

– Typeset by GMNI & Foil<sub>E</sub>TEX –

**LENGUAJE C:  
Conceptos Generales  
sobre Almacenamiento de Información**  
F. Navarrina, I. Colominas, H. Gómez, J. París, M. Casteleiro



**GMNI — GRUPO DE MÉTODOS NUMÉRICOS EN INGENIERÍA**

**Departamento de Métodos Matemáticos y de Representación  
Escuela Técnica Superior de Ingenieros de Caminos, Canales y Puertos  
Universidad de A Coruña, España**

e-mail: [fnavarrina@udc.es](mailto:fnavarrina@udc.es)

página web: <http://caminos.udc.es/gmni>





# ÍNDICE

- ▶ Bit
- ▶ Byte
- ▶ Código ASCII
- ▶ Códigos ASCII extendidos
- ▶ Unidades de medida de información
- ▶ Archivo
- ▶ Palabra
- ▶ Tipos de Variables





# Bit (I)

## BIT

- ▶ Componente de memoria con 2 estados posibles.
- ▶ Elemento básico de la codificación binaria (de números, caracteres, etc.)
- ▶ Variable capaz de almacenar 2 números enteros

$$bit \in \{0, 1\}.$$

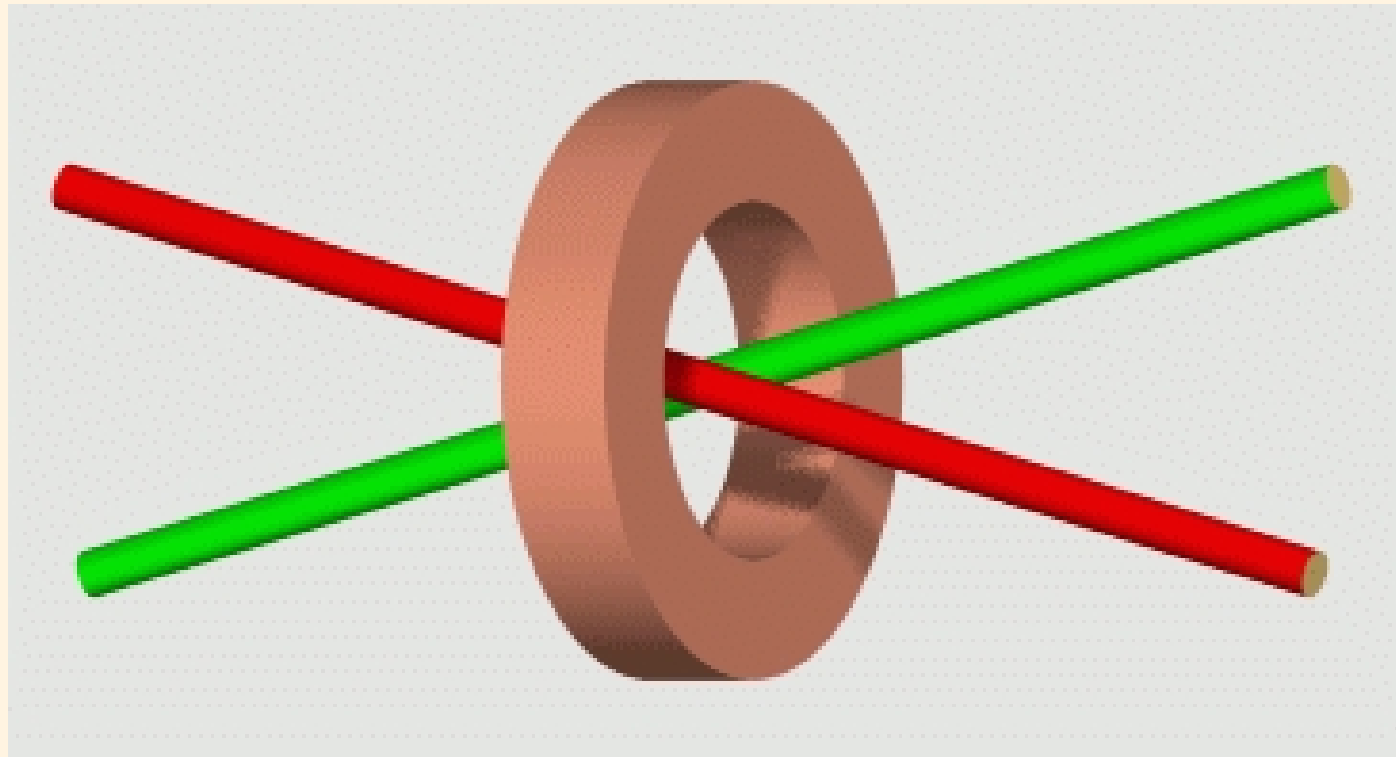
- ▶ Unidad de medida de información.





