Sistemas Digitales
Sistemas de Numeración y Aritmética Binaria

Alfonso Sánchez-Macián Pérez

Grado en Ingeniería Informática

UNIVERSIDAD NEBRIJA

ÍNDICE

Sistema de numeración decimal

- 0-9 representan una cantidad.
- El dígito más a la derecha representa las unidades (peso 1)
- Más dígitos si queremos una cantidad superior, decenas (peso 10), centenas (peso 100). En general, peso 10^i

$$75438 = 7x10000 + 5x1000 + 4x100 + 3x10 + 8x1$$
$$= 7x10^4 + 5x10^3 + 4x10^2 + 3x10^1 + 8x10^0$$

$$0{,}3564 = 3 \times 0{,}1 + 5 \times 0{,}01 + 6 \times 0{,}001 + 4 \times 0{,}0001$$
$$= 3 \times 10^{-1} + 5x10^{-2} + 6x10^{-3} + 4 \times 10^{-4}$$

Sistema de numeración binario

- 0-1 representan una cantidad.

- Los pesos en este caso son potencia de 2 (base 2). En general, peso 2^i

$$1100 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$$

¿Equivalente decimal?

$$= 1 \times 8 + 1 \times 4 + 0 \times 2 + 0 \times 1 = 12$$

TABLE 2–2

Binary weights.

Positive Powers of Two (Whole Numbers)									Negative Powers of Two (Fractional Number)					
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
256	128	64	32	16	8	4	2	1	1/2 0.5	1/4 0.25	1/8 0.125	1/16 0.0625	1/32 0.03125	1/64 0.015625

Contar en binario

- En decimal, cuando llegamos al valor máximo del dígito (9) pasamos a 0 y sumamos 1 al siguiente dígito (19→20, 99→100).
- En binario, cuando llegamos al valor máximo del dígito (1) pasamos a 0 y sumamos 1 al siguiente dígito (01→10, 011→100).

TABLE 2–1

Decimal Number	Binary Number			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Máximo número decimal

1 bit: máximo	1	$= 2^0$		$= 1$
2 bits: máximo	11	$= 2^1+2^0$	$= 2+1$	$= 3$
3 bits: máximo	111	$= 2^2+2^1+2^0$	$= 4+2+1$	$= 7$
4 bits: máximo	1111	$= 2^3+2^2+2^1+2^0$	$= 8+4+2+1$	$= 15$

En general, para ***n*** bits ➔ máximo número decimal = $2^n - 1$

¿Cuál es el máximo número decimal que se puede representar con 6 bits?

¿Cuál es el mínimo número de bits que necesito para representar 126?

Conceptos de bit menos significativo (LSB) y más significativo (MSB)

Conversión decimal a binario: Método de la suma de pesos

- Reconocer las potencias de 2 (1, 2, 4, 8, 16, 32, 64, 128, 256 …)
- Separar el número decimal a convertir en suma de dichas potencias.

Algoritmo:

1. Empiezo por la inmediatamente inferior al número (ej: 134→128, 126 →64)
2. Resto esa potencia y tomo el resultado (ej: 134-128 = 6)
3. Repito 1 y 2 con el resultado hasta que obtenga el número.
4. Paso los números a potencias de 2. (ej: 128 = 2^7)
5. Pongo un 1 en esas posiciones y un 0 en el resto hasta la posición más baja (0 si entero o el correspondiente fraccionario).

