Unidad 3. Codificación digital

Circuitos Electrónicos Digitales
E.T.S.I. Informática
Universidad de Sevilla

Jorge Juan <jjchico@dte.us.es>

Contenidos

- Codificación digital
- Unidades digitales
- Números binarios
- Octal y hexadecimal
- Números fraccionarios
- Códigos binarios

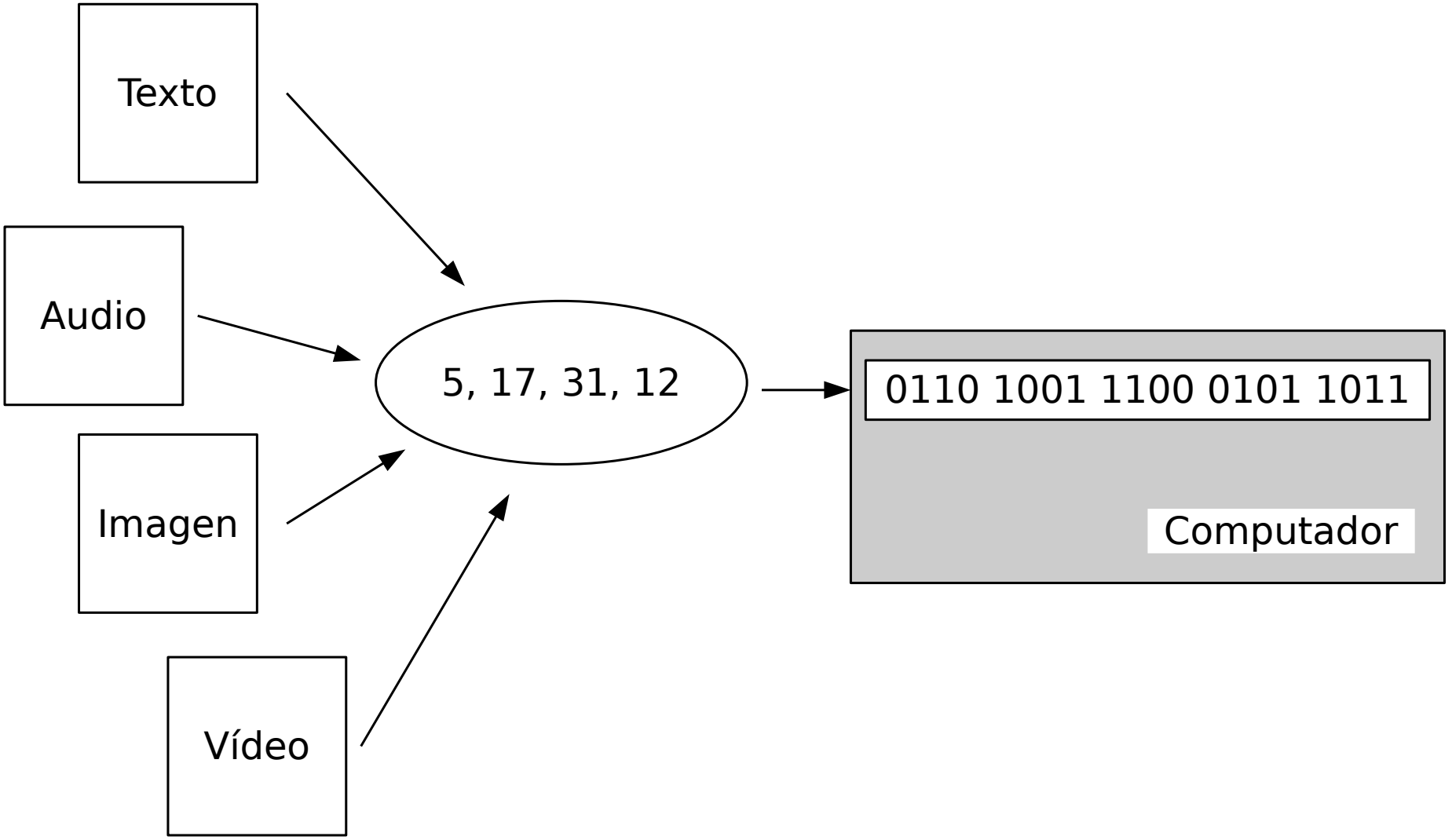
Competencias

- Competencias principales
  1) Expresar cantidades en bits y bytes en unidades del sistema internacional y del sistema binario.
  2) Representar cualquier número decimal con parte fraccionaria finita en cualquier base, y viceversa. (F)
  3) Realizar cambios de base directos entre la base 2 y 8, y la base 2 y 16. (F)
  4) Representar valores decimales mediante códigos BCD y saber interpretar datos binarios codificados en BCD.
  5) Determinar la paridad (par o impar) de una palabra binaria y saber calcular el bit de paridad par o impar de una palabra binaria.
- Otras competencias
  - Saber construir la tabla de código Gray de cualquier número de bits.
  - Calcular el tamaño sin comprimir de imágenes y fragmentos de audio conocidos sus parámetros de codificación.

