Sistemas de numeración y codificación

Norberto Malpica
Área de Tecnología Electrónica

norberto.malpica@urjc.es

Contenido

1.1 Sistemas de numeración

Sistema de numeración: conjunto de reglas y signos para representar los números.

Un sistema de representación numérica es un sistema consistente en:

- un conjunto ordenado de símbolos (dígitos o cifras).
- un conjunto de reglas bien definidas para las operaciones aritméticas de suma, resta, multiplicación, división, etc.

Números : secuencia de dígitos que pueden tener parte entera y parte fraccionaria, ambas separadas por una coma.

$$(N)_r = [(\text{parte entera}) \text{ , } (\text{parte fraccionaria})]_r$$

Base (r) : nº en que se fundamenta el sistema de numeración. Especifica el nº de dígitos o cardinal de dicho conjunto ordenado.

1.1 Sistemas de numeración: Conceptos básicos

Sistema posicional : **cada dígito tiene un valor distinto dependiendo de su posición.**

Así, un número N en base r se representa de la siguiente manera:

$(N)_r = (a_{p-1}\ a_{p-2} \ldots a_1\ a_0\ ,\ a_{-1}\ a_{-2} \ldots a_{-q})_r$ ó

$N_{\lfloor r} = a_{p-1}\ a_{p-2} \ldots a_1\ a_0\ ,\ a_{-1}\ a_{-2} \ldots a_{-q\ \lfloor r}$ ó

$N = a_{p-1}\ a_{p-2} \ldots a_1\ a_0\ ,\ a_{-1}\ a_{-2} \ldots a_{-q}$ si se sobreentiende que está en base r

Donde:

- $\mathbf{a_i}$ son los dígitos,
- **p** es el número de dígitos enteros,
- **q** es el número de dígitos fraccionarios,
- $\mathbf{a_{p-1}}$ es el dígito más significativo,
- $\mathbf{a_{-q}}$ es el dígito menos significativo.

Al ser N un número en base r, sus dígitos deben situarse entre 0 y r-1, es decir:

$$0 \le a_i \le r-1, \forall i \ \text{con} \ -q \le i \le p-1$$

1.1 Sistemas de numeración: Conceptos básicos

Cada dígito del número es más significativo que el que se encuentra a su derecha, siendo el valor del número la suma acumulada de los productos de cada dígito por su peso:

$$N = \sum_{i=-q}^{p-1} a_i \cdot r^i$$

Ejemplo: $(N)_r = (a_{p-1}\ a_{n-2} \ldots a_1\ a_0\ , a_{-1}\ a_{-2} \ldots a_{-q})_r$

$$(1283)_{10} = (a_3 = 1\ \ a_2 = 2\ \ a_1 = 8\ \ a_0 = 3,\ \ a_{-i} = 0)_{10} =$$

$$= 1 \times 10^3 + 2 \times 10^2 + 8 \times 10^1 + 3 \times 10^0 = 1 \times 1000 + 2 \times 100 + 8 \times 10 + 3 \times 1$$

1.1 Sistemas de numeración

r=10	DECIMAL	0,1,2,3,4,5,6,7,8,9
r=2	BINARIO	0,1 BIT
r=8	OCTAL	0,1,2,3,4,5,6,7
r=16	HEXADECIMAL	0,1,2,3,4,5,6,7,8,9, A, B, C, D, E, F

1.1 Sistemas de numeración

Equivalencias entre los 17 primeros números de los sistemas decimal, binario, octal y hexadecimal:

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Binario	0	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111	10000
Octal	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17	20
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10

1.2 Conversión entre bases

Convertir de cualquier base a base 10: **Se evalúa directamente la expresión**

$$(1101{,}01)_2 = 1\times 2^3 + 1\times 2^2 + 0\times 2^1 + 1\times 2^0 + 0\times 2^{-1} + 1\times 2^{-2} = 13{,}25_{10}$$

$$(14)_{16} = 1\times 16^1 + 4\times 16^0 = 1\times 16 + 4\times 1 = 20_{10}$$

Convertir de base 10 a cualquier base (s):

- La parte entera se convierte mediante divisiones sucesivas entre $(s)_r$
- La parte fraccionaria se convierte mediante productos sucesivos por $(s)_r$

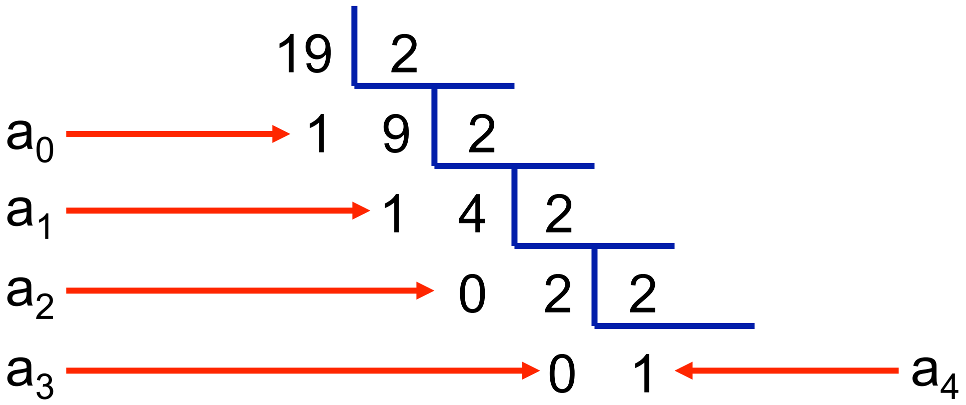
