Representación de la información (Repaso)

Sistemas Informáticos

Sistema de numeración decimal

- El sistema que solemos utilizar nosotros para contar es el sistema **decimal** o **base 10**, formado por 10 dígitos, del 0 al 9.
- Combinamos estos dígitos para formar cantidades: 0, 1, 2, 3, ... 10, ... 102, ... 438 ...
  - Observa que cuando se acaban los dígitos de 1 cifra, se vuelve al principio y se añade un dígito más, y lo mismo pasa cuando se acaban las 2 cifras, etc: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, **10**, 11, 12... 98, 99, **100**, 10, 102, ... 998, 999, 1000, etc.

Teorema fundamental de la numeración

Si tenemos un número en un sistema determinado, y lo descomponemos multiplicando cada uno de sus dígitos por la potencia de su base en la posición que está, y sumamos estos resultados, obtendremos el equivalente de ese número en decimal

En el sistema decimal...

$$8888 = 8x10^3 + 8x10^2 + 8x10^1 + 8x10^0 = 8000 + 800 + 80 + 8$$

Sistema de numeración binario

- En el mundo informático, como hemos visto, todo se representa utilizando dos estados, llamados bits.
- El sistema de numeración formado por estos dos elementos se llama **sistema binario o base 2**, y utiliza sólo dos dígitos: 0 y 1
  - La secuencia de números para contar en este sistema quedaría: 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, ...
- Cada número binario tiene su correspondiente número decimal: el 0 binario es el 0 decimal, el 1 es el 1, el 10 es el 2, el 11 es el 3, etc.
- Para almacenar la información, lo que se hace es convertir los datos de sistema decimal a binario, y luego para mostrarla otra vez, se convierte de binario a decimal

0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

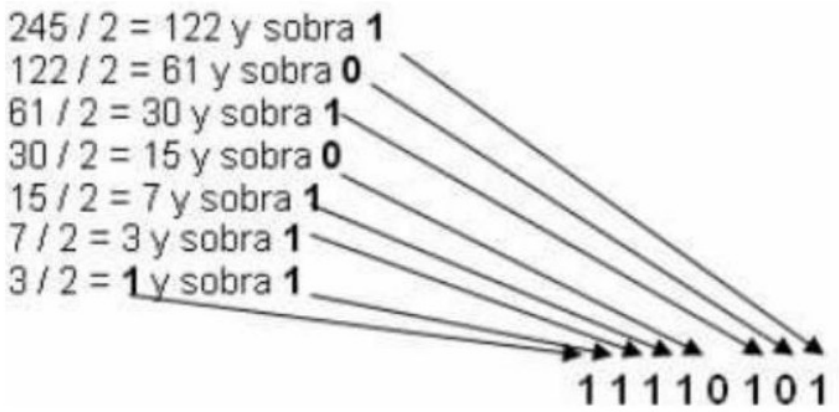
De binario a natural

- Consiste en separar los bits del número binario, y colocar cada una de las cifras debajo de una de las potencias de 2
- Sumando las potencias de 2 que coincidan con un 1 del número, tenemos el resultado en decimal
- Ejemplo: para el número binario 1101011_2

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	1	1	0	1	0	1	1
64+32+8+2+1= **107**							

De decimal a binario

- Para convertir un número de decimal a binario, tenemos que ir dividiéndolo entre 2 sucesivamente, y quedándonos con todos los restos, y el último cociente (cuando ya no se pueda dividir más)
  - Los restos y cociente serán 0 o 1
- Después, se pone primero el último cociente, y los restos desde el último al primero (en orden inverso a como hemos ido dividiendo)
- Ejemplo: para el número 245_{10}



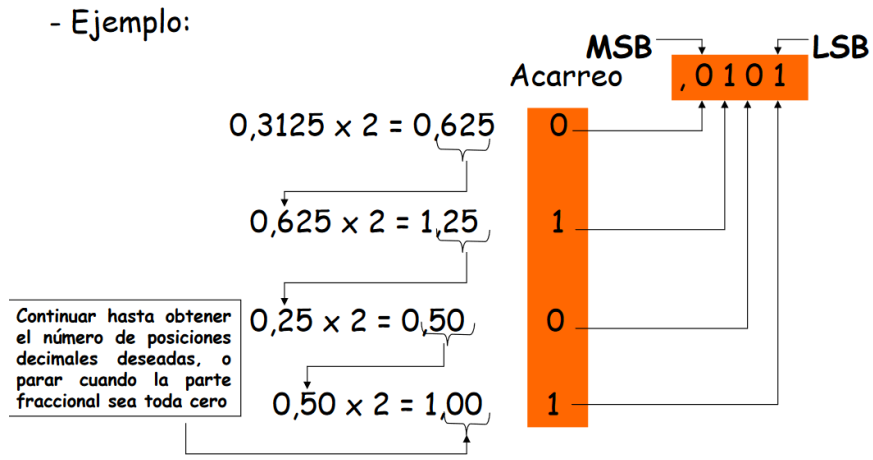
De decimal a binario

La parte decimal de un número la podemos convertir a binario con el método de las multiplicaciones sucesivas