Lecturas recomendadas

- Parity bit (wikipedia)
- Codificación de caracteres (wikipedia)
- ASCII (wikipedia)
- Unicode (wikipedia)
- Imagen de mapa de bits (wikipedia)
- Codificadores de forma de onda (wikipedia)
- Modulación por impulsos codificados –PCM– (wikipedia)

Codificación digital



Unidades digitales

- BIT (b) (BInary digiT)
  - Símbolo del conjunto {0,1}
  - Unidad mínima de información
- Palabra
  - Conjunto de 'n' bits, típicamente 4, 8, 16, 32 o 64.
  - Los ordenadores operan con palabras completas
- Nibble – Cuarteto (¿quién usa esto?)
  - Palabra de 4 bits
- Byte – Octeto (B)
  - Palabra de 8 bits
  - Unidad base en computación y telecomunicaciones

Unidades digitales

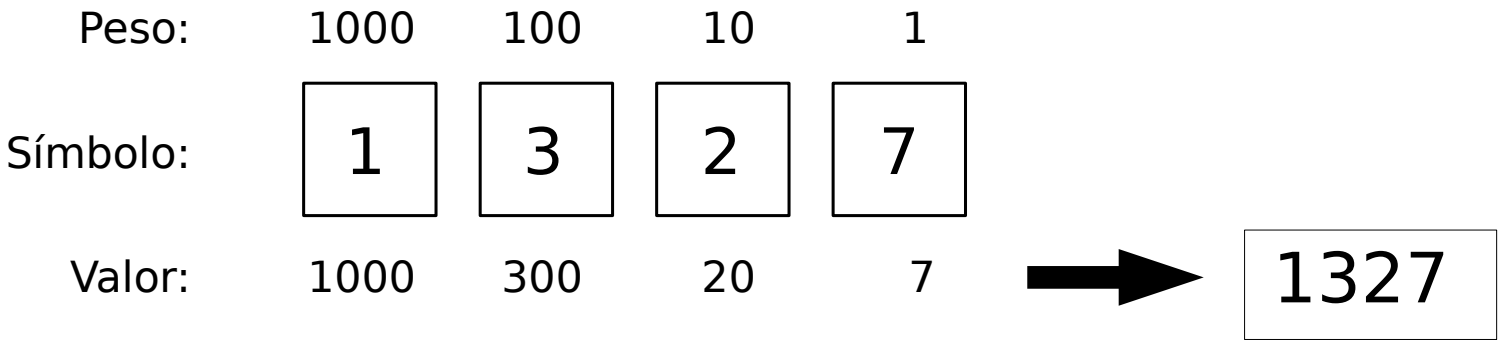
- Tradición: Unidades del SI con significado ligeramente modificado (potencias de 2 en vez de 10)
- Uso no uniforme de unidades digitales:
  - Diskette: 1.44MB = 1000 KB = 1000x1024B
  - Discos 160GB = 160000 MB = 160x1000x1024x1024B
  - DVD 4,7GB = 4700MB = 4,7x1000x1024x1024B
- Estándar para unidades digitales binarias (no muy usado):
  - IEC, IEEE-1541-2002

SI			**Binary**	**IEC**	
kilo	k	10^3	2^{10}	kibi	Ki
mega	M	10^6	2^{20}	mebi	Mi
giga	G	10^9	2^{30}	gibi	Gi
tera	T	10^{12}	2^{40}	tebi	Ti
peta	P	10^{15}	2^{50}	pebi	Pi
exa	E	10^{18}	2^{60}	exbi	Ei
zetta	Z	10^{21}	2^{70}	zebi	Zi

Números binarios. Notación posicional

- Sistema decimal:
  - 10 símbolos {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
  - Base 10

