Fundamentos de Informática
E.U.P. Universidad de Sevilla

Capítulo 1:
Representación de la información

Departamento de Arquitectura y Tecnología de Computadores
UNIVERSIDAD DE SEVILLA

1

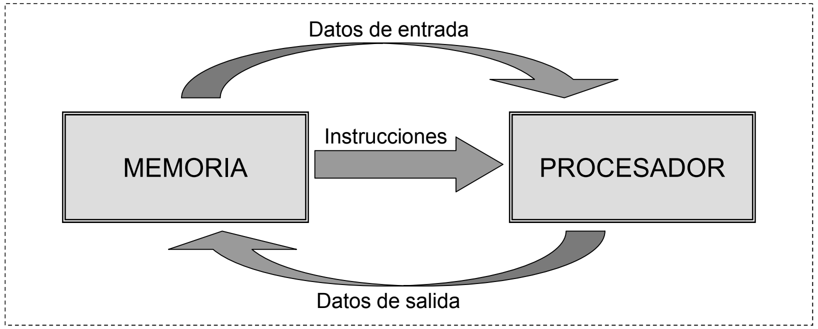
2

Índice

3

Introducción

Organización básica de un computador: **Modelo de *Von Neumann***



- ¿Cómo se representan los datos e instrucciones en el computador?
- ¿Cómo se representan los números?

4

Sistemas de representación numérica
Introducción

- Representación de los números, ejemplos:
  - Números Romanos
    I, II, III, IV, V, VI, VII, VIII, IX, X, ...
  - Sistema decimal
    0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, ...
- Un mismo número o cantidad se representa de forma diferente en cada sistema:
  - Ejemplos:

III	3
X	10
XXI	21

5

Sistemas de representación numérica
Bases de numeración

- **Base de numeración**: Nº de signos diferentes utilizados por el Sistema de Numeración para representar los números.
- Cada signo se denomina **dígito**.

Sistema de numeración	Base de numeración	Nº de dígitos	Dígitos usados
Decimal	Base 10	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Binario	Base 2	2	0, 1
Hexadecimal	Base 16	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

- Para indicar la base de un número se utiliza la siguiente notación:

Decimal	$23_{(10}$	$110_{(10}$	$5_{(10}$
Binario	$101_{(2}$	$110_{(2}$	$1_{(2}$
Hexadecimal	$23_{(16}$	$110_{(16}$	$A3F0_{(16}$

6

Sistemas de representación numérica
Sistema decimal

- Base 10 (10 dígitos): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Cada dígito tiene un valor diferente (peso) según su posición:

...	Centenas	Decenas	Unidades
...	100	10	1
...	10^2	10^1	10^0

Potencias de 10

Más significativo ← Menos significativo

- Ejemplo:
  324 $= 3\times10^2 + 2\times10^1 + 4\times10^0 = 300 + 20 + 4$
- Las operaciones aritméticas son fáciles de realizar siguiendo una serie de reglas

7

Sistemas de representación numérica
Sistema binario (1)

- Base 2 (2 dígitos): 0, 1

Binario	**0**	**1**	**10**	**11**	**100**	**101**	**110**	**111**	**...**
Decimal	0	1	2	3	4	5	6	7	...

- Cada dígito tiene un valor diferente (peso) según su posición:

...	2^3	2^2	2^1	2^0
...	8	4	2	1

Potencias de 2

Más significativo ← Menos significativo

- Ejemplo:
  $\mathbf{10110}_{(2} = 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0 = 16 + 4 + 2 = 22_{(10}$

2^4	2^3	2^2	2^1	2^0
16	8	4	2	1
1	**0**	**1**	**1**	**0**

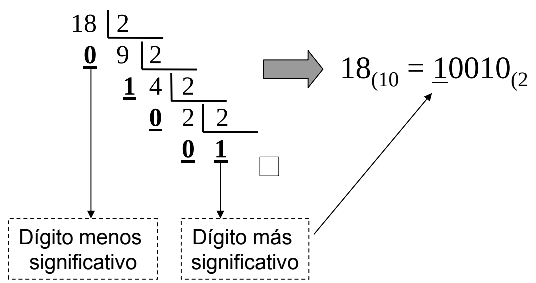
16 + 4 + 2

Conversión de un número binario a decimal

8

Sistemas de representación numérica
Sistema binario (2)