## Bit (II)

- ▶ Origen: **NÚCLEOS DE FERRITA**



Esquema de un componente elemental de memoria de ferrita:  
Núcleo de ferrita, hilo de escritura e hilo de lectura.

(Fuente: <[http://www.lsi.us.es/~rovayo/ferrita/ferrita\\_pf.html](http://www.lsi.us.es/~rovayo/ferrita/ferrita_pf.html)>, M. Rovayo, Univ. de Sevilla)



# Byte (I)

## BYTE

- ▶ Conjunto de 8 bits con  $2^8 = 256$  estados posibles.
- ▶ Memoria necesaria para codificar el conjunto habitual de caracteres (letras mayúsculas y minúsculas, signos de puntuación, símbolos matemáticos más habituales, caracteres de control de impresión (CR, LF), caracteres latinos, etc.
- ▶ Variable capaz de almacenar  $2^8 = 256$  números enteros

$$\text{byte} \in \{0, \dots, 255\}.$$

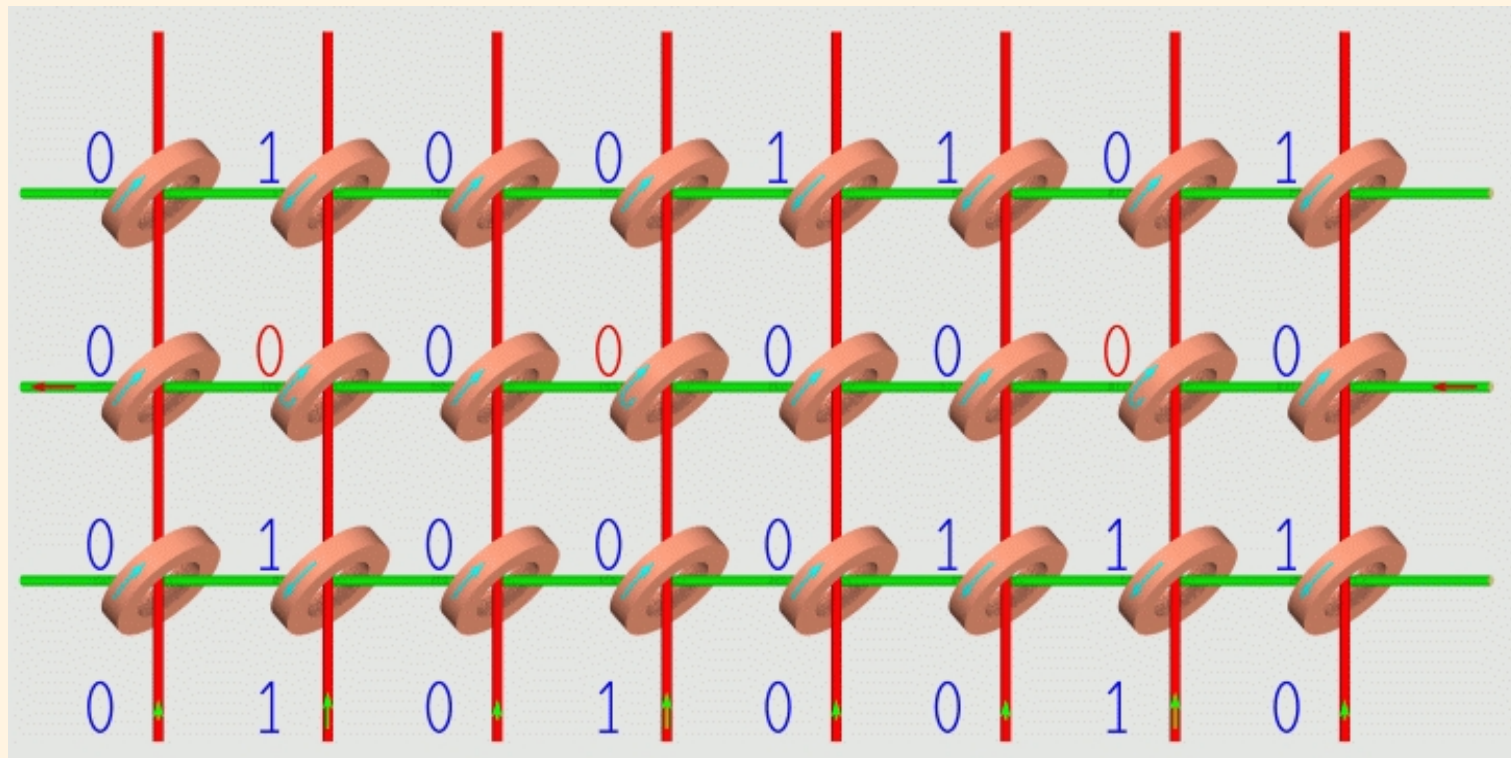
- ▶ Unidad de medida de información.