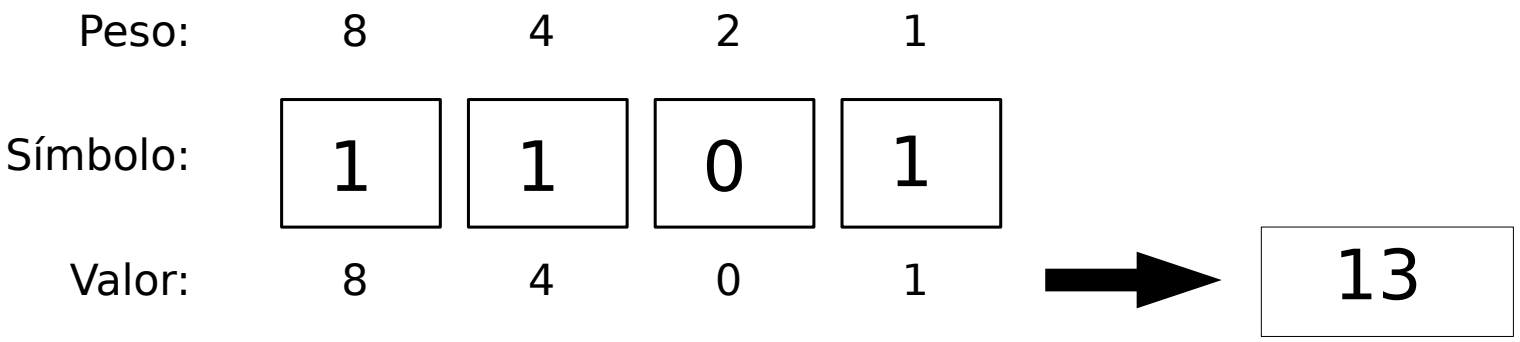
$$1327 = 1 \times 10^3 + 3 \times 10^2 + 2 \times 10^1 + 7 \times 10^0$$



Números binarios. Notación posicional

- Sistema binario:
  - 2 símbolos {0,1}
  - Base 2

$$1101 = 1\times 2^3 + 1\times 2^2 + 0\times 2^1 + 1\times 2^0$$



Números binarios. Notación posicional

- En general:
  - Magnitud x
  - Base b
  - n cifras: $\{x_i\}$

$$x = x_{n-1} \times b^{n-1} + \ldots + x_1 \times b^1 + x_0 \times b^0$$

Número máximo representable: b^n-1

Conversión de base *b* a base 10

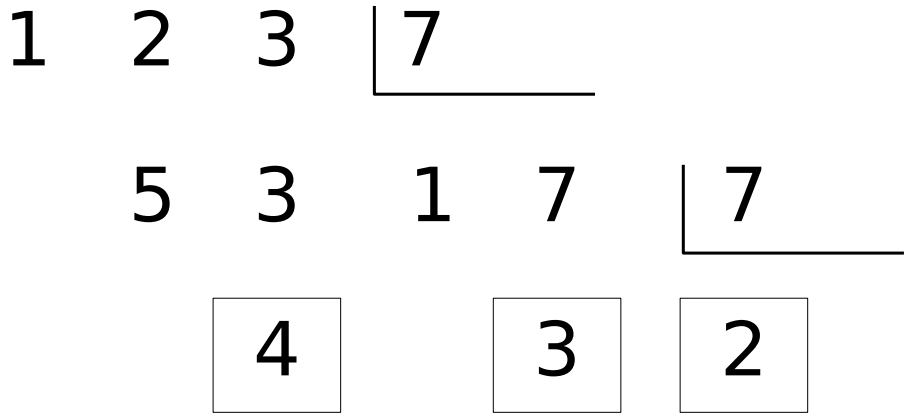
- Basta aplicar la fórmula y hacer los cálculos en base 10
  - Ej: $234_{(7}$

$$x = x_{n-1} \times b^{n-1} + \ldots + x_1 \times b^1 + x_0 \times b^0$$

Conversión de base 10 a base *b*

- Divisiones sucesivas por la base objetivo
  - Los restos sucesivos son las cifras de la representación en base n, empezando por las menos significativas.

