

El **formateo** nos permite “olvidar” la estructura física y trabajar sobre la lógica

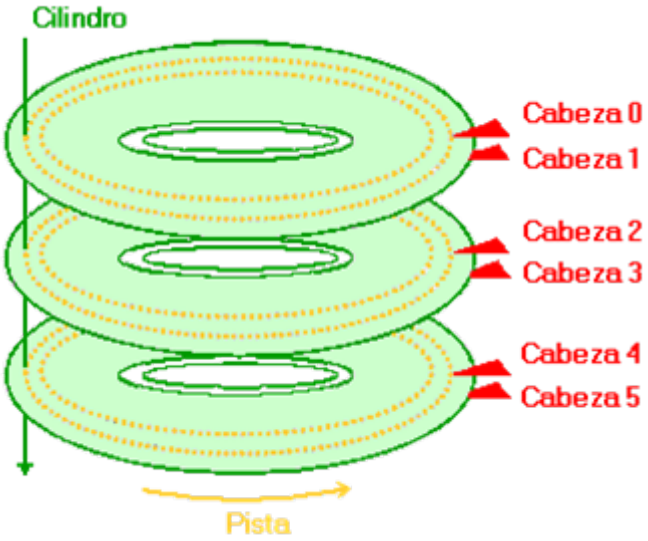
El disco duro (estructura física)

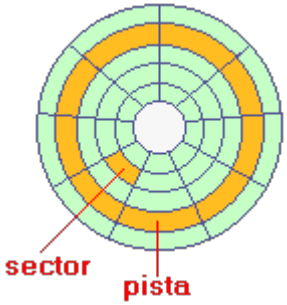


Estructura física (no de los SSD, solid state drive):

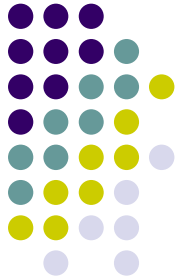
- Un conjunto de platos de superficie magnética sujetos a un eje que gira a miles de rpm
- Un conjunto de cabezas de lectura/escritura sujetas sobre un brazo móvil. Cada cabeza lee sobre una superficie distinta.
- Un bus de datos que conecta las cabezas con la electrónica de la unidad

El disco duro (estructura física)





El disco duro (estructura física)



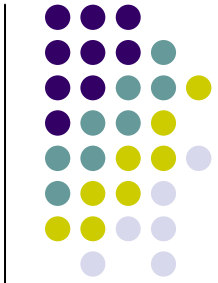
- Cada sector contiene 512 bytes (se está empezando a trabajar en sectores de 4KB)
- Cada disco contiene el mismo número de cilindros
- Tenemos tantas superficies como cabezas

Tamaño del disco duro

512 bytes * (nº de sectores por cilindro)

* (cilindros por disco) * (nº de cabezas)

El disco duro (estructura física)

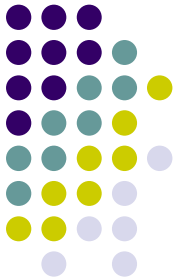


No cualquier número de cilindros, cabezas y sectores está permitido; por ejemplo, en las especificaciones originales:

- El número de cilindros puede ocupar 10 bits (2^10 = 1024 cilindros máximos)
- El número de cabezas ocupa 8 bits (2^8 = 256 cabezas máximas)
- El número de sectores ocupa 6 bits (2^6 = 64 sectores, 63 disponibles)

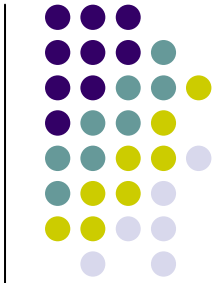
1024 * 256 * 63 * 512 bytes aprox. 8 GB, tamaño máximo

El disco duro (estructura física)



Cada posición (sector) del disco duro, se indexa por medio de una tupla (C, H, S), (cabeza, cilindro, sector)

El disco duro (estructura física)



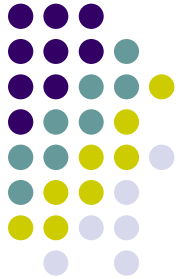
Direccionamiento LBA (logical block addressing)

Pensado para solventar las anteriores limitaciones; los sectores se numeran de forma consecutiva (0, 1… N - 1)

El direccionamiento LBA puede ser de 28 ó 48 bits; con 28 bits, el límite de disco duro sería 2^28^* 512 bytes, con 48 bits, 2^48 * 512 bytes, aprox 128 PiB

Las direcciones CHS se pueden traducir a direcciones LBA para mantener la compatibilidad de dispositivos antiguos

El disco duro (estructura lógica)



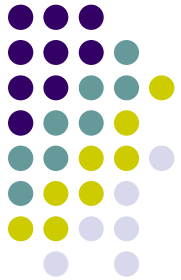
Por medio del formateo le damos estructura lógica al disco.