$2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

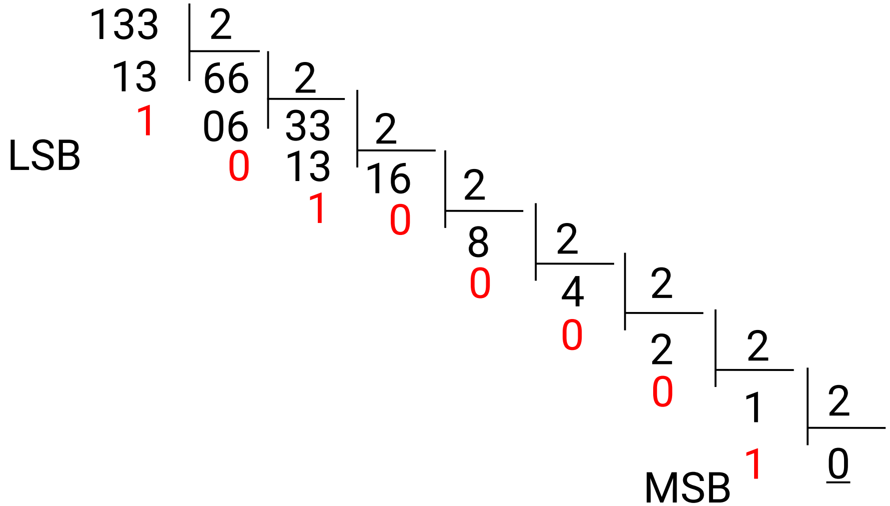
- 134 = 128 + 6 = 128 + 4 + 2 = 128 + 4 + 2 = $2^7 + 2^2 + 2^1$ ➔10000110
- 126 = 64 + 62 = 64 + 32 + 30 = 64 + 32 + 16 + 14 = 64 + 32 + 16 + 8 + 6 = 64 + 32 + 16 + 8 + 4 + 2 = $2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1$ ➔ 1111110

$2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

Conversión decimal a binario: Método de división sucesiva por 2

Para números enteros.

1. Se divide el número por 2 (ej: 133/2).
2. Se repite el punto 1 con el cociente de la división anterior hasta que el cociente sea 0.
3. Los restos conforman el número binario. El primer resto es el bit menos significativo (LSB) y el último el bit más significativo



10000101

Ejemplos

Convertir a binario los siguientes números enteros decimales usando los dos métodos vistos:

- 12
- 45
- 58
- 82

Conversión decimal a binario: Fracciones

Suma de pesos:

- Reconocer las potencias negativas de 2 (0,5 0,25 0,125, 0,0625…)
- Separar el número decimal a convertir en suma de dichas potencias.

Ej: 0,625 = 0,5 + 0,125 = $2^{-1} + 2^{-3}$ → 0,101

Multiplicación sucesiva por 2:

- Se multiplica el número por 2. La parte entera conforma el número binario.
- Se repite el proceso con la parte fraccionaria hasta que esta sea 0.

Ej: 0,625 x 2 = 1,25 → 1 (MSB)
0,25 x 2 = 0,5 → 0
0,5 x 2 = 1,00 → 1 (LSB)

Resultado: 0,101

Ejemplos

Convertir a binario los siguientes números decimales fraccionarios usando los dos métodos vistos:

- 0,125
- 0,375

Usa el método de la suma de pesos para convertir el siguiente número a binario

- 45,5

ÍNDICE

Suma binaria

Similar a la suma decimal. Acarreo = “me llevo una”

0 + 0 = 0 (acarreo 0)
0 + 1 = 1 (acarreo 0)
1 + 0 = 1 (acarreo 0)
1 + 1 = 0 (acarreo 1)

Decimal

```
acarreo → 1 1
           74
          +48
         ----
          122
```

Binario

```
acarreo → 1 1
           11        3
          +01       +1
         ----       --
          100        4
```

Ejemplos

Sumar los siguientes números binarios:

- 11+ 11
- 100 + 10
- 111 + 11
- 110 + 100
- 1111 + 1100

Resta binaria

$$\begin{aligned} 0 - 0 &= 0 \\ 1 - 0 &= 1 \\ 1 - 1 &= 0 \\ 10 - 1 &= 1 \end{aligned}$$

Decimal

acarreo →

$$\begin{array}{r} 74 \\ -\ {}^{1}8 \\ \hline 66 \end{array}$$

Binario

acarreo →

$$\begin{array}{r} 10 \\ -\ {}^{1}1 \\ \hline 01 \end{array} \qquad \begin{array}{r} 2 \\ -\ 1 \\ \hline 1 \end{array}$$

Ejemplos

Resta los siguientes números binarios:

- 11 – 01
- 11 – 10
- 111 – 100

Multiplicación binaria

Igual que la decimal.

- Primero se generan los productos parciales desplazando cada uno 1 posición a la izquierda.
- Esos productos parciales son binarios y se suman como tales.

0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1

```
   111
 x 101
------
   111
  000
+111
------
100011
```

División binaria

Igual que la decimal.

$$\begin{array}{r|l} 110 & 11 \\ \hline 00 & 10 \end{array}$$

ÍNDICE

- Sistemas de Numeración.
- Aritmética binaria sin signo.
- Aritmética con signo.
- Otras codificaciones.
- Detección de errores.