## Byte (II)

► Origen: **MEMORIAS DE FERRITA**



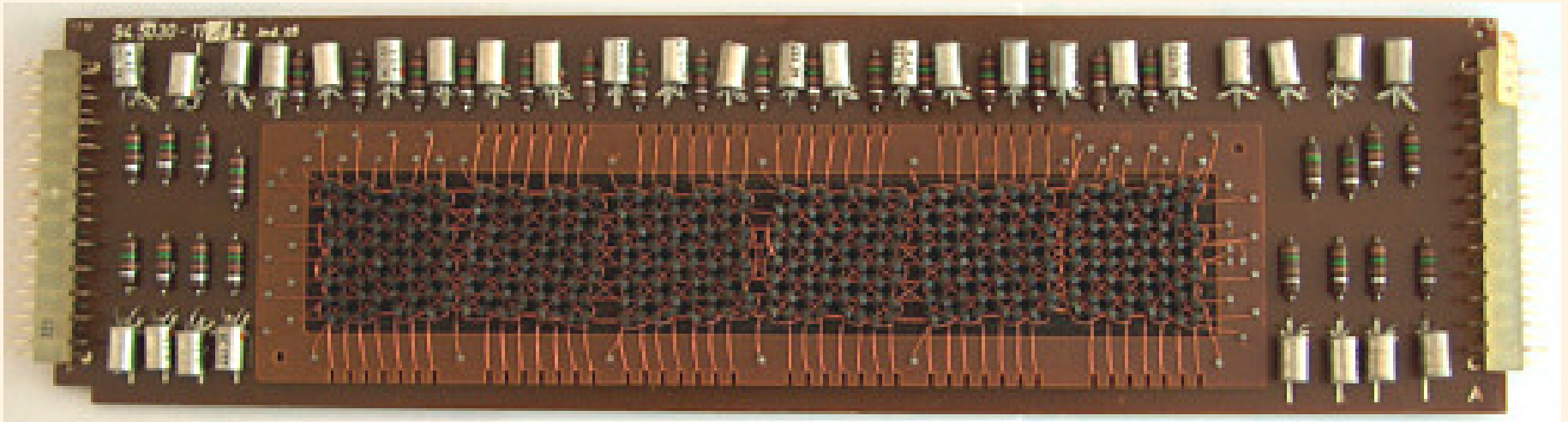
Esquema de un componente elemental de memoria de ferrita:  
Núcleo de ferrita, hilo de escritura e hilo de lectura.

(Fuente: <[http://www.lsi.us.es/~rovayo/ferrita/ferrita\\_pf.html](http://www.lsi.us.es/~rovayo/ferrita/ferrita_pf.html)>, M. Rovayo, Univ. de Sevilla)



## Byte (Illa)

► Ejemplo:



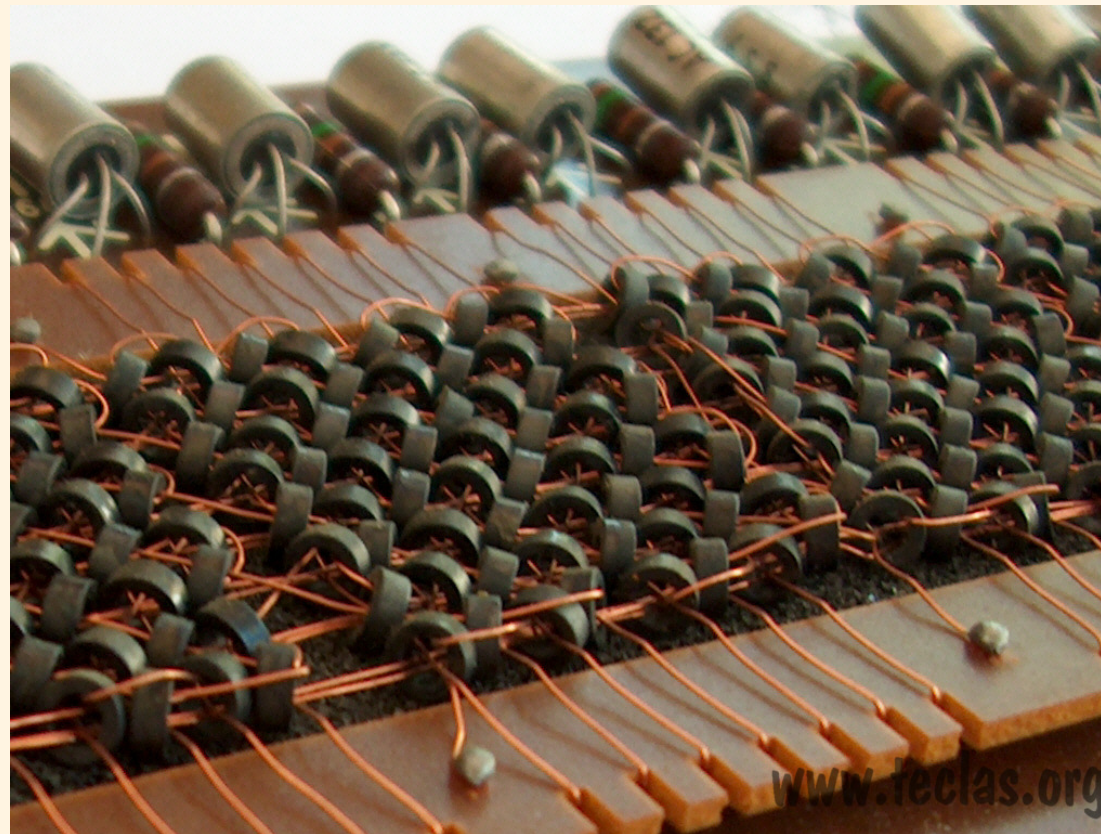
Memoria de una calculadora Olympia RAE 4/30-3 (1966 aprox.).  
(Fuente: <<http://www.teclas.org>>)





## Byte (IIIb)

► Ejemplo:



Memoria de una calculadora Olympia RAE 4/30-3 (1966 aprox.). Detalle.  
(Fuente: <<http://www.teclas.org>>)



## Byte (IIc)

► Ejemplo:



Calculadora Olympia RAE 4/30-3 (1966 aprox.).  
(Fuente: <<http://www.teclas.org>>)



# Código ASCII (I)

## CÓDIGO ASCII (\*)

- ▶ Creado en 1963 por **ASA** (\*\*), actualmente **ANSI** (\*\*\*).
  - Desarrollado a partir de los códigos telegráficos usados en los Estados Unidos.
  - 7 bits para codificar  $2^7 = 128$  caracteres (caracteres imprimibles del alfabeto inglés y códigos de control).
  - 1 bit adicional se utilizaba inicialmente para **control de paridad (detección de errores)**.
- ▶ Revisado en 1967 (inclusión de minúsculas) y en 1986.
- ▶ Base de códigos posteriores, más completos (letras acentuadas, otros alfabetos).

---

(\*) **ASCII**  $\implies$  American Standard Code for Information Interchange.

(\*\*) **ASA**  $\implies$  American Standards Association.

(\*\*\*) **ANSI**  $\implies$  American National Standards Institute.





# Código ASCII (Ila)

## ► **CÓDIGO ASCII:** (caracteres 0—31)

BIN.	HEX.	DEC.	REPRESENTACIÓN
0000 0000	00	0	<b>NUL</b> (null character)
0000 0001	01	1	<b>SOH</b> (start of header)
0000 0010	02	2	<b>STX</b> (start of text)
0000 0011	03	3	<b>ETX</b> (end of text)
0000 0100	04	4	<b>EOT</b> (end of transmission)
0000 0101	05	5	<b>ENQ</b> (enquiry)
0000 0110	06	6	<b>ACK</b> (acknowledgement)
0000 0111	07	7	<b>BEL</b> (bell)
0000 1000	08	8	<b>BS</b> (back space)
0000 1001	09	9	<b>HT</b> (horizontal tabulation)
0000 1010	0A	10	<b>LF</b> (line feed)
0000 1011	0B	11	<b>VT</b> (vertical tabulation)
0000 1100	0C	12	<b>FF</b> (form feed)
0000 1101	0D	13	<b>CR</b> (carriage return)
0000 1110	0E	14	<b>SO</b> (shift out)
0000 1111	0F	15	<b>SI</b> (shift in)

BIN.	HEX.	DEC.	REPRESENTACIÓN
0001 0000	10	16	<b>DLE</b> (data link escape)
0001 0001	11	17	<b>DC1</b> (device control 1)
0001 0010	12	18	<b>DC2</b> (device control 2)
0001 0011	13	19	<b>DC3</b> (device control 3)
0001 0100	14	20	<b>DC4</b> (device control 4)
0001 0101	15	21	<b>NAK</b> (negative acknowl.)
0001 0110	16	22	<b>SYN</b> (synchronous idle)
0001 0111	17	23	<b>ETB</b> (end of transm. block)
0001 1000	18	24	<b>CAN</b> (cancel)
0001 1001	19	25	<b>EM</b> (end of medium)
0001 1010	1A	26	<b>SUB</b> (substitute)
0001 1011	1B	27	<b>ESC</b> (escape)
0001 1100	1C	28	<b>FS</b> (file separator)
0001 1101	1D	29	<b>GS</b> (group separator)
0001 1110	1E	30	<b>RS</b> (record separator)
0001 1111	1F	31	<b>US</b> (unit separator)