- Conversión de decimal a binario mediante divisiones sucesivas



9

Sistemas de representación numérica
Sistema binario (3): Operaciones aritméticas

- Suma
  - 0 + 0 = 0
  - 0 + 1 = 1
  - 1 + 1 = 10
- Producto
  - $0 \times 0 = 0$
  - $0 \times 1 = 0$
  - $1 \times 1 = 1$
- Las operaciones aritméticas con números binarios se realizan utilizando reglas similares a las del sistema decimal

Acarreo

```
  1     1
   1 0 0 1 0 1 1
 + 1 1 0 0 0 1 0
 ---------------
 1 0 1 0 1 1 0 1
```

```
      1 1 0
    × 1 1
    -------
      1 1 0
    1 1 0
    -------
  1 0 0 1 0
```

10

Sistemas de representación numérica
Sistema hexadecimal

- **Base 16** (16 digitos): 0 al 9 y A a la F

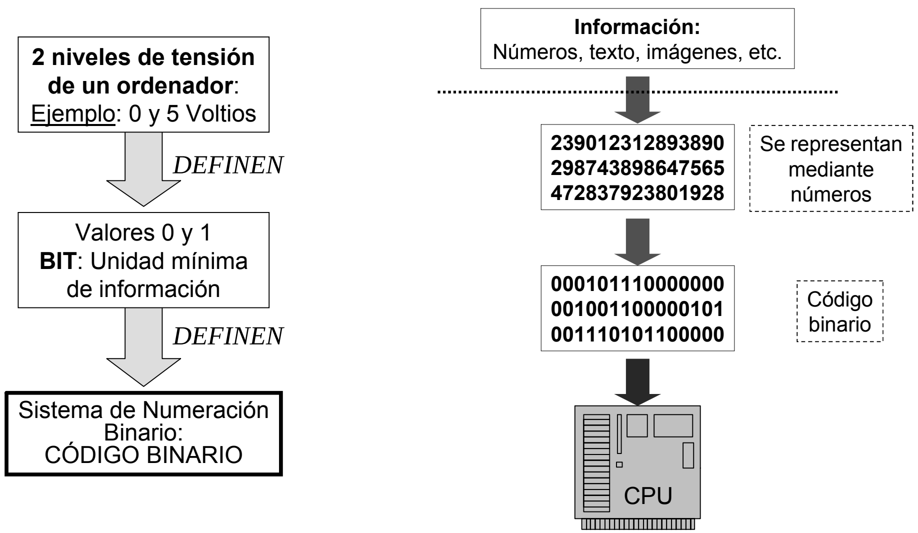
Hexadecimal	Binario	Decimal	Hexadecimal	Binario	Decimal
0	0000	0	**8**	1000	8
1	0001	1	**9**	1001	9
2	0010	2	**A**	1010	10
3	0011	3	**B**	1011	11
4	0100	4	**C**	1100	12
5	0101	5	**D**	1101	13
6	0110	6	**E**	1110	14
7	0111	7	**F**	1111	15

- Conversiones entre números hexadecimales y binarios: Cada dígito hexadecimal se representa con 4 dígitos binarios

$$247_{(10} = \underbrace{1111}\,\underbrace{0111}_{(2} = F7_{(16}$$

11

Representación de la información en el computador: El sistema binario



12

Representación de números naturales en el computador (1)

- Los computadores utilizan un número fijo de bits para representar los valores.
- Con N bits se pueden representar 2^N números naturales: desde 0 hasta 2^N-1
  - 1 bit, 2^1 valores: 0 y 1
  - 2 bits, 2^2 = 4 valores: 00, 01, 10, 11
  - 3 bits, 2^3 = 8 valores: 000, 001, 010, 011, 100, 101, 110, 111
  - etc.
- 1 **byte** → 8 **bits**