Por ejemplo, para convertir a binario el número decimal fraccionario 0,3125, empezamos multiplicando por 2, y después se multiplica cada parte fraccional resultante del producto por 2, hasta que el producto fraccionario sea cero o hasta que se alcance el número deseado de posiciones decimales.

245 = 11110101
0,3125 = 0101

245,3125 = 11110101,0101



Código ASCII

- Es una tabla que codifica un total de 256 símbolos diferentes, asociándoles a cada uno un número (del 0 al 255)
  - Si el fichero de texto está guardado en formato ASCII (uno de los más utilizados), cada símbolo que se escribe se almacena directamente con su código
  - Ejemplo: si escribimos “Hola”:
    - H = 01001000
    - o = 01101111
    - l = 01101100
    - a = 01100001
  - Se almacenaría 01001000 01101111 01101100 01100001.

Código ASCII

Oct	Hex	Dec	Carácter
0	00	0	NUL NULl
1	01	1	SOH Start Of Heading
2	02	2	STX Start of TeXt
3	03	3	ETX End of TeXt
4	04	4	EOT End of Transmission
5	05	5	ENQ ENQuiry
6	06	6	ACK ACKnowledge
7	07	7	BEL BELl
10	08	8	BS BackSpace
11	09	9	TAB horizontal TAB
12	0A	10	LF new Line Feed
13	0B	11	VT Vertical Tab
14	0C	12	FF new page From Feed
15	0D	13	CR Carriage Return
16	0E	14	SO Shift Out
17	0F	15	SI Shift In
20	10	16	DLE Data Link Escape
21	11	17	DC1 Device Control 1
22	12	18	DC2 Device Control 2
23	13	19	DC3 Device Control 3
24	14	20	DC4 Device Control 4
25	15	21	NAK negative acknowledge
26	16	22	SYN SYNchronous idle
27	17	23	ETB End of Transmission. Block
30	18	24	CAN CANcel
31	19	25	EM End of Medium
32	1A	26	SUB SUBstitute
33	1B	27	ESC ESCape
34	1C	28	FS File Separator
35	1D	29	GS Group Separator
36	1E	30	RS Record Separator
37	1F	31	US Unit Separator

Oct	Hex	Dec	Carácter
40	20	32	
41	21	33	!
42	22	34	"
43	23	35	#
44	24	36	$
45	25	37	%
46	26	38	&
47	27	39	'
50	28	40	(
51	29	41	)
52	2A	42	*
53	2B	43	+
54	2C	44	,
55	2D	45	-
56	2E	46	.
57	2F	47	/
60	30	48	0
61	31	49	1
62	32	50	2
63	33	51	3
64	34	52	4
65	35	53	5
66	36	54	6
67	37	55	7
70	38	56	8
71	39	57	9
72	3A	58	:
73	3B	59	;
74	3C	60	<
75	3D	61	=
76	3E	62	>
77	3F	63	?

Oct	Hex	Dec	Carácter
100	40	64	@
101	41	65	A
102	42	66	B
103	43	67	C
104	44	68	D
105	45	69	E
106	46	70	F
107	47	71	G
110	48	72	H
111	49	73	I
112	4A	74	J
113	4B	75	K
114	4C	76	L
115	4D	77	M
116	4E	78	N
117	4F	79	O
120	50	80	P
121	51	81	Q
122	52	82	R
123	53	83	S
124	54	84	T
125	55	85	U
126	56	86	V
127	57	87	W
130	58	88	X
131	59	89	Y
132	5A	90	Z
133	5B	91	[
134	5C	92	\
135	5D	93	]
136	5E	94	^
137	5F	95	_