# Código ASCII (Iib)

## ► CÓDIGO ASCII: (caracteres 32—63)

BIN.	HEX.	DEC.	REPRESENTACIÓN
0010 0000	20	32	SP (espacio)
0010 0001	21	33	!
0010 0010	22	34	”
0010 0011	23	35	#
0010 0100	24	36	\$
0010 0101	25	37	%
0010 0110	26	38	&
0010 0111	27	39	,
0010 1000	28	40	(
0010 1001	29	41	)
0010 1010	2A	42	*
0010 1011	2B	43	+
0010 1100	2C	44	,
0010 1101	2D	45	-
0010 1110	2E	46	.
0010 1111	2F	47	/

BIN.	HEX.	DEC.	REPRESENTACIÓN
0011 0000	30	48	0
0011 0001	31	49	1
0011 0010	32	50	2
0011 0011	33	51	3
0011 0100	34	52	4
0011 0101	35	53	5
0011 0110	36	54	6
0011 0111	37	55	7
0011 1000	38	56	8
0011 1001	39	57	9
0011 1010	3A	58	:
0011 1011	3B	59	;
0011 1100	3C	60	<
0011 1101	3D	61	=
0011 1110	3E	62	>
0011 1111	3F	63	?





# Código ASCII (IIC)

## ► **CÓDIGO ASCII:** (caracteres 64—95)

BIN.	HEX.	DEC.	REPRESENTACIÓN
0100 0000	40	64	@
0100 0001	41	65	A
0100 0010	42	66	B
0100 0011	43	67	C
0100 0100	44	68	D
0100 0101	45	69	E
0100 0110	46	70	F
0100 0111	47	71	G
0100 1000	48	72	H
0100 1001	49	73	I
0100 1010	4A	74	J
0100 1011	4B	75	K
0100 1100	4C	76	L
0100 1101	4D	77	M
0100 1110	4E	78	N
0100 1111	4F	79	O

BIN.	HEX.	DEC.	REPRESENTACIÓN
0101 0000	50	80	P
0101 0001	51	81	Q
0101 0010	52	82	R
0101 0011	53	83	S
0101 0100	54	84	T
0101 0101	55	85	U
0101 0110	56	86	V
0101 0111	57	87	W
0101 1000	58	88	X
0101 1001	59	89	Y
0101 1010	5A	90	Z
0101 1011	5B	91	[
0101 1100	5C	92	\
0101 1101	5D	93	]
0101 1110	5E	94	^
0101 1111	5F	95	_





# Código ASCII (IId)

## ► **CÓDIGO ASCII:** (caracteres 96—127)

BIN.	HEX.	DEC.	REPRESENTACIÓN
0110 0000	60	96	'
0110 0001	61	97	a
0110 0010	62	98	b
0110 0011	63	99	c
0110 0100	64	100	d
0110 0101	65	101	e
0110 0110	66	102	f
0110 0111	67	103	g
0110 1000	68	104	h
0110 1001	69	105	i
0110 1010	6A	106	j
0110 1011	6B	107	k
0110 1100	6C	108	l
0110 1101	6D	109	m
0110 1110	6E	110	n
0110 1111	6F	111	o

BIN.	HEX.	DEC.	REPRESENTACIÓN
0111 0000	70	112	p
0111 0001	71	113	q
0111 0010	72	114	r
0111 0011	73	115	s
0111 0100	74	116	t
0111 0101	75	117	u
0111 0110	76	118	v
0111 0111	77	119	w
0111 1000	78	120	x
0111 1001	79	121	y
0111 1010	7A	122	z
0111 1011	7B	123	{
0111 1100	7C	124	
0111 1101	7D	125	}
0111 1110	7E	126	~
0111 1111	7F	127	DEL (delete)





## Código ASCII (Ile)

- ▶ **CÓDIGO ASCII:** (caracteres 128—255)
  - Su primer bit es igual a 1.
  - En realidad, las correspondientes secuencias de bits **NO forman parte del Código ASCII.**
  - Dan lugar a extensiones compatibles del Código ASCII.