$123_{(10} \rightarrow 234_{(7}$



Octal y hexadecimal

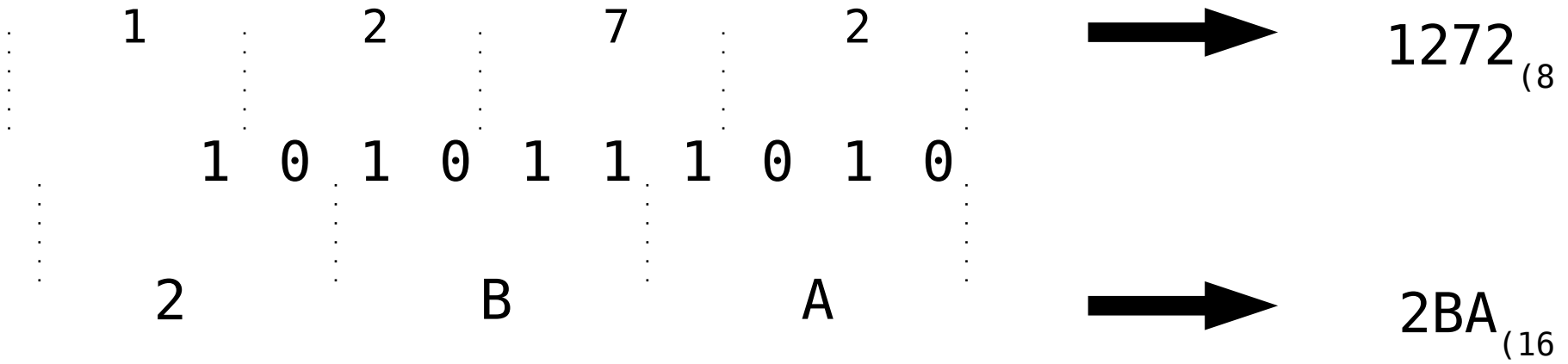
- Base 8 (octal):
  - {0, 1, 2, 3, 4, 5, 6, 7}
- Base 16 (hexadecimal):
  - {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}
- Formas compactas de representar números binarios
  - 1 cifra octal = 3 cifras binarias
  - 1 cifra hexadecimal = 4 cifras binarias

Octal y hexadecimal

B-8	B-2
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

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

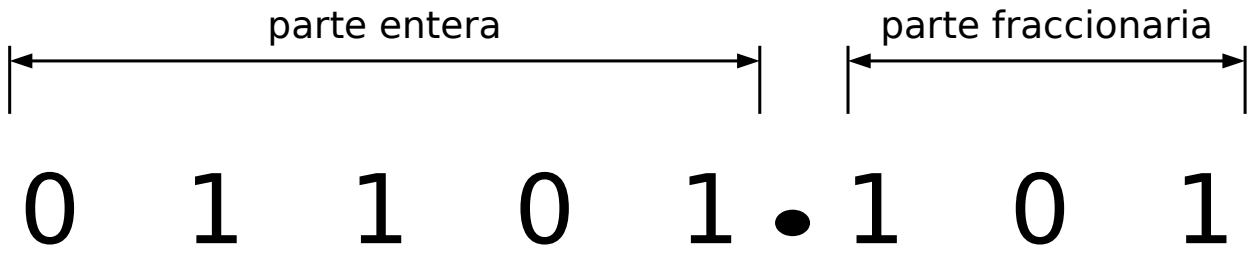
Octal y hexadecimal



$2BA_{(16} = 2BA_{h} = 2BA_{hex} = \#2BA = \$2BA = 0x2BA = 10'h2BA$

Números fraccionarios

$$x = x_{n-1} \times b^{n-1} + \ldots + x_0 \times b^0 + x_{-1} \times b^{-1} + \ldots + x_{-m} \times b^{-m}$$



Números fraccionarios

- Conversión de base *b* a base 10:
  - Directamente: basta operar en base 10

$$10{,}101_2 = 1\times 2^1 + 0\times 2^0 + 1\times 2^{-1} + 0\times 2^{-2} + 1\times 2^{-3}$$
$$2 + 1/2 + 1/8 = 2{,}625_{10}$$

- Conversión de base 10 a base *b*:
  - Parte entera: como con números enteros
  - Parte fraccionaria: multiplicaciones sucesivas por la base objeto. Se toma la parte entera del resultado

Números fraccionarios

- Ejemplo: $12,3_{(10}$
  - $12_{(10} = 1100_{(2}$
  - 0,3 x 2 = 0,6 → “0”
  - 0,6 x 2 = 1,2 → “1”
  - 0,2 x 2 = 0,4 → “0”
  - 0,4 x 2 = 0,8 → “0”
  - 0,8 x 2 = 1,6 → “1”
  - 0,6 x 2 = 1,2 → “1” (primer bit repetido)
- $12,3_{(10} = 1100,0100110011001..._{(2} = 1100,0\overline{1001}_{(2}$
- Al cambiar de base, un número puede pasar de tener un número finito a infinito de cifras (y viceversa).
- ¿Podrían las cifras no repetirse periódicamente?
- ¿Cuándo un número tendrá un número infinito de cifras en una base dada?