Oct	Hex	Dec	Carácter
140	60	96	`
141	61	97	a
142	62	98	b
143	63	99	c
144	64	100	d
145	65	101	e
146	66	102	f
147	67	103	g
150	68	104	h
151	69	105	i
152	6A	106	j
153	6B	107	k
154	6C	108	l
155	6D	109	m
156	6E	110	n
157	6F	111	o
160	70	112	p
161	71	113	q
162	72	114	r
163	73	115	s
164	74	116	t
165	75	117	u
166	76	118	v
167	77	119	w
170	78	120	x
171	79	121	y
172	7A	122	z
173	7B	123	{
174	7C	124	|
175	7D	125	}
176	7E	126	~
177	7F	127	DELETE

Oct	Hex	Dec	Carácter
200	80	128	Ç
201	81	129	ü
202	82	130	é
203	83	131	â
204	84	132	ä
205	85	133	à
206	86	134	å
207	87	135	ç
210	88	136	ê
211	89	137	ë
212	8A	138	è
213	8B	139	ï
214	8C	140	î
215	8D	141	ì
216	8E	142	Ä
217	8F	143	Å
220	90	144	É
221	91	145	æ
222	92	146	Æ
223	93	147	ô
224	94	148	ö
225	95	149	ò
226	96	150	û
227	97	151	ù
230	98	152	ÿ
231	99	153	Ö
232	9A	154	Ü
233	9B	155	ø
234	9C	156	£
235	9D	157	Ø
236	9E	158	×
237	9F	159	ƒ

Oct	Hex	Dec	Carácter
240	A0	160	á
241	A1	161	í
242	A2	162	ó
243	A3	163	ú
244	A4	164	ñ
245	A5	165	Ñ
246	A6	166	ª
247	A7	167	º
250	A8	168	¿
251	A9	169	®
252	AA	170	¬
253	AB	171	½
254	AC	172	¼
255	AD	173	¡
256	AE	174	«
257	AF	175	»
260	B0	176	▒
261	B1	177	▓
262	B2	178	▓
263	B3	179	│
264	B4	180	┤
265	B5	181	Á
266	B6	182	Â
267	B7	183	À
270	B8	184	©
271	B9	185	╣
272	BA	186	║
273	BB	187	╗
274	BC	188	╝
275	BD	189	¢
276	BE	190	¥
277	BF	191	┐

Oct	Hex	Dec	Carácter
300	C0	192	└
301	C1	193	┴
302	C2	194	┬
303	C3	195	├
304	C4	196	─
305	C5	197	┼
306	C6	198	ã
307	C7	199	Ã
310	C8	200	╚
311	C9	201	╔
312	CA	202	╩
313	CB	203	╦
314	CC	204	╠
315	CD	205	═
316	CE	206	╬
317	CF	207	¤
320	D0	208	ð
321	D1	209	Ð
322	D2	210	Ê
323	D3	211	Ë
324	D4	212	È
325	D5	213	ı
326	D6	214	Í
327	D7	215	Î
330	D8	216	Ï
331	D9	217	┘
332	DA	218	┌
333	DB	219	▒
334	DC	220	▄
335	DD	221	¦
336	DE	222	Ì
337	DF	223	▀

Oct	Hex	Dec	Carácter
340	E0	224	Ó
341	E1	225	ß
342	E2	226	Ô
343	E3	227	Ò
344	E4	228	õ
345	E5	229	Õ
346	E6	230	µ
347	E7	231	þ
350	E8	232	Þ
351	E9	233	Ú
352	EA	234	Û
353	EB	235	Ù
354	EC	236	ý
355	ED	237	Ý
356	EE	238	¯
357	EF	239	´
360	F0	240	-
361	F1	241	±
362	F2	242	‗
363	F3	243	¾
364	F4	244	¶
365	F5	245	§
366	F6	246	÷
367	F7	247	¸
370	F8	248	°
371	F9	249	¨
372	FA	250	·
373	FB	251	¹
374	FC	252	³
375	FD	253	²
376	FE	254	■
377	FF	255	

Octal y hexadecimal

- **Sistema octal** o **base 8**: utiliza los dígitos del 0 al 7: 0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, ... 17, 20, ... etc
- **Sistema hexadecimal** o **base 16**: utiliza 16 símbolos, que son los 10 dígitos (del 0 al 9) más 6 letras (de la A a la F): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, 20, etc.