# Códigos ASCII extendidos (I)

## CÓDIGOS ASCII EXTENDIDOS (\*)

### ▶ **CÓDIGO ISO-8859-1 (o ISO Latin 1)** (\*):

- 8 bits para codificar  $2^8 = 256$  caracteres (caracteres imprimibles del alfabeto latino y códigos de control).
- Los primeros 128 códigos (primer bit = 0) coinciden con los **ASCII**.
- Los restantes 128 códigos (primer bit = 1) permiten codificar caracteres adicionales:
  - ▷ letras de otros alfabetos (ñ, Ñ, ç, Ç, ...)
  - ▷ letras acentuadas (á, à, â, ..., ü, ...)
  - ▷ otros caracteres (©, ¡, ¿, ...)

### ▶ **CÓDIGOS UNICODE:**

- Utilizan más de 1 byte para codificar (casi) todos los alfabetos.
- Los primeros 256 códigos coinciden con los **ISO-8859-1**.

---

(\*) **ISO**  $\implies$  **International Organization for Standardization**.





## UNIDADES DE MEDIDA DE INFORMACIÓN

### ► Unidades Básicas:

- $1 \text{ b} \implies 1 \text{ bit}$
- $1 \text{ B} \implies 1 \text{ byte}$

### ► Otras Unidades:

- **PREFIJO** + Unidad Básica



# Unidades de medida de información (IIa)

## ► Prefijos del Sistema Internacional (\*):

SÍMBOLO	NOMBRE	SIGNIFICADO	EQUIVALENCIA	
k	kilo	$10^3$	1.000	= $10^3$
M	mega	$10^6$	1.000.000	= $10^6$
G	giga	$10^9$	1.000.000.000	= $10^9$
T	tera	$10^{12}$	1.000.000.000.000	= $10^{12}$
P	peta	$10^{15}$	1.000.000.000.000.000	= $10^{15}$
E	exa	$10^{18}$	1.000.000.000.000.000.000	= $10^{18}$
Z	zetta	$10^{21}$	1.000.000.000.000.000.000.000	= $10^{21}$
Y	yotta	$10^{24}$	1.000.000.000.000.000.000.000.000	= $10^{24}$

(\*) Uso habitual en telecomunicaciones (velocidad de transmisión: [bps/Bps](#)=bits/bytes por segundo).





# Unidades de medida de información (IIb)

## ► Prefijos Tradicionales (\*):

SÍMBOLO	NOMBRE	SIGNIFICADO	EQUIVALENCIA	
k,K	kilo	$2^{10}$	1.024	$\approx 10^3$
M	mega	$2^{20}$	1.048.576	$\approx 10^6$
G	giga	$2^{30}$	1.073.741.824	$\approx 10^9$
T	tera	$2^{40}$	1.099.511.627.776	$\approx 10^{12}$
P	peta	$2^{50}$	1.125.899.906.842.624	$\approx 10^{15}$
E	exa	$2^{60}$	1.152.921.504.606.846.976	$\approx 10^{18}$
Z	zetta	$2^{70}$	1.180.591.620.717.411.303.424	$\approx 10^{21}$
Y	yotta	$2^{80}$	1.208.925.819.614.629.174.706.176	$\approx 10^{24}$

### ● Problemas:

- ▷ Los errores respecto a los prefijos del Sistema Internacional son cada vez mayores.
- ▷ Ambigüedad: ¿cuánto cabe en un disco de 320 GB?

(\*) Uso habitual en informática (capacidad de almacenamiento en memoria, direccionamiento).





# Unidades de medida de información (IIC)

## ► Nuevos prefijos binarios, IEEE Standard 1541-2002 (\*):

SÍMBOLO	NOMBRE	SIGNIFICADO	EQUIVALENCIA	
Ki	kibi	$2^{10}$	1.024	$\approx 10^3$
Mi	mebi	$2^{20}$	1.048.576	$\approx 10^6$
Gi	gibi	$2^{30}$	1.073.741.824	$\approx 10^9$
Ti	tebi	$2^{40}$	1.099.511.627.776	$\approx 10^{12}$
Pi	pebi	$2^{50}$	1.125.899.906.842.624	$\approx 10^{15}$
Ei	exbi	$2^{60}$	1.152.921.504.606.846.976	$\approx 10^{18}$
Zi	zebi	$2^{70}$	1.180.591.620.717.411.303.424	$\approx 10^{21}$
Yi	yobi	$2^{80}$	1.208.925.819.614.629.174.706.176	$\approx 10^{24}$

(\*) IEEE  $\implies$  Institute of Electrical and Electronics Engineers.





## ARCHIVO

▶ Ristra de **BYTES**  
almacenada en una unidad de memoria de cualquier tipo.