Números fraccionarios. Implicaciones para los sistemas digitales

- Si el número de bits está limitado (ej. 8) no podemos obtener una representación exacta en base 2 de muchos números que sí se pueden representar exactamente en base 10.
- Potencial fuente de graves errores, incluso a nivel software.

$$12{,}3_{10} = 1100{,}\,0\,\widehat{1001}_2 \approx 1100{,}0100_2 = 12{,}25_{10}$$

Números fraccionarios. Implicaciones para los sistemas digitales

```
$ python3

>>> x = 12.3     # el valor se almacena internamente en b. 2
>>> y = 3 * x    # las operaciones se realizan en b. 2
>>> z = y / 3

>>> x == z       # !?
False

>>> z - x
1.7763568394002505e-15

>>> from decimal import Decimal

>>> Decimal(x)   # rep. en b.10 del valor almacenado de x
Decimal('12.300000000000000710542735760100185871124267578125')
>>> Decimal(z)
Decimal('12.300000000000002486899575160350650548934936523437 5')

>>> if x != z:   # kaboooom!
...     print("Destruir el mundo!!!")
...
Destruir el mundo!!!
```

Bases en Python

```
$ python3

>>> x = 215
>>> hex(x)          # repr. hexadecimal
'0xd7'
>>> bin(x)          # repr. base 2
'0b11010111'

>>> y = 0b1011      # valor binario
>>> y
11
>>> z = 0x2c        # valor hexadecimal
>>> z
44

>>> x + y
226
>>> hex(x+y)
'0xe2'
>>> bin(x+y)
'0b11100010'

>>> x - z
171
>>> hex(x-z)
'0xab'
>>> bin(x-z)
'0b10101011'
```

```
>>> int('321',7)         # base 7
162

# 321(7 + 121(3
>>> x = int('321',7) + int('212',3)

>>> x
185
>>> hex(x)
'0xb9'
>>> bin(x)
'0b10111001'

# repr. en base n?

>>> from numpy import base_repr
>>> x = 52
>>> base_repr(x, 2)
'110100'
>>> base_repr(x, 3)
'1221'
>>> base_repr(x, 4)
'310'
>>> base_repr(x, 5)
'202'
>>> base_repr(x, 6)
'124'
>>> base_repr(x, 7)
'103'
```

Códigos binarios

- Asignación de valores binarios a un conjunto de símbolos
  - Números, caracteres (letras), colores, etc.
- Normalmente se emplean palabras de anchura fija (ej. 8 bits).
- La asignación del código se hace en base a la obtención de propiedades convenientes
  - Códigos similares para colores similares
  - Códigos similares para letras minúsculas y mayúsculas
  - Etc.
- No todas las palabras binarias tienen que tener un símbolo asociado.

Código binario natural

- Codificación de números enteros usando la palabra correspondiente al número expresado en base 2

Número	Código
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Código Gray

Número	1 bit	2 bits	3 bits	4 bits
0	0	00	000	0000
1	1	01	001	0001
2		11	011	0011
3		10	010	0010
4			110	0110
5			111	0111
6			101	0101
7			100	0100
8				1100
9				1101
10				1111
11				1110
...				...

- Codifica enteros positivos
- Símbolos consecutivos sólo se diferencian en 1 bit (distancia 1)
- El código de n bits se construye reflejando la tabla del código de n-1 bits.

Código *one-hot*

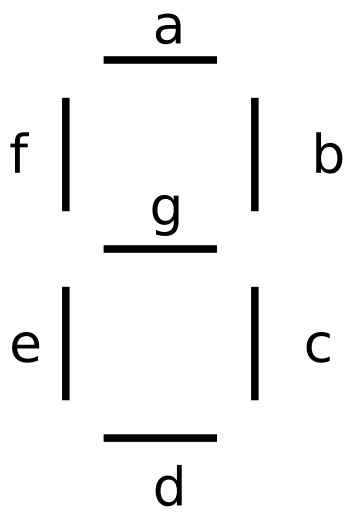
Número	Código
0	00000001
1	00000010
2	00000100
3	00001000
4	00010000
5	00100000
6	01000000
7	10000000

- Cada palabra del código sólo tiene un bit a 1
- Ventajas:
  - Fácil de decodificar
  - Permite detección de errores
- Inconvenientes
  - Palabras del código con número elevado de bits

Código BCD (*Binary Coding Decimal*)

- Codificación binaria de las 10 cifras decimales

Cifra	natural	exceso-3	2-4-2-1	2-out-of-5	7 segm. abcdefg
0	0000	0011	0000	00011	1111110
1	0001	0100	0001	00101	0110000
2	0010	0101	0010	00110	1101101
3	0011	0110	0011	01001	1111001
4	0100	0111	0100	01010	0110011
5	0101	1000	1011	01100	1011011
6	0110	1001	1100	10001	1011111
7	0111	1010	1101	10010	1110000
8	1000	1011	1110	10100	1111111
9	1001	1100	1111	11000	1110011



Bit de paridad

- La paridad de una palabra binaria es par/impar si el número de bits a “1” de la palabra es par/impar.
- Bit de paridad: bit adicional que se añade a un código para conseguir una paridad determinada, par o impar.