Complemento a 1 y complemento a 2

- Para representar números negativos.
  - Permite ver rápidamente positivo y negativo.
  - Facilita la aritmética con signo.

100011

- Complemento a 1 → Invertir todos los bits.

011100

- Complemento a 2 → Sumar 1 al complemento a 1.

011101

Números con signo. Bit de signo y Signo-magnitud

Uso de bit más significativo como bit de signo:

- 0 = positivo, 1= negativo

Formato Signo-Magnitud. Al quitar el bit de signo, el resto representa la magnitud.

011011   +27
111011   -27

Números con signo. Formato Complemento a 1

- Números positivos como signo-magnitud.
  011011 → +27

- Números negativos como complemento a 1 del equivalente positivo.

¿Cómo representar <u>-27</u>?

1. Tomamos +27 → 011011
2. Hallamos su complemento a 1 → <u>100100</u>

Los números negativos siguen teniendo el primer bit a 1

Números con signo. Formato Complemento a 2

- Números positivos como signo-magnitud.
  011011 → +27

- Números negativos como complemento a 2 del equivalente positivo.

¿Cómo representar <u>-27</u>?

1. Tomamos +27 → 011011
2. Hallamos su complemento a 1 → 100100
3. Sumamos 1 para el complemento a 2 → 100101

Los números negativos siguen teniendo el primer bit a 1

Conversión a decimal.

- Números en formato signo-magnitud.
  - Como suma de pesos y anteponiéndole el signo.

- Números en formato complemento a 1. Similar, pero:
  - El peso del bit de signo se resta.
  - Se suma 1 cuando es negativo.

Ejemplos:

01010100 ➔ $0x(-2^7)+1x2^6+0x2^5+1x2^4+0x2^3+1x2^2+0x2^1+0x2^0$

$= -0 + 64 + 0 + 16 + 0 + 4 + 0 + 0 =$ *84*

$-2^7\,2^6\,2^5\,2^4\,2^3\,2^2\,2^1\,2^0$

10101011 ➔ $1x(-2^7)+0x2^6+1x2^5+0x2^4+1x2^3+0x2^2+1x2^1+1x2^0$

$= -128 + 0 + 32 + 0 + 8 + 0 + 2 + 1 = -85$

-85+1 = *-84*

Conversión a decimal.

- Números en formato complemento a 2. Similar a signo-magnitud, pero:
  - El peso del bit de signo se resta.

Ejemplos:

01010100 ➔ $0x(-2^7)+1x2^6+0x2^5+1x2^4+0x2^3+1x2^2+0x2^1+0x2^0$

$-2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$ = -0 + 64 + 0 + 16+ 0 + 4 + 0 + 0 = *84*

10101100 ➔ $1x(-2^7)+0x2^6+1x2^5+0x2^4+1x2^3+1x2^2+0x2^1+0x2^0$

= -128 + 0 + 32 + 0 + 8 + 4 + 0 + 0 = - 84

Rango de representación. Formato Complemento a 2

- Números en formato complemento a 2. Con n bits se pueden representar los números desde $-(2^{n-1})$ hasta $+(2^{n-1}-1)$

- Por ejemplo, con 4 bits se puede representar desde $-(2^{4-1})= -8$ hasta $+(2^{4-1}-1)=+7$

Positivos: 0000 (+0)→ 0111 (+7)
Negativos: 1111(-1) →1000 (-8)

Suma de números binarios con signo

- Complemento a 2: sumar los dos números y descartar el acarreo final si existe.

```
   00000111 (+7)        00001111 (+15)      00010000 (+16)       11111011 (-5)
   00000100 (+4)        11111010 (-6)       11101000 (-24)       11110111 (-9)
 ________________    __________________   __________________   __________________
   00001011 (+11)     1 00001001 (+9)       11111000 (-8)      1 11110010 (-14)
```

- Cuidado con el desbordamiento ➔ cambio de signo.

```
   01111101 (+125)
   00111010 (+58)
 ________________
   10110111 (+183)   Error: -73
```

Resta de números binarios con signo

- Cambiar de signo al sustraendo y sumarlo al minuendo.