▶ Tipos:

- **ARCHIVOS DE TEXTO / ASCII**

programas FORTRAN: (\***f**, \***for**),

programas C: (\***c**),

archivos TXT (O CON FORMATO): (\***txt**),

programas de comandos: (\***bat**),

...

- **ARCHIVOS BINARIOS / DE TIPO IMAGEN**

programas OBJETO: (\***o**), \***obj**

programas EJECUTABLES: (\***exe**, sin extensión),

archivos SIN FORMATO: (\***xls**, \***dwg**, \***jpg**)...

...



### ▶ ARCHIVOS DE TEXTO /ASCII

- Los bytes que contienen representan (exclusivamente) códigos ASCII de
  - ▷ caracteres imprimibles (letras, números, símbolos) y de
  - ▷ códigos de control de “impresión” (CR, LF, FF, ...).
- Los “controles de carro” (CR LF) crean una organización **por líneas**.
- Ejemplos: (véase la carpeta [EjemplosDeArchivosDeTextoYBinarios](#))
  - ▷ El archivo [hello.c](#) es un archivo de texto que contiene un programa muy sencillo escrito en C.
  - ▷ El archivo [hello.c.txt](#) muestra los bytes que forman el archivo [hello.c](#),
  - ▷ y el archivo [hello.c.b.txt](#) muestra los correspondientes bits.



### ▶ ARCHIVOS BINARIOS / DE TIPO IMAGEN

- Los bytes que contienen pueden tomar cualquier valor (entre 0 y 255):
  - ▷ Algunos pueden representar códigos ASCII de caracteres no de códigos de impresión (texto de títulos, menús de opciones, preguntas o respuestas, mensajes de error o advertencia, etc.)
  - ▷ Con carácter general tendrán otro significado (instrucciones, datos, ...) que depende del sistema operativo o de la aplicación que use el archivo.
- No están organizados por líneas.
- Ejemplos: (véase la carpeta [EjemplosDeArchivosDeTextoYBinarios](#))
  - ▷ El archivo [hello.o](#) es un archivo binario que contiene el programa objeto que se obtiene al compilar [hello.c](#).
  - ▷ El archivo [hello.o.txt](#) muestra los bytes que forman el archivo [hello.o](#),
  - ▷ y el archivo [hello.o.b.txt](#) muestra los correspondientes bits.
  - ▷ El archivo [hello.exe](#) es un archivo binario que contiene el programa ejecutable que se obtiene al linkar [hello.o](#).
  - ▷ El archivo [hello.exe.txt](#) muestra los bytes que forman el archivo [hello.exe](#),
  - ▷ y el archivo [hello.exe.b.txt](#) muestra los correspondientes bits.





# Palabra (I)

## WORD (palabra)

- ▶ Grupos de  $n$  bits (con  $2^n$  estados posibles) con los que el **procesador** es capaz de realizar una operación elemental (comparación, suma, etc.) en un ciclo de reloj.
- ▶ Típicamente:
  - $n = 8$  [70's]  $\implies$  Intel 8008, Z80 (Spectrum), ...
  - $n = 16$  [80's]  $\implies$  Intel 8086–80286 (IBM-PC/XT/AT), AMD Am386, PDP-11, ...
  - $n = 32$  [90's]  $\implies$  Intel 80386–80486–Pentium I/II/III/4, AMD K5–K6, VAX, ...
  - $n = 64$  [00's]  $\implies$  Intel Core 2, AMD Athlon 64, AXP, ...
  - $n = 128$  [??'s]  $\implies$  Videojuegos (de momento).
- ▶ La longitud de palabra determina el formato preferente de almacenamiento de datos numéricos (enteros y reales).





## Palabra (II)

- ▶ Variable capaz de almacenar  $2^n$  números enteros (sin signo)

$$word \in \{0, \dots, 2^n - 1\}.$$

- ▶ La longitud de palabra ( $n$  bits) determina la capacidad de direccionamiento directo en memoria (\*)...

Típicamente:

- $n = 8$  [70's]  $\implies$  256 B
- $n = 16$  [80's]  $\implies$  64 KiB
- $n = 32$  [90's]  $\implies$  4 GiB
- $n = 64$  [00's]  $\implies$  16 EiB  $\approx 16 \cdot 10^9$  GiB
- $n = 128$  [??'s]  $\implies$   $256 \cdot 10^{12}$  YiB  $\approx 256 \cdot 10^{27}$  GiB

---

(\*) Algunos procesadores utilizan varias palabras lo que permite **extender** su capacidad de direccionamiento.