Ej. código binario natural con bit de paridad par inicial.

Número	Código
0	0000
1	1001
2	1010
3	0011
4	1100
5	0101
6	0110
7	1111

Codificación de texto

- A cada símbolo de texto “carácter” se asigna un número que se almacena en binario.
- Además de símbolos gráficos se incluyen símbolos de control: nueva línea, nueva página, fin de fichero, etc.
- Existen varios asignamientos llamados "codificaciones".

ASCII

0 00 NUL	32 20 SPC	64 40 @	96 60 `
1 01 SOH	33 21 !	65 41 A	97 61 a
2 02 STX	34 22 "	66 42 B	98 62 b
3 03 ETX	35 23 #	67 43 C	99 63 c
4 04 EOT	36 24 $	68 44 D	100 64 d
5 05 ENQ	37 25 %	69 45 E	101 65 e
6 06 ACK	38 26 &	70 46 F	102 66 f
7 07 BEL	39 27 '	71 47 G	103 67 g
8 08 BS	40 28 (	72 48 H	104 68 h
9 09 HT	41 29)	73 49 I	105 69 i
10 0A LF	42 2A *	74 4A J	106 6A j
11 0B VT	43 2B +	75 4B K	107 6B k
12 0C FF	44 2C ,	76 4C L	108 6C l
13 0D CR	45 2D -	77 4D M	109 6D m
14 0E SO	46 2E .	78 4E N	110 6E n
15 0F SI	47 2F /	79 4F O	111 6F o
16 10 DLE	48 30 0	80 50 P	112 70 p
17 11 DC1	49 31 1	81 51 Q	113 71 q
18 12 DC2	50 32 2	82 52 R	114 72 r
19 13 DC3	51 33 3	83 53 S	115 73 s
20 14 DC4	52 34 4	84 54 T	116 74 t
21 15 NAK	53 35 5	85 55 U	117 75 u
22 16 SYN	54 36 6	86 56 V	118 76 v
23 17 ETB	55 37 7	87 57 W	119 77 w
24 18 CAN	56 38 8	88 58 X	120 78 x
25 19 EM	57 39 9	89 59 Y	121 79 y
26 1A SUB	58 3A :	90 5A Z	122 7A z
27 1B ESC	59 3B ;	91 5B [	123 7B {
28 1C FS	60 3C <	92 5C \	124 7C |
29 1D GS	61 3D =	93 5D]	125 7D }
30 1E RS	62 3E >	94 5E ^	126 7E ~
31 1F US	63 3F ?	95 5F _	127 7F DEL

ISO/IEC-8859-15

ASCII

extensión

ISO-8859-15																
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
8x	PAD	HOP	BPH	NBH	IND	NEL	SSA	ESA	HTS	HTJ	VTS	PLD	PLU	RI	SS2	SS3
9x	DCS	PU1	PU2	STS	CCH	MW	SPA	EPA	SOS	SGCI	SCI	CSI	ST	OSC	PM	APC
Ax	NBSP	¡	¢	£	€	¥	Š	§	š	©	ª	«	¬	SHY	®	¯
Bx	°	±	²	³	Ž	µ	¶	·	ž	¹	º	»	Œ	œ	Ÿ	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