- Ejemplo: 16-24 = 16 + (-24)
  - 00010000 [+16]
  - 00011000 [+24] ➔ Complemento a 2 = 11100111+1= 11101000

00010000	(+16)
11101000	(-24)
11111000	(-8)

Multiplicación binaria con signo

1. Identificar los signos de los operandos. Si son iguales, el resultado será positivo. Si son diferentes será negativo.
2. Convertimos los números negativos en su valor real haciendo el complemento a 2.
3. Realizamos el producto como hicimos en la multiplicación sin signo.
4. Si en 1) se ha visto que el resultado será negativo, se halla el complemento a 2 del producto.

Ejemplo: -7x5 = -35
1001 x 0101
1) El resultado será negativo.
2) 1001 → 0111
   0101 → 0101

3) ➡

```
   0111
  x0101
  -----
   0111
  0000
 0111
0000
-------
0100011
```

4) ➡ 1011101

División binaria con signo. Restas sucesivas

¿21/7?

21-7 = 14 -7 = 7 -7 = 0

Hemos restados 3 veces 7 de 21 antes de obtener el resto 0.

Los restos parciales han sido 14 y 7.

División binaria con signo

1. Identificar los signos de los operandos. Si son iguales, el resultado será positivo. Si son diferentes será negativo.
2. Inicializamos el cociente a 0.
3. Convertimos los números negativos en su valor real haciendo el complemento a 2.
4. Realizamos la división restando del dividendo (o el resto parcial) el divisor mediante la suma con complemento a 2 y sumamos 1 al cociente.
    1. Si el resto parcial es positivo, repetir el paso 4 con el resto parcial.
    2. Si es cero o negativo, se acabó la división.
5. Si en 1) se ha visto que el resultado será negativo, se halla el complemento a 2 del producto.

División binaria con signo. Ejemplo

01100100 / 00011001

1. Ambos son positivos ➔ Resultado positivo
2. Cociente = 00000000
3. Los números son positivos, no hace falta convertir
4. Restamos

```
  01100100
+ 11100111 ← Complemento a 2 de 00011001
  01001011  ──────────→  Cociente = 00000001
+ 11100111
  00110010  ──────────→  Cociente = 00000010
+ 11100111
  00011001  ──────────→  Cociente = 00000011
+ 11100111
  00000000  ──────────→  Cociente = 00000100
```

ÍNDICE

- Sistemas de Numeración.
- Aritmética binaria sin signo.
- Aritmética con signo.
- Otras codificaciones.
- Detección de errores.

Sistema de numeración hexadecimal

- 16 caracteres. 0-F representan una cantidad.
- Cada dígito hexadecimal se corresponde con un número binario de 4 bits.

TABLE 2–3

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

0.0

Sistema de numeración hexadecimal

- Los pesos en este caso son potencia de 16 (base 16). En general, peso 16^i

$$F1A = F \times 16^2 + 1 \times 16^1 + A \times 16^0 = 15 \times 16^2 + 1 \times 16^1 + 10 \times 16^0$$

¿Equivalente decimal?

$$= 15 \times 256 + 1 \times 16 + 10 \times 1 = 3840 + 16 + 10 = 3866$$

- Contar: 0, 1, …, 9, A, B, C, D, E, F, 01, 02, 03 … 0F, 10 … 1F, ………FF, 100…

0.0

Conversión entre hexadecimal y binario (positivos)

- De hexadecimal a binario. Truco: tomar cada dígito y transformarlo a binario.

Ejemplo:

	hexadecimal	decimal (solo como ayuda visual)	binario
10A4 →	1 0 A 4 →	1 0 10 4 →	0001 0000 1010 0100
CF8E →	C F 8 E →	12 15 8 14 →	1100 1111 1000 1110

- De binario a hexadecimal. Truco: tomar cada 4 dígitos (empezando por el final) y transformarlos a hexadecimal.

Ejemplo:

	binario	decimal (solo como ayuda visual)	hexadecimal
111111000101101001 →	11 1111 0001 0110 1001 →	3 15 1 6 9 →	3F169

Sistema de numeración octal

- 8 caracteres. 0-7 representan una cantidad. Cada dígito corresponde a 3 binarios

TABLE 2–4

Octal/binary conversion.