Dos funciones:

1. Divide el disco duro en “unidades de asignación” (o clusteres), de tamaño 512 bytes, 1KB, 2KB, 4KB…64KB.

El cluster es un conjunto contiguo de sectores que componen la unidad más pequeña de almacenamiento (o espacio mínimo que ocupará cualquier archivo en disco)

Disco duro (estructura lógica)

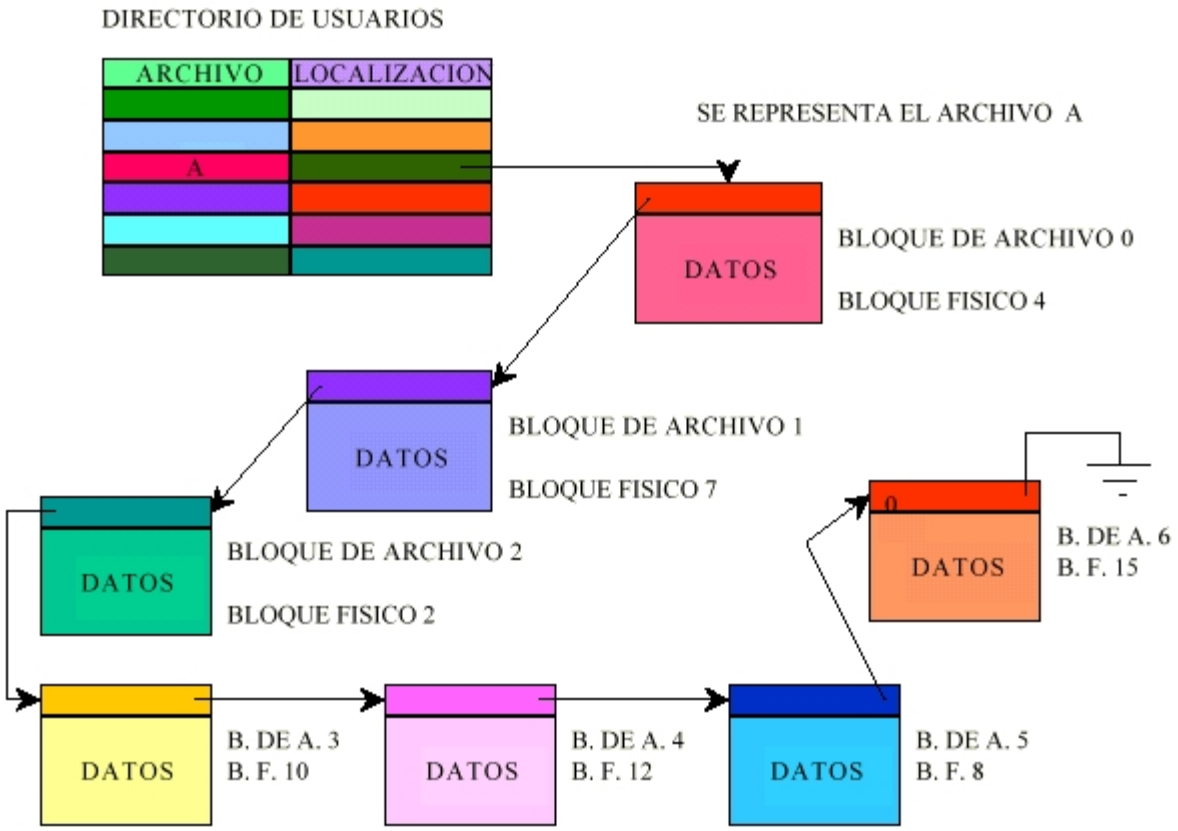


Por medio del formateo le damos estructura lógica al disco.

Dos funciones:

2. Crea una tabla de asignación de archivos (FAT, MFT…) donde se almacenan las coordenadas de cada “cluster”, el nombre del fichero allí contenido y alguna otra propiedad (depende del formato: FAT32, NTFS…)

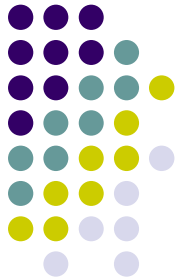
Disco duro (funcionamiento)



Figura˜4.4: Encadenamiento de bloques o lista ligada de bloques.

Esta estructura corresponde a discos duros con sistema de archivos FAT; sistemas de archivos más modernos contienen otro tipo de indexación de archivos

Sistema de arranque (MBR)



La BIOS (ROM BIOS, Basic Input/Output System) realiza comprobaciones básicas y carga el MBR (primer sector, 512 bytes) de la partición activa del disco de arranque

Primer sector físico del disco: 512 Bytes

446 Bytes: Código máquina (gestor de arranque)

64 Bytes: Tabla de particiones

2 Bytes: Firma de unidad arrancable ("55h AAh" en hexadecimal)

Limitaciones del MBR

- Sólo puede contener 4 particiones primarias
- Sólo una de las particiones puede ser activa