Lección 1. *Representación de números*

1.1 Sistemas de numeración

Empecemos comentando cual es el significado de la notación decimal a la que estamos tan acostumbrados. Normalmente se escribe en notación abreviada: 821

Que en realidad significa: $8 \cdot 10^2 + 2 \cdot 10^1 + 1 \cdot 10^0 = 821$

En este caso hablamos que 10 es la base del sistema de números decimales. Evidentemente se ha escogido 10 pero podríamos haber escogido cualquier otro nº relativamente pequeño de símbolos. ¿Por qué el 10? $\Rightarrow$ 10 dedos!

1.2 Sistema binario

El sistema numérico con base 2 se conoce como binario y tiene como dos únicos elementos al 1 y al 0. ¿Cómo construimos los nº binarios? Análogamente al caso decimal, si tenemos en binario:

$$1011 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 11_{Dec}$$

¿Por qué serán importantes para nosotros los nº representados en sist. binario? Porque las máquinas trabajan mejor y son más sencillos de construir cuando sólo manejan información codificada en 1 y 0. El 1 y 0 lógicos en una máquina van a consistir en una tensión alta y una tensión baja respectivamente.

0 – 0.8 V $\Rightarrow$ "0" lógico

Lógica positiva 2.4 – 5 V $\Rightarrow$ "1" lógico

Lógica negativa es al revés.

Así, la representación de los 8 primeros números decimales en binario será:

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

Al 1 y al 0 se le denominan bits. Con 3 bits podemos representar hasta $2^3 = 8$ números $\Rightarrow$ 0 al 7.

1.3 Cambios de base

Veamos ahora como se pasa de un nº de base *n* a otro de la base *m*. Así, para pasar de decimal a binario:

		Resto
163	2	
81	2	1
40	2	1
20	2	0
10	2	0
5	2	0
2	2	1
1	2	0
0		1

$163_{10} = 10100011_2$ (comprobar)

de decimal a base 5

163	5	
32	5	3
6	5	2
1	5	1
0		1

$163_{10} = 1123_5$ (comprobar) = 125 + 25 + 10 + 3

1.4 Bases octal y hexadecimal

Base 8 $\Rightarrow$ octal

Base 16 $\Rightarrow$ hexadecimal

Son importantes por su íntima relación con la base 2

$8 = 2^3 \Rightarrow$ cada dígito octal son 3 binarios

$16 = 2^4 \Rightarrow$ cada dígito hexadecimal son 4 binarios

Para pasar de un nº binario a su correspondiente en octal se separan de derecha a izquierda en grupos de 3 bits y a cada grupo se le asigna su correspondiente en octal.

$653_{10} = 1\ 010\ 001\ 101_2 = 1215_8$

Análogamente se procede para pasar de binario a hexadecimal dividendo en grupos de 4 de derecha a izquierda pero en este caso se usan los diez nº decimales y las seis primeras letras del alfabeto.

Dec:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hex:	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Ej.: $653_{10} = 10\ 1000\ 1101 = 28D_{16}$

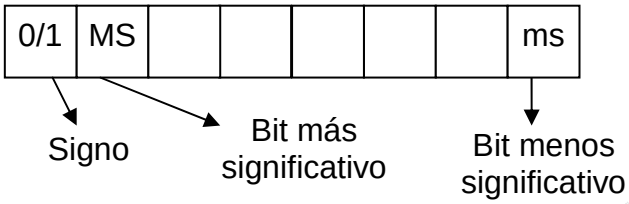
La base hexadecimal se usa mucho más que la octal.

1.5 Representación en magnitud y signo

Hasta ahora no hemos hablado en ningún momento del signo de los nº, sólo sabemos representar nº sin signo. Esto se soluciona fácilmente si se adopta el convenio de que el bit más a la izquierda de un conjunto de bits (8 bits = 1 byte) se corresponde con el signo del nº, es decir:

Signo + ≡ "0" Signo – ≡ "1"

Luego, por ejemplo en un byte podríamos representar:



Ej. 0 0000111 ⇒ +7

Si se desean representar nº mayores que 2^7 tendremos que usar 2 bytes = 16 bits = palabra (word). En este caso si tuviéramos dos registros:

donde ms corresponde al byte menos significativo, MS sería el byte más significativo y **s** es el bit de signo. En este caso, la casilla correspondiente al bit de signo del byte menos significativo queda inutilizado.

Podemos representar nº < 2^{14}. Cuando se emplea 4 bytes se habla de precisión simple, mientras que si usamos 8 bytes hablamos de doble precisión.