## Palabra (III)

► Ejemplo:



Detalle del Panel frontal de un ordenador HP 2116-B (1966 aprox.): procesador de 16 bits, circuitos integrados, 16 KiB de memoria (expandible a 64 KiB), programable en Ensamblador y FORTRAN. Obsérvense los interruptores que permiten introducir palabras en binario. (Fuente: <http://www.hp-museum.net>)





# Tipos de Variables (I)

## TIPOS ELEMENTALES

### ▶ **VARIABLES ENTERAS: CON SIGNO (+, -) Y SIN SIGNO (+)**

- **Enteros de 8 bits**      —→      1 byte
- **Enteros de 16 bits**      —→      2 bytes
- **Enteros de 32 bits**      —→      4 bytes
- **Enteros de 64 bits**      —→      8 bytes

### ▶ **VARIABLES REALES (en coma flotante)**

- **Reales de 32 bits**      —→      4 bytes
- **Reales de 64 bits**      —→      8 bytes
- **Reales de 128 bits**      —→      16 bytes

### ▶ **VARIABLES NO NUMÉRICAS**

- **Lógicas**      —→      1–8 bytes
- **Alfanuméricas**      —→      "n" bytes





## Tipos de Variables (II)

### RANGO DE LAS VARIABLES NUMÉRICAS (valores más comunes):

TIPO	BITS	MÍNIMO	MÁXIMO
<b>SIN SIGNO</b>	8	0	$2^8 - 1 = 255$
(idem.)	16	0	$2^{16} - 1 = 65.535$
(idem.)	32	0	$2^{32} - 1 = 4.294.967.295$
(idem.)	64	0	$2^{64} - 1 = 18.446.744.073.709.551.615$
<b>ENTERO</b>	8	$-2^7 = -128$	$2^7 - 1 = 127$
(idem.)	16	$-2^{15} = -32.768$	$2^{15} - 1 = 32.767$
(idem.)	32	$-2^{31} = -2.147.483.648$	$2^{31} - 1 = 2.147.483.647$
(idem.)	64	$-2^{64} = 9.223.372.036.854.775.808$	$2^{64} - 1 = 9.223.372.036.854.775.807$
<b>REAL</b>	32	$-2^{128} \approx -3.403 \cdot 10^{38}$	$2^{128} \approx 3.403 \cdot 10^{38}$
(idem.)	64	$-2^{1024} \approx -1.798 \cdot 10^{308}$	$2^{1024} \approx 1.798 \cdot 10^{308}$
(idem.)	128	$-2^{16384} \approx -1.190 \cdot 10^{4932}$	$2^{16384} \approx 1.190 \cdot 10^{4932}$





## Tipos de Variables (III)

### PRECISIÓN DE LAS VARIABLES NUMÉRICAS (valores más comunes):

TIPO	BITS	MÍNIMO  $\neq 0$		ERROR DE MÁQUINA	
<b>SIN SIGNO</b>	8	1		0	
(idem.)	16	1		0	
(idem.)	32	1		0	
(idem.)	64	1		0	
<b>ENTERO</b>	8	1		0	
(idem.)	16	1		0	
(idem.)	32	1		0	
(idem.)	64	1		0	
<b>REAL</b>	32	$2^{-126} \approx$	$1.175 \cdot 10^{-38}$	$2^{-24} \approx$	$5.960 \cdot 10^{-8}$
(idem.)	64	$2^{-1022} \approx$	$2.225 \cdot 10^{-308}$	$2^{-53} \approx$	$1,110 \cdot 10^{-16}$
(idem.)	128	$2^{-16382} \approx$	$2.362 \cdot 10^{-4932}$	$2^{-113} \approx$	$9.630 \cdot 10^{-35}$





## Tipos de Variables (IV)

### DECLARACIÓN DE TIPOS ELEMENTALES (en máquinas de 32 bits):

TIPO	BITS	FORTRAN	LENGUAJE C
SIN SIGNO	8		unsigned char
(idem.)	16		unsigned short int
(idem.)	32		unsigned int, unsigned long int
(idem.)	64		unsigned long long
ENTERO	8	BYTE, INTEGER*1	char
(idem.)	16	INTEGER*2	short int
(idem.)	32	INTEGER*4, INTEGER	int, long int
(idem.)	64	INTEGER*8	long long
REAL	32	REAL*4, REAL	float
(idem.)	64	REAL*8, DOUBLE PRECISION	double
(idem.)	128	REAL*16, QUADRUPLE PRECISION	long double
COMPLEJO	64–256	COMPLEX*4/*8/*16, COMPLEX	
LÓGICO	8–64	LOGICAL*1/*2/*4/*8, LOGICAL	
ALFANUM.	$8 \cdot n$	CHARACTER*(n)	