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

- Los pesos en este caso son potencia de 8 (base 8). En general, peso 8^i

$$174 = 1 \times 8^2 + 7 \times 8^1 + 4 \times 8^0$$

¿Equivalente decimal?

$$= 1 \times 64 + 7 \times 8 + 4 \times 1 = 124$$

- Contar: 0, 1, …, 7, 10, 11, … 17, 20 … 27, ……….77, 100…

0.0

Conversión entre octal y binario (positivos)

- De octal a binario. Truco: tomar cada dígito y transformarlo a binario.

Ejemplo:

octal binario

107 ➔ 1 0 7 ➔ 001 000 111
733 ➔ 7 3 3 ➔ 111 011 011

- De binario a octal. Truco: tomar cada 3 dígitos (empezando por el final) y transformarlos a octal.

Ejemplo:
111111000101101001 ➔111 111 000 101 101 001 ➔ 7 7 0 5 5 1

binario octal

0.0

BCD - Decimal codificado en binario

- En vez de transformar todo el número a binario, cada dígito decimal se transforma en su equivalente binario.

- Números de 0 a 9 ➔ necesitamos 4 bits (0000 → 1001).

TABLE 2–5

Decimal/BCD conversion.

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

- Ejemplo: 170 ➔ 0001   0111 0000 ➔ 000101110000

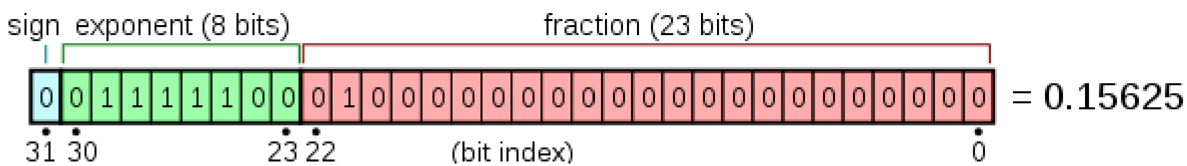
- Los valores 1010, 1011, 1100, 1101, 1110 y 1111 no son válidos en esta representación.

Números en coma flotante (floating point)

- Para representar a la vez números muy grandes y muy pequeños, incluidos fraccionarios.

- Estándar IEEE 754. Precisión simple (32 bits), precisión doble (64 bits), precisión ampliada (80 bits).

- Ejemplo precisión simple: Divide los bits en signo (1 bit), mantisa o fracción (23 bits) y exponente (8 bits).

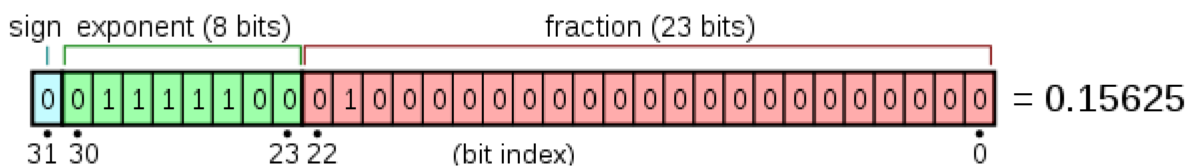


Números en coma flotante (single precision)

- Los números 0, infinito, NaN (Not a Number) y números muy pequeños se codifican de forma especial.

- El resto siguen la fórmula siguiente:

$$(-1)^{sign} \times (1.fraction) \times 2^{exponent-127}$$



$$(-1)^0 \times (1.01000000000000000000000) \times 2^{124-127} = (1.01)_2 \times 2^{-3} =$$
$$= (1\times2^0+0\times2^{-1}+1\times2^{-2})\times2^{-3} =$$
$$= 1{,}25 * 0{,}125 = 0{,}15625$$

ÍNDICE

Paridad para detección de errores (paridad par)

- Se añade un bit adicional que es 0 si hay un número par de 1s en el número o 1 si hay un número impar de 1s

0000 ➔ 0000 **0**
0111 ➔ 0000 **1**
¿0101? ➔ 0101 _

- Si se produce un error en el dato, se detecta al recalcular la paridad porque no coincide.

0000➔ 00000 ➔ (Error) 0100**0** ➔ Paridad 0100 (1) no coincide con (0)

Bibliografía e imágenes

Floyd, T. L. (2016), Fundamentos de Sistemas Digitales, Pearson, 11ª Edición