0	0	0	0	2 (0–1); 4 (0–3); 8 (0–7)
0	0	1	1	
0	1	0	2	
0	1	1	3	
1	0	0	4	
1	0	1	5	
1	1	0	6	
1	1	1	7	

Representación de números naturales en el computador (2)

- Para manejar cómodamente números binarios "grandes" se definen una serie de cantidades a modo de referencia, a las que se les da un nombre:

Kilo **K** = 2^{10} = 1024	**Kilo bits** **Kb** = 2^{10} bits	**Kilo bytes** **KB** = 2^{10} bytes
Mega **M** = 2^{20} = 1024K	**Mega bits** **Mb** = 2^{20} bits = 1024Kb	**Mega bytes** **MB** = 2^{20} bytes = 1024KB
Giga **G** = 2^{30} = 1024M	**Giga bits** **Gb** = 2^{30} bits = 1024Mb	**Giga bytes** **GB** = 2^{30} bytes = 1024MB

NOTA: a veces en telecomunicaciones y en otros entornos un "K-bit" (Kb) equivale a 10^3, un "Mega-bit" a 10^6 bits y un "Giga-bit" a 10^9 bits

- **Ejemplo**: Con 2 bits podemos numerar hasta cuatro bytes, ¿cuántos bytes podemos numerar con 32 bits?
Solución: con N bits podemos representar 2^N números, por lo tanto con 32 bits podemos numerar 2^{32} bytes = $2^2 \times 2^{30}$ bytes = 2^2 GB = 4 GB

Representación del signo
Introducción

- Introducción
  - En el sistema decimal se añade los símbolos **+** y **–** para expresar el signo.
  - Puesto que el computador sólo puede almacenar 0 y 1, no puede utilizar un símbolo nuevo para los nº negativos
- Para representar números negativos se utilizan diferentes codificaciones que utilizan el bit más significativo para indicar el signo del número.
  - Signo magnitud
  - Complemento a dos
- NOTA: los números hexadecimales se utilizan para representar números binarios naturales de forma compacta; por lo tanto, no tienen signo.

Representación del signo
Signo magnitud (1)

- Se añade un bit de signo que vale 0 para los números positivos y 1 para los negativos:

N bits

Signo	Magnitud
1 bit	**N-1 bits**

- Ejemplos:

-3 → 1011	-4 → 1100	-7 → 1111
+3 → 0011	+4 → 0100	+7 → 0111

- Existen dos representaciones del cero: +0 y –0

+0	0 00 ... 0
–0	1 00 ... 0

Representación del signo
Signo magnitud (2)

- Rango de valores:
  - Con N bits tenemos 1 bit de signo y N−1 bits de magnitud, por lo tanto podemos representar 2^{N-1} valores positivos y 2^{N-1} valores negativos
  - Los valores positivos van de +0 hasta $2^{N-1}-1$
  - Los valores negativos van de –0 hasta $-(2^{N-1}-1)$
  - El rango de valores que podemos representar con N bits es:

$$\mathbf{[\ -(2^{N-1}-1),\ \ 2^{N-1}-1\ ]}$$

  - Ejemplo:
    Con 8 bits podemos representar $[-(2^7-1), 2^7-1]$ → $[-127,127]$
- Ventajas:
  - Representación intuitiva y fácil
- Inconvenientes:
  - Existen dos representaciones del 0
  - El signo se trata de un modo diferente en las sumas y restas → Mayor complejidad del Hardware del ordenador

Representación del signo
C2: Complemento a dos (1)