https://es.wikipedia.org/wiki/ISO/IEC_8859-15

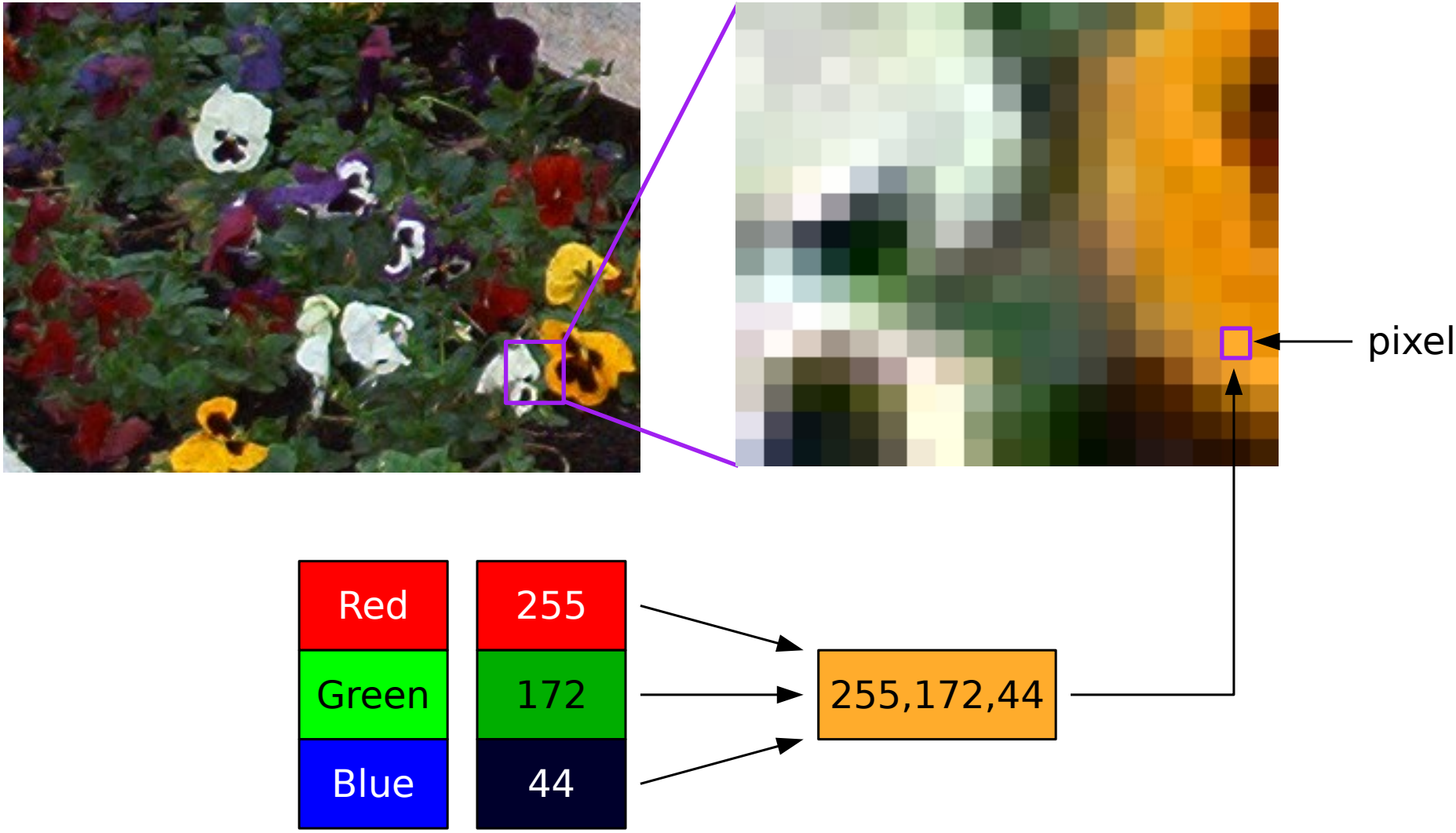
Códigos de control ASCII

- Pensados para ser usados en teletipos (muchos ya no se usan en la actualidad). Utilidad:
  - Formato: tabulador, fin de línea, etc.
  - Control: fin de transmisión, fin de texto, etc.
- Presentes en documentos de texto y en el terminal
  - Terminal: se introducen con la tecla Ctrl+<letra> (^<letra>)
- Ejemplos
  - ETX (^C): fin de texto
  - EOT (^D): fin de transmisión
  - BEL (^G): timbre
  - BS (^H): retroceso
  - HT (^I): tabulador horizontal
  - LF (^J): salto de línea
  - CR (^M): retorno de carro
  - DEL (^?): suprimir

Codificación de texto. Ejemplo

```
$ echo "Muñoz Pérez 5€ debe" > texto.txt

$ cat texto.txt
Muñoz Pérez 5€ debe

$ hd texto.txt
00000000  4d 75 c3 b1 6f 7a 20 50  c3 a9 72 65 7a 20 35 e2  |Mu..oz P..rez 5.|
00000010  82 ac 20 64 65 62 65 0a                           |.. debe.|

$ iconv -f utf8 -t latin1 texto.txt > texto_l1.txt
iconv: secuencia de entrada ilegal en la posición 15

$ iconv -f utf8 -t latin9 texto.txt > texto_l9.txt
$ cat texto_l9.txt
Mu?oz P?rez 5? debe

jjchico@valhalla:tmp $ hd texto_l9.txt
00000000  4d 75 f1 6f 7a 20 50 e9  72 65 7a 20 35 a4 20 64  |Mu.oz P.rez 5. d|
00000010  65 62 65 0a                                       |ebe.|

$ file texto.txt texto_l9.txt
texto.txt:    UTF-8 Unicode text
texto_l9.txt: ISO-8859 text

$ ls -l texto.txt texto_l9.txt
-rw-rw-r-- 1 jjchico jjchico 20 oct  1 16:27 texto_l9.txt
-rw-rw-r-- 1 jjchico jjchico 24 oct  1 16:23 texto.txt
```

