

# Representación de la Información (2)

Alberto Ruiz Cristina

# Introducción

- Sabemos que el ordenador utiliza el sistema binario, pero ¿cómo se almacenan los números?
- Normalmente se asignan  $n$  bits para representar un número.
- Esos  $n$  bits son la longitud de una palabra, es decir, la información que puede procesarse de una vez

# Introducción

- El tamaño de palabra es muy importante en un equipo, ya que afecta...
  - A la representación de los números
  - Al tamaño idóneo de los registros
  - A la longitud de las direcciones de memoria
  - A la longitud de las instrucciones
- Hablamos de palabra, media palabra, doble palabra...
- Cuando los caracteres se almacenaban con 6 bits, los tamaños de palabra solían ser múltiplos de 6. Ahora lo son de 8 (16, 32, 64...)

# Representación de los números

- Vamos a estudiar algunos convenios de representación de números enteros y racionales en binario
- **Rango de representación:** intervalo comprendido entre el menor y el mayor número representable.
  - Con  $n$  bits podremos representar  $2^n$  combinaciones y por tanto números distintos.
  - Ejemplo: con 2 bits puedo representar  $2^2=4$  números distintos (00, 01, 10, 11)

# Coma fija sin signo

- Coma fija: el lugar donde se sitúa el punto de referencia es fijo. Para números enteros estará a la derecha del todo, y no se representa
- Esta representación coincide con el binario puro
- Ejemplo:  $27 = 00011011$
- En todos los ejemplos manejaremos tamaño de palabra 8
- Rango de representación:  $0, 2^n - 1$

# Coma fija con signo

- Lo llamamos “signo-magnitud”. El primer bit indica si el número que va a continuación es positivo (0) o negativo (1)

- Ejemplo:

Bit de Signo (BS)

↙

$$27 = 00011011$$
$$-27 = 10011011$$

- Perdemos un bit para representar números, así que el rango será:

$$-(2^{n-1}-1), 2^{n-1}-1$$

# Coma fija con signo

- Una peculiaridad:
  - 10000000
  - 00000000
- Tenemos dos representaciones para el 0: el +0 y el -0
- Esto complica los circuitos de detección de operaciones que dan 0 como resultado
- Por otra parte, necesitamos circuitos sumadores y restadores independientes

# Complemento a la base

- Permite transformar restas en sumas
- Expresamos un número por lo que le falta para llegar a la potencia superior
- Ejemplos en base 10:
  - $72 \rightarrow 100 - 72 = 28$
  - $28 \rightarrow 100 - 28 = 72$
  - $485 \rightarrow 1000 - 485 = 515$
- En base 2, hablamos de complemento a 2:
  - $101 \rightarrow 1000 - 101 = 011$
  - $011 \rightarrow 1000 - 11 = 101$
  - $1101 \rightarrow 10000 - 1101 = 0011$
- El complemento a la base de un número complementado, es el mismo número

# Complemento a 2


- Utilizaremos esta propiedad para convertir restas en sumas.
- $A - B = A + \text{Complemento}(B)$  despreciando el acarreo
- Ejemplo en base 10:
  - ¿72 - 43?
  - Complemento a 10 de 43  $\rightarrow 100 - 43 = 57$
  - $72 - 43 = 72 + (-43) = 72 + 57 = 129$
- Ejemplo en base 2:
  - ¿111 - 101?
  - Complemento a 2 de 100  $\rightarrow 1000 - 101 = 11$
  - $111 - 101 = 111 + 011 = 1010$

# Cálculo del Complemento a 2

- Método simplificado 1:

- Invierto todos los bits del número
- Sumo 1 al resultado
- $1001010 \longrightarrow 0110101$   
$$\begin{array}{r} + \quad \quad 1 \\ \hline 0110110 \end{array}$$

- Método simplificado 2:

- Trabajo el número de derecha a izquierda
- Mantengo todos los ceros que encuentre y el primer uno
- A partir de ahí y hasta llegar al final por la izquierda, invierto todos los bits
- $1001010$   
 $0110110$   


# Complemento a 2

- Los números que empiezan por 0 son positivos y se interpretan como binario puro
- Los números que empiezan por 1 son complementados y NO podemos conocer su valor directamente, hay que complementarlos
  - 11101 → Sé que es negativo pero NO sé cuánto vale
  - Complemento y queda 11, es decir, era el -3

# Complemento a 2

- Se adaptan a cualquier longitud de palabra, igual que en signo-magnitud:
  - $11111101 = 11101 = 101$
  - $00000011 = 00011 = 011$
- No tienen doble representación del 0
- Con un circuito sumador hacemos restas
- Caso especial: 100 complementado da... ¡100! Esto quiere decir que es el -4
- El rango del Ca2 es asimétrico:

$$-2^{n-1}, 2^{n-1}-1$$

# Complemento a la base menos 1

- Es el complemento a la base menos 1
- Ejemplo en base 10:
- El  $\text{Ca}_{10}$  era:
  - $72 \rightarrow 100 - 72 = 28$
  - $28 \rightarrow 100 - 28 = 72$
- El  $\text{Ca}_9$  es:
  - $72 \rightarrow 100 - 1 - 72 = 27$
  - $27 \rightarrow 100 - 1 - 27 = 72$

# Complemento a 1

- Ejemplo en base 2:
- El Ca2 era:
  - $101 \rightarrow 1000 - 101 = 11$
  - $011 \rightarrow 1000 - 011 = 101$
- El Ca1 es:
  - $101 \rightarrow 1000 - 1 - 101 = 10$
  - $010 \rightarrow 1000 - 1 - 010 = 101$
- En este caso hay doble representación del cero:
- $1000 = 0000 = 0$

# Complemento a 1

- En este caso, si hay acarreo, se suma al resultado:
- $A - B = A + \text{Complemento}(B) + \text{acarreo}$
- Ejemplo en base 10:
  - ¿72 - 43?
  - Complemento a 9 de 43  $\rightarrow 100 - 1 - 43 = 56$
  - $72 - 43 = 72 + (-43) = 72 + 56 = 128 \rightarrow 28 + 1 = 29$
- Ejemplo en base 2:
  - ¿111 - 101?
  - Complemento a 1 de 100  $\rightarrow 1000 - 1 - 101 = 10$
  - $111 - 101 = 111 + 010 = 1001 \rightarrow 001 + 1 = 10$

# Cálculo del Complemento a 1

- Método simplificado:

- Invierto todos los bits del número

- 1001010  0110101

- Ca1 frente a Ca2:

- En Ca1 hay doble representación del 0
- El Ca1 se calcula más fácilmente
- Al operar con Ca1 se tiene en cuenta el acarreo

- Rango del Ca1:





$$-2^{n-1}-1, 2^{n-1}-1$$

	Binario	S.Magn.	Ca1	Ca2
000	0	0	0	0
001	1	1	1	1
010	2	2	2	2
011	3	3	3	3
100	4	-0	-3	-4
101	5	-1	-2	-3
110	6	-2	-1	-2
111	7	-3	-0	-1





# Números reales

- La coma fija exige reservar un número determinado de dígitos para la parte entera y para la fraccionaria
- El resultado es desaprovechar espacio en ocasiones
- Además, el rango de valores es muy reducido
- A cambio, los circuitos son más simples

# Efecto de mover la coma

- ¿Qué le ocurre a un número al mover la coma?
- En base 10:
  - 12.25
  - 122.5 
  - 1.225 
- En base 2:
  - 1100.01 (12.25)
  - 11000.1 (24.5) 
  - 110.001 (6.125) 

# Efecto de mover la coma

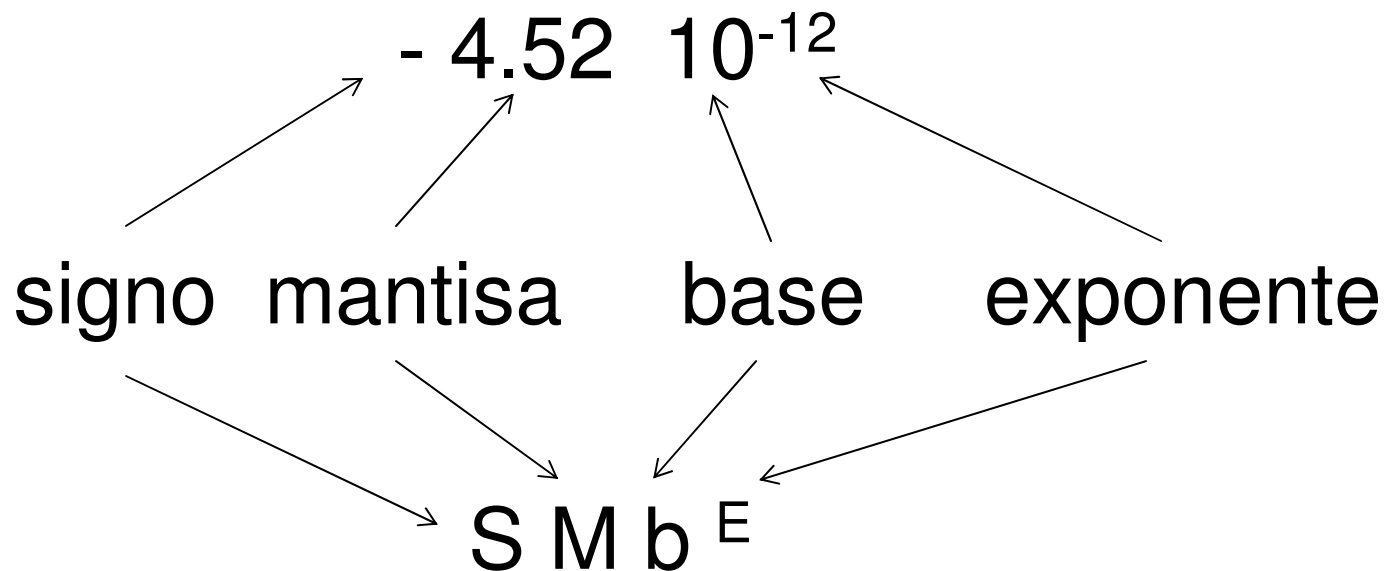
- ¿Qué le ocurre a un número al mover la coma?
- En base 10:
  - 12.25
  - 122.5  \*10
  - 1.225  /10
- En base 2:
  - 1100.01 (12.25)
  - 11000.1 (24.5)  \*2
  - 110.001 (6.125)  /2

# Coma flotante

- Se corresponde con lo que se llama “notación científica”
- Se usa para expresar números muy pequeños o muy grandes
- Ejemplo:
- $4\ 520\ 000\ 000\ 000 = 4.52 \cdot 10^{12}$
- $0,000\ 000\ 000\ 004\ 52 = 4.52 \cdot 10^{-12}$

# Coma flotante

- Componentes de la notación en coma flotante:



# Coma flotante

- Normalmente, un número se almacena como 2 números en coma fija:
  - Uno representa el signo del número y la mantisa
  - El otro representa el exponente
- La base no se representa, se da por sabida
- La **precisión** de los cálculos viene dada por el tamaño de la mantisa
- El **rango** de representación viene dado por el tamaño del exponente

# Normalización

- ¿Cómo se escribe el número 6.25 en binario usando notación en coma flotante?
  - $0110.01 \cdot 2^{000}$
  - $011.001 \cdot 2^{001}$
  - $01.1001 \cdot 2^{010}$
  - $0.11001 \cdot 2^{011}$
- Para evitar representaciones múltiples, se adopta el convenio de situar la coma en un lugar fijo.

# Normalización

- ¿Dónde ponemos la coma? A la izquierda del primer dígito significativo
- $0110.01 \ 2^{000} \rightarrow 0.11001 \ 2^{011}$
- Ventajas de coma flotante normalizada:
  - Dos números: 4132.524 y 0.03423
  - En coma fija, si reservamos 4 bits para la parte entera y 2 para la fraccionaria:
    - 4132.52 y 0000.03
  - En coma flotante normalizada:
    - $0.413252 \ 10^4$  y  $0.342300 \ 10^{-1}$

# Estándar IEEE 754

- IEEE: Institute of Electrical and Electronics Engineers
  - 1 bit de signo de la mantisa
  - 8 bits del exponente
  - 23 bits de mantisa



- Total: 32 bits.
- Hay versiones distintas del estándar de 64 bits, aumentando el tamaño de la mantisa

# IEEE 754

- La mantisa está expresada en signo-magnitud y normalizada: se asume que la coma está a la derecha del primer bit significativo (que valga 1)
- Es decir, el primer bit de la mantisa normalizada siempre será 1
- Por tanto no lo almacenamos, y que el circuito encargado de operar lo restablezca sobre la marcha

# IEEE 754

- Ejemplo:
  - Número:  $0110.01 \cdot 2^{000}$
  - Normalizamos:  $1.1001 \cdot 2^{010}$
  - ¿Para qué almacenar el primer 1 si siempre va a estar ahí?  $.1001 \cdot 2^{010}$
- Es decir, que todos los números expresados en IEEE 754 tienen la forma  $1.xxxx$ , pero el 1 no se representa

# IEEE 754

- Exponente: se almacena en exceso a 127
  - Con 8 bits mi rango es 0, 255
- No voy a almacenar el exponente real  $E$ , sino  $E+127$
- Para calcular el auténtico  $E$ , restaré 127 a lo almacenado
  - Por tanto el rango real es -127, 128
- Ventaja: el signo va implícito. Almacenar el exponente en positivo facilita algunas operaciones

# IEEE 754

- Ejemplo:  $-6.125_{(10)} = 110.001$
- Normalizamos:
  - $1.10001 \cdot 2^{010}$
  - El bit de signo de la mantisa será 1
  - El exponente será  $2+127 =$

$$\begin{array}{r} 1111111 \\ \hline 10 \\ \hline 1000001 \end{array}$$

1 1000001 100010000000000000000000

# IEEE 754

- $-118.625_{(10)} = 1110110.101$
- Movemos la coma:  $1.110110101 \cdot 2^6$
- $127 + 6 = 133 = 10000101$   
(¡quizá te sea más fácil sumar en binario!)

1 10000101 110110101000000000000000

- En general, la coma flotante complica los circuitos de operación, pero a cambio se ahorra en capacidad de almacenamiento

# IEEE 754

- ¿Y al revés?

1 1000001 100010000000000000000000

Signo: negativo

Mantisa: 1.10001

Exponente: 1000001 – 01111111 = 10

1.10001  $2^{010}$

Es decir, 110.001 = - 6.125

¡No olvidéis el signo!

# IEEE 754

- Quiero representar el 1  $\rightarrow 1.0 \cdot 2^0$
- El exponente será  $0 + 127 = 127$

```
0 01111111 00000000000000000000000000000000
```

- Quiero representar el 0
- La mantisa siempre será 1.0, no se puede representar. Por convenio, si todos los números están a 0  $\rightarrow$  Es el 0

```
0 00000000 00000000000000000000000000000000
```

# BCD

- Binary Coded Decimal, decimales codificados en binario
- Se usa para displays, calculadoras y aparatos en los que es más sencillo trabajar en decimal
- Necesitaré 4 bits para poder representar del 0 al 9, y sobrarán algunos valores
- No es eficiente ni para almacenar ni para calcular, pero en algunos casos interesa

# BCD

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9

Expresar 754 en BCD:

0111 0101 0100

¿Qué número es 01111001?

7 9

# Código ASCII

- **American Standard Code for Information Interchange**
- *Código Norteamericano Estándar para el Intercambio de Información*
- Creado en 1963
- Usa 7 bits (128 valores)
- El código ASCII extendido (8 bits) permite dos “mapas” de caracteres
  - Primer bit 0 → Los otros 7 son el ASCII estándar
  - Primer bit 1 → Los otros 7 son el ASCII extendido

# Código ASCII

1	␣	33	!	65	A	97	a	129	␣	161	ı	193	Á	225	á
2	␣	34	"	66	B	98	b	130	,	162	ϕ	194	Â	226	â
3	␣	35	#	67	C	99	c	131	f	163	£	195	Ã	227	ã
4	␣	36	\$	68	D	100	d	132	"	164	*	196	Ä	228	ä
5		37	%	69	E	101	e	133	...	165	¥	197	Å	229	å
6	-	38	&	70	F	102	f	134	†	166	ı	198	Æ	230	æ
7	•	39	'	71	G	103	g	135	‡	167	§	199	Ç	231	ç
8	█	40	(	72	H	104	h	136	^	168	˘	200	É	232	è
9		41	)	73	I	105	i	137	%o	169	©	201	É	233	é
10		42	*	74	J	106	j	138	Š	170	ª	202	Ê	234	ê
11	¿	43	+	75	K	107	k	139	<	171	«	203	Ë	235	ë
12	□	44	,	76	L	108	l	140	œ	172	¬	204	Ì	236	ì
13		45	-	77	M	109	m	141	␣	173	-	205	Í	237	í
14	␣	46	.	78	N	110	n	142	Ž	174	@	206	Î	238	î
15	⌘	47	/	79	O	111	o	143	␣	175	¯	207	Ï	239	ï
16	†	48	0	80	P	112	p	144	␣	176	°	208	Ð	240	ð
17	◀	49	1	81	Q	113	q	145	'	177	±	209	Ñ	241	ñ
18	↓	50	2	82	R	114	r	146	'	178	²	210	Ò	242	ò
19	!!	51	3	83	S	115	s	147	"	179	³	211	Ó	243	ó
20	¶	52	4	84	T	116	t	148	"	180	´	212	Ô	244	ô
21	±	53	5	85	U	117	u	149	•	181	µ	213	Õ	245	õ
22	␣	54	6	86	V	118	v	150	-	182	¶	214	Ö	246	ö
23	‡	55	7	87	W	119	w	151	—	183	·	215	×	247	×
24	↑	56	8	88	X	120	x	152	˘	184	¸	216	Ø	248	ø
25	‡	57	9	89	Y	121	y	153	™	185	˙	217	Ù	249	ù
26	→	58	:	90	Z	122	z	154	š	186	°	218	Ú	250	ú
27	←	59	;	91	[	123	{	155	>	187	»	219	Û	251	û
28		60	<	92	\	124		156	œ	188	¼	220	Ü	252	ü
29		61	=	93	]	125	}	157	␣	189	½	221	Ý	253	ý
30		62	>	94	^	126	~	158	ž	190	¾	222	Þ	254	þ
31		63	?	95	_	127	␣	159	ÿ	191	¿	223	ß	255	ÿ
32		64	@	96	`	128	€	160		192	À	224	à		

# En resumen...

- Lo que hay que saber:
  - Entender y expresar números enteros en binario puro, signo-magnitud, complemento a 2 y complemento a 1
  - Entender y expresar números reales en coma flotante (formato IEEE 754)
  - Entender y expresar números en BCD
  - Conocer los rangos de representación de todos estos sistemas de representación