- Características generales:
  - El bit más significativo indica el signo: vale 0 para los nº positivos y 1 para los negativos
  - Existe una sola representación del cero
- Rango de valores:
  - Con N bits tenemos 1 bit de signo y N–1 bits para representar los valores
  - Podemos representar 2^{N-1} valores positivos y 2^{N-1} valores negativos
  - Los valores positivos van de +0 hasta 2^{N-1}-1
  - Los valores negativos van de –1 hasta -2^{N-1}
  - El rango de valores que podemos representar con N bits es:

$$\mathbf{[\ -2^{N-1},\ \ 2^{N-1}-1\ ]}$$

  - <u>Ejemplo</u>:

    Con 8 bits podemos representar $[-2^7, 2^7\mathbf{-}1]$ → [-128,127]

Representación del signo
C2: Complemento a dos (2)

- Conversión de decimal a C2 con N bits (procedimiento A)
  - Sea $\boldsymbol{k}$ el valor que queremos convertir a C2
  - $\boldsymbol{k}$ debe pertenecer al rango $[\ -2^{N-1},\ \ 2^{N-1}-1\ ]$
  - Si $\boldsymbol{k} \geq \mathbf{0}$, su representación en C2 es su equivalente en binario natural

    <u>Ejemplo</u>:
    - 3 en C2 con 5 bits es **00011**
  - Si $\boldsymbol{k} < \mathbf{0}$, su representación en C2 es la representación en binario natural de $\mathbf{2^N-|\boldsymbol{k}|}$

    <u>Ejemplos</u>:
    - –3 en C2 con 5 bits es la representación en binario natural de $2^5 - 3 = 32 - 3 = 29$ → **11101**
    - –16 en C2 con 5 bits es la representación en binario natural de $2^5 - 16 = 32 - 16 = 16$ → **10000**

Representación del signo
C2: Complemento a dos (3)

- Conversión de decimal a C2 con N bits (procedimiento B)
  - Sea $\boldsymbol{k}$ el valor que queremos convertir a C2
  - $\boldsymbol{k}$ debe pertenecer al rango $[\ -2^{N-1},\ \ 2^{N-1}-1\ ]$
  - Si $\boldsymbol{k} \geq \mathbf{0}$, su representación en C2 es su equivalente en binario natural (igual que en el procedimiento A)
  - Si $\boldsymbol{k} < \mathbf{0}$, realizamos los siguientes pasos:
    - Convertimos $|k|$ en binario natural con 5 bits
    - Cambiamos los unos por ceros y viceversa
    - Sumamos 1 al resultado anterior y nos quedamos con los N bits menos significativos (no tomamos el acarreo)

    <u>Ejemplo</u>: –3 en C2 con 5 bits

–3	→	3	→	00011	→	11100	→	11100
								+ 1
								11101 ← Resultado

Representación del signo
C2: Complemento a dos (4)

- Conversión de C2 con N bits a decimal
  - Si es un nº positivo (el bit más significativo es 0), lo convertimos a decimal igual que si fuera binario natural.

    <u>Ejemplo</u>: <u>**0**</u>0101 → 5
  - Si es un nº negativo (el bit más significativo es 1), realizamos los siguientes pasos:
    - Cambiamos los unos por ceros y viceversa
    - Sumamos 1 al resultado anterior y nos quedamos con los N bits menos significativos
    - Convertimos el resultado de la suma a decimal como si se tratara de un nº binario natural y negamos el valor obtenido

    <u>Ejemplo</u>: Conversión a decimal del nº 11011

11011	→	00100	→	00100				
				+ 1				
				00101	→	5	→	**–5**

Representación del signo
C2: Complemento a dos (5)

- Ventajas de la representación C2
  - Existe una sola representación del cero
  - Las sumas y restas se realizan como si fueran sumas, esto hace que el hardware necesario sea menos complejo

    Ejemplo: Supongamos que queremos calcular 5 – 1 utilizando nº en complemento a dos de 5 bits

```
–1 → 1 → 00001 → 11110 →  11110
                        +     1
                        -------
                          11111      Calculamos –1 en C2

 5  →  00101
– 1 → +11111
      ------
      100100 → 4                     Sumamos a 5 el valor –1
```