1.6 Representación en complemento a 1 (Ca1)

Una forma más útil de representar los nº negativos es mediante su complementario.

El Ca1 de un nº se define como: $C1(a) = 2^n - a - 1$

donde "n" es el nº de bits de los que se dispone para representar al mismo y "a" es el nº a complementar.

Veamos los 8 primeros nº binarios y sus complementarios:

0	0000
+1	0001
+2	0010

1111	0
1110	-1
1101	-2

+3	0011
+4	0100
+5	0101
+6	0110
+7	0111

1100	-3
1011	-4
1010	-5
1001	-6
1000	-7

Aparece el inconveniente de dos representaciones para el cero.

Ej.: Sea el 5 ¿C1? Con n=4 bits

C1(5 <<0101>>) = $2^4 - 5 - 1 = 16 - 6 = 10$ <<1010>>

Regla mnemotécnica: Poner los 1 a 0 y los 0 a 1.

1.7 Representación en complemento a 2

Para evitar la doble representación anterior para el 0 se define el C2 como:

C2(a) = $2^n - a$

0	0000
+1	0001
+2	0010
+3	0011
+4	0100
+5	0101
+6	0110
+7	0111

0000	0
1111	-1
1110	-2
1101	-3
1100	-4
1011	-5
1010	-6
1001	-7

Aparece una configuración libre "–8 ≡ 1000" y es igual a su complementario que no existe.

Ej.: 5 <<0101>> C2(5) = $2^4 - 5 = 11$ <<1011>>

Regla mnemotécnica: se barre de derecha a izquierda hasta encontrar el primer "1". Hasta ahí el número se deja igual incluido ese "1". A partir de él hacemos el C1 del resto del número.

1.8 Representación en punto fijo y punto flotante

1.8.1. Punto fijo

Hasta ahora no hemos comentado como se representan los nº fraccionarios en el sistema binario. Veamos esto con un ejemplo.

Ej.: $0{,}375_{10}$

Análogamente al caso de nº enteros donde se usaban potencias de 2 positivas ahora se usarán potencias de 2 negativas. Así:

2^{-1} 2^{-2} 2^{-3} 2^{-4} 2^{-5} 2^{-6} 2^{-7}

0,5 0,25 0,125 0,0625 0,03125 0,015625 0,0078125

En el caso que nos ocupa será:

$0{,}375_{10} = .011_2 = 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3}$

Regla práctica: En general, para obtener el valor decimal en cualquier base se multiplica el número por la base y nos quedamos con la parte entera, y la parte fraccionaria la seguimos multiplicando. Lo mismo sirve para el caso binario.

Ej.: $0{,}375_{10}$

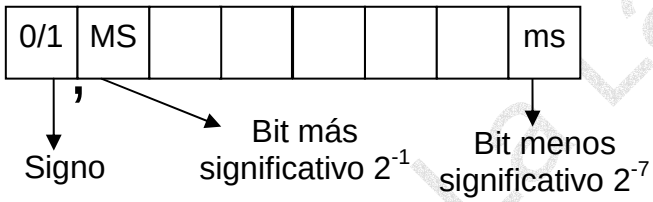
0,375 x 2 = **0**,75 < 1 ⇒ Pongo un "0"

0,75 x 2 = **1**,5 > 1 ⇒ Pongo un "1"

0,5 x 2 = **1** ⇒ Pongo un "1"

Es decir: .011

Si disponemos de 1 byte para representar un nº fraccionario (< 1).



1.8.2. Punto flotante (floating point)

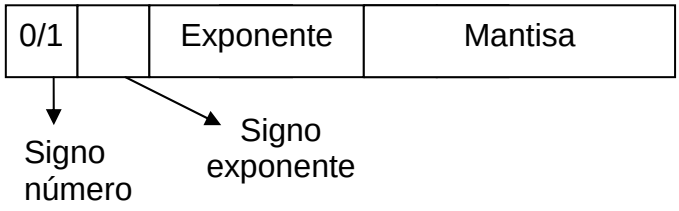
El punto flotante es una representación de los nº binarios análoga a la notación científica en decimal.

0,098 ⇒ 9.8E-2 Notación científica, donde "9.8" es la mantisa, "E" es la base (en este caso base 10) y "–2" el exponente con signo.

Los nº en punto flotante están dotados de exponente y parte fraccionaria o mantisa:

$$\pm m2^X$$

Para representarlos: la mantisa por convenio ha de estar entre $0{,}5 \leq m < 1$



Aparecen diferentes notaciones en punto flotante, una de ellos es la denominada EXCESO 64:

0/1	Exponente con signo (7 bits)	Mantisa (8 bits)

Donde se considera que el exponente 0 es el nº binario 64.