Gráficos (imágenes)

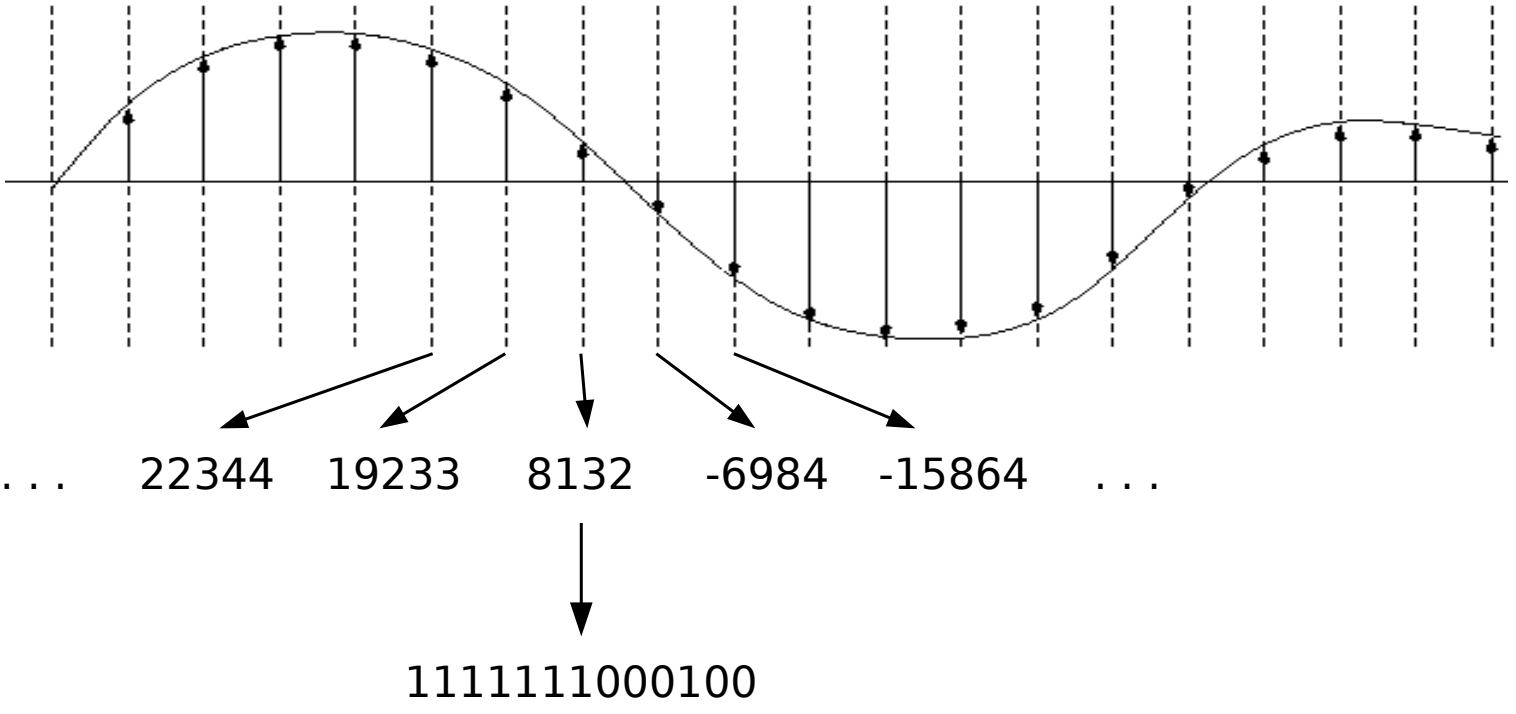


Gráficos. Cálculos

- Profundidad de color
  - Número de datos (bits) empleados en codificar el color de un pixel.
- Número de pixels de una imagen
  - no. pixels = anchura en pixel x altura en pixel
- Tamaño de datos de imagen sin comprimir
  - tamaño = número de pixels x profundidad de color
- Resolución de la imagen
  - resolución = no. pixel lineales / distancia
  - puede ser vertical y horizontal (suelen coincidir)

Sonido

- Parámetros de codificación (calidad)
  - Frecuencia de muestreo (ej. 44100Hz)
  - Resolución de bits (ej. 16 bits/muestra)
  - Número de canales (ej. 2 -stereo-)



Sonido. Cálculos

- Tasa de datos: datos empleados para codicar el sonido por unidad de tiempo.
- Tasa de datos sin comprimir
  - tasa datos = resolución bits x frec. muestreo x no. canales
- Tamaño de datos sin comprimir
  - tamaño datos = tasa datos x tiempo