No se tiene en cuenta el acarreo

Representación del signo:
Conclusiones

- Una misma secuencia de bits puede representar a valores diferentes según el sistema de representación que se utilice
- Ejemplos:

Secuencia de bits	Binario natural	Signo magnitud	C2
1111	15	– 7	– 1
1000	8	– 0	– 8
0110	6	6	6

Representación de números con decimales: Punto fijo

- También se llama Coma fija
- El número de decimales es fijo.
- La parte entera se trabaja de igual forma a lo visto hasta ahora.
- La parte decimal se trata de forma parecida pero las potencias de 2 adquieren exponentes negativos. Sólo comentamos para pasar de binario a decimal con números positivos. El proceso inverso (de decimal a binario) puede no dar un número exacto y se realiza aproximando potencias de 2 negativas.
- Ejemplo: ¿10.011 en decimal?

  $10.011 = 1x2^{1} + 0x2^{0} + 0x2^{-1} + 1x2^{-2} + 1x2^{-3} = 2 + 0 + 0 + 0.25 + 0.125 = 2.375$

Representación de números con decimales: Punto flotante (1)

- También se llama coma flotante: Se usa la normativa estándar IEEE 754
- El valor se representa como:

  $$(-1)^{\text{signo}} \times \text{mantisa} \times 2^{\text{exponente}}$$

  - **Por ejemplo:**

    $2.375_{(10} = 10.011_{(2}$ se representa como $(-1)^{0} \times 1.0011 \times 2^{1}$

    $-0.9375_{(10} = -0.10011_{(2}$ se representa como $(-1)^{1} \times 1.0011 \times 2^{-1}$
- Tres campos: signo (1 bit), mantisa y exponente
  - Signo: 0 positivo y 1 negativo.
  - Mantisa: Se normaliza de forma que obtengamos un número en coma fija de la forma 1.XXXXX y se almacena solamente la parte fraccionaria
  - Exponente: Puede ser un valor positivo o negativo que se codifica de forma parecida a C2. Con N bits, el exponente se codifica convirtiendo a binario natural el valor resultante de la siguiente expresión:

    $$\text{bits\_exponente} = \text{valor\_exponente} + (2^{N-1} - 1)$$

    A partir del código del exponente podemos obtener su valor:

    $$\text{valor\_exponente} = \text{bits\_exponente} - (2^{N-1} - 1)$$

Representación de números con decimales: Punto flotante (2)

- La norma IEEE 754 define los siguientes formatos:
  - Precisión simple Rango aproximado $[-10^{38}, 10^{38}]$
    valor = $(-1)^{signo} \times 1.mantisa \times 2^{\text{exponente}-127}$

signo	exponente	mantisa
1 bit	8 bits	23 bits

  - Precisión doble Rango aproximado $[-10^{308}, 10^{308}]$
    valor = $(-1)^{signo} \times 1.mantisa \times 2^{\text{exponente}-1023}$

signo	exponente	mantisa
1 bit	11 bits	52 bits

  - Precisión extendida Rango aproximado $[-10^{4932}, 10^{4932}]$
    valor = $(-1)^{signo} \times 1.mantisa \times 2^{\text{exponente}-16383}$

signo	exponente	mantisa
1 bit	15 bits	64 bits

Representación de la lógica
Álgebra de Boole

- Dos valores: **0** (FALSO) y **1** (VERDADERO)
  - En el álgebra de Boole no existen valores como el 10 o 11.
- Un conjunto de operaciones definidas sobre los valores anteriores:
  - Operaciones UNARIAS:
    - Negación lógica (**NOT**): A'
  - Operaciones BINARIAS:
    - Y lógico (**AND**): A • B
    - O lógico (**OR**): A + B
    - O exclusivo (**XOR**): A ⊕ B

NOT + prioritaria
AND
OR
XOR - prioritaria

A	A'
0	1
1	0

  - La operación OR o suma lógica no tiene nada que ver con la suma aritmética de números binarios:
    1 + 1 = 10 (suma aritmética)
    1 + 1 = 1 (álgebra de Boole)