Se usan porque permiten convertir fácilmente un número binario en uno octal o hexadecimal, y así poder tratar más fácilmente una cifra binaria que sería demasiado grande

$1437_{10} = 1x10^3 + 4x10^2 + 3x10^1 + 7x10^0 = 1000 + 400 + 30 + 7 =$
1437_{10} (vemos que nos da el mismo número)

734_8 (octal) $= 7x8^2 + 3x8^1 + 4x8^0 = 7x64+3x8+4 = 476_{10}$

$1B3_{16} = 1x16^2 + 11(B)x16^1 + 3x16^0 = 256 + 176 + 3 = 435_{10}$

$11010_2 = 1x2^4 + 1x2^3 + 0x2^2 + 1x2^1 + 0x2^0 = 26_{10}$

Octal a binario y viceversa

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

3 bits

$245_{10} = 11110101_2 = 365_8$

hexadecimal a binario y viceversa

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

4 bits

$245_{10} = 11110101_2 = F5_{16}$

Equivalencias

Decimal	Binario	Octal	Hexadecimal
0	000	0	0
1	001	1	1
2	010	2	2
3	011	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14

Signo Magnitud

Según este método, si se utilizan n bits para representar un número, se reserva un bit (normalmente el de mayor peso) para indicar el signo, y el resto de bits se utilizan para representar la magnitud. El convenio, un acuerdo arbitrario, dice que se utiliza la siguiente codificación para un número entero:

- Si el número es **positivo** su bit de mayor peso será 0:

0	*Magnitud* (n-1 bits)

- Si el número es **negativo** su bit de mayor peso será 1:

1	*Magnitud* (n-1 bits)

El bit de mayor peso que indica el signo del número recibe el nombre de **bit de signo**.

Ejemplo: utilizando 8 bits (n = 8), representa los números +25 y –25, siguiendo el convenio de signo y magnitud.

En primer lugar se convierte la magnitud o valor absoluto, 25, a binario natural con n-1=7 bits, completando con ceros los bits de mayor peso si fuera necesario:

$$25_{10} = 0011001_2$$

En segundo lugar se añade el bit de signo:

$$+25_{10} = \mathbf{0}0011001_2$$

$$-25_{10} = \mathbf{1}0011001_2$$

Ejemplo: obtén el valor decimal correspondiente a 011010_2 y 100010_2 sabiendo que están representados en signo y magnitud utilizando 6 bits (n = 6).

$011010_2 \longrightarrow$ dado que su bit de mayor peso (bit de signo) es 0, sabemos que se trata de un número positivo con magnitud $11010_2 = 26_{10}$ por lo que $011010_2 = +26_{10}$

$100010_2 \longrightarrow$ dado que su bit de mayor peso (bit de signo) es 1, sabemos que se trata de un número negativo con magnitud $00010_2 = 2_{10}$ por lo que $100010_2 = -2_{10}$

Doble representación del 0. (+0 y -0)

Complemento a 1

El sistema de numeración binaria de complemento a uno se caracteriza por el complemento bit de cualquier valor entero que es el negativo aritmético del valor. Es decir, invertir todos los bits de un número (el complemento lógico) produce el mismo resultado que restar el valor de 0.

Complemento a uno	Decimal
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	−0
1110	−1
1101	-2
1100	−3
1011	−4
1010	−5
1001	−6
1000	−7

Complemento a uno con enteros de 4 bits

Doble representación del 0. (+0 y -0)

Complemento a 2

1 0 1 1 0 1 ← NUMERO BINARIO ORIGINAL

↓ ↓ ↓ ↓ ↓ ↓

0 1 0 0 1 0 ← NUMERO BINARIO EN COMPLEMENTO A 1

+ 1

0 1 0 0 1 1 ← NUMERO BINARIO EN COMPLEMENTO A 2

Suma binaria
0 + 0 = 0 0 + 1 = 1 1 + 0 = 1 1 + 1 = 0 y acarreo 1

Figura. Suma binaria.

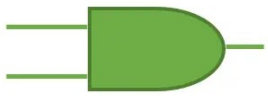
$$\begin{array}{cccc} 1 & 1 & 1 & \\ & 1 & 1 & 1 \rightarrow 7_{10)} \\ + & 1 & 0 & 1 \rightarrow 5_{10)} \\ \hline 1 & 1 & 0 & 0 \rightarrow 12_{10} \end{array}$$

Número decimal	Signo y magnitud	Complemento a 1	Complemento a 2
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
0	0000	0000	0000
-0	1000	1111	No existe
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001
-8	no existe	No existe	1000

Puertas lógicas

AND

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1



Compuerta AND

OR

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

Compuerta OR

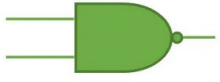
NOT

Q	Q´
0	1
1	0

Compuerta NOT

NAND

A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0



Compuerta NAND

NOR

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

Compuerta NOR

XOR

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

Compuerta XOR