↑
|
|
65 exp +1 ⇒ 1000001
64 exp 0 ⇒ 1000000
63 exp -1 ⇒ 0111111
|
|
↓

De forma resumida emplearemos 2 bytes: 1 bit signo, 7 bits exp y 8 bits para la mantisa. En realidad se usan 4 bytes: 1 bit signo, 8 bits exp y 23 bits para la mantisa.

0/1	Exp. + signo (8 bits)	Mantisa (23 bits)

¿Cómo obtener ese exp y la mantisa a partir de un nº decimal cualquiera?

11 $11/2^4 = 0{,}687$

Ahora pasamos $0{,}687_{10}$ a base 2 ⇒ .10101111

Exponente 4 ⇒ en exceso 64: 1000100

Signo + ⇒ 0

Con lo que finalmente queda: 0 1000100 10101111

Ej.: Pasar a punto flotante los nº decimales 27, 0,8 y 0,1

27 $27/2^5 = 0.84375$

0.84375 ⇒ 11011000
exponente 5: 1000101 0 1000101 11011000
signo +: 0

0.8 $0.8/2^0 = 0.8$

0.8 ⇒ 11001100
exponente 0: ⇒ 1000000 0 1000000 11001100
signo +: 0

0.1 $0.1/2^{-3} = 0.8$

0.8 ⇒ 11001100
exponente -3: ⇒ 0111101 0 0111101 11001100
signo +: 0

Realizar los siguientes ejercicios:

1º. 0.1_8 a base 7

$0.1_8 = 0 \cdot 8^0 + 1 \cdot 8^{-1} = 0.125_{10} = 0.06060606_7$

0.125 x 7 = 0.875 ⇒ 0
0.875 x 7 = 6.125 ⇒ 6
0.125 x 7 = 0.875 ⇒ 0
… … … …

2º. 0.1_8 a base 2

$0.1_8 = 0.125_{10}$ = de forma directa = 000.001_2

0.125 x 2 = 0.250 ⇒ 0
0.250 x 2 = 0.500 ⇒ 0
0.500 x 2 = 1.000 ⇒ 1

3º 0.5_8 a base 2

$0.5_8 = 0.625_{10} = 000.101_2$

0.625 x 2 = 1.25 ⇒ 1
0.250 x 2 = 0.50 ⇒ 0
0.500 x 2 = 1.00 ⇒ 1

4º 0.56_8 a base 2

$0.56_8 = 0.71875_{10} = 0.101110$

0.71875 x 2 = 1.4375 ⇒ 1
0.43750 x 2 = 0.8750 ⇒ 0
0.87500 x 2 = 1.7500 ⇒ 1
0.75000 x 2 = 1.5000 ⇒ 1
0.50000 x 2 = 1.0000 ⇒ 1

5º $13.5A_{16} = 19.3323012_{10}$ = directo = $00010011.01011010_2 = 103.1122_4 = 23.264_8$

si se hiciera de la otra forma pueden aparecer incluso errores de redondeo.

6º 6.375_{10} a base 5

6	5	
1	5	1
0		1

0.375 x 5 = 1.875 ⇒ 1
0.875 x 5 = 4.375 ⇒ 4
0.375 x 5 = 1.875 ⇒ 1
… … … …

$6.375_{10} = 11.14141\ldots_5$

7º 13.652_7 a base 2

$13.652_7 = 1\cdot7^1 + 3\cdot7^0 + 6\cdot7^{-1} + 5\cdot7^{-2} + 2\cdot7^{-3} = 10.965014_{10}$

10	2	
5	2	0
2	2	1
1	2	0
0		1

0.965014 x 2 = 1,930028 ⇒ 1
0,930028 x 2 = 1,860056 ⇒ 1
0,860056 x 2 = 1,720112 ⇒ 1
0,720112 x 2 = 1,440224 ⇒ 1
0,440224 x 2 = 0,880448 ⇒ 0
0,880448 x 2 = 1,760896 ⇒ 1
0,760896 x 2 = 1,521792 ⇒ 1
0,521792 x 2 = 1,043584 ⇒ 1

$13.652_7 = 101.11110111$

8º -13.16_{10} a punto flotante

$13.16/2^4 = 0.8225$
$0.8225 \Rightarrow 11010010$
exponente +4: $\Rightarrow$ 1000100 1 1000100 11010010
signo –: 1