A	B	A • B	A + B	A ⊕ B
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Álgebra de Boole
Leyes fundamentales (1)

	OR (suma)	AND (producto)
Ley conmutativa	A+B = B+A	A • B = B • A
Ley asociativa	A + (B + C) = (A + B) + C	A • (B • C) = (A • B) • C
Ley distributiva	A + B • C = (A + B) • (A + C)	A • (B + C) = A • B + A • C
Ley de idempotencia	A + A = A	A • A = A
Ley de absorción	A + A • B = A	A • (A + B) = A
Ley de De Morgan	(A + B)' = A' • B'	(A • B)' = A' + B'
Ley de involución	(A')' = A	
Leyes del 0 y 1 (elementos neutros)	A + A' = 1 A + 0 = A A + 1 = 1	A • A' = 0 A • 1 = A A • 0 = 0

Álgebra de Boole
Leyes fundamentales (2)

- **Principio de dualidad**: a todas las leyes mostradas en la transparencia anterior le corresponde su dual, que se obtiene intercambiando los operadores + y •, así como los 0 y 1
- Los paréntesis alteran el orden de evaluación
- El Álgebra de Boole puede ser útil a la hora de operar y simplificar expresiones de condición en nuestros programas

Ejemplo:
Simplificar la expresión S = (A + B')' • A + B • A

S = (A + B')' • A + B • A = A' • B • A + B • A = (A' • B + B) • A = B • A

Representación de textos
Código ASCII (1)

- Para representar texto, se asigna a cada carácter un valor numérico (o código) que lo identifica.
- Código ASCII (*American Standard Code for Interchange)*
  - Codifica 128 elementos distintos utilizando 7 bits: letras del abecedario, dígitos decimales y caracteres especiales (caracteres de puntuación, etc.)
  - El código ASCII extendido utiliza 8 bits para codificar 256 elementos: ñ, á, é, í, ó, ú, ...

Carácter	ASCII		Carácter	ASCII	
	Decimal	Binario		Decimal	Binario
A	65	100 0001	0	48	011 0000
B	66	100 0010	1	49	011 0001
C	67	100 0011	…	…	…
…	…	…	9	57	011 1001
Z	90	101 1010	…	…	…
…	…	…	=	61	011 1101
a	97	110 0001	…	…	…

- Por ejemplo, la palabra "**Hola**" se representa con la secuencia de valores {$72_{(10}$, $111_{(10}$, $108_{(10}$, $97_{(10)}$} representados en binario natural: {$01001000_{(2}$, $01101111_{(2}$, $01101100_{(2}$, $01100001_{(2)}$}

Representación de textos
Código ASCII (2)

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Representación de textos
Código ASCII (3)

- Los caracteres del código ASCII están ordenados:
  - Las letras aparecen en orden alfabético

    Ejemplos:
    - El código ASCII de la 'a' (97) más 1 es el código ASCII de la 'b' (98):
      ASCII('a') + 1 → 97 + 1 → 98 → ASCII('b')
    - El código ASCII de la 'd' (100) menos el código ASCII de la 'a' (97) nos da la distancia que hay entre las letras 'a' y 'd':
      ASCII('d') – ASCII('a') → 100 – 97 → 3
  - Las letras mayúsculas aparecen antes que las letras minúsculas (sus códigos ASCII son menores)
  - Los dígitos del 0 al 9 aparecen en orden creciente

    Ejemplos:
    - El código ASCII del carácter '0' + 3 es el código ASCII del carácter '3':
      ASCII('0') + **3** → 48 + 3 → 51 → ASCII(**'3'**)
    - El código ASCII de cualquier dígito menos el del '0' nos da el valor al que representa dicho dígito:
      ASCII(**'2'**) – ASCII('0') → 50 – 48 → **2**
      ASCII(**'7'**) – ASCII('0') → 55 – 48 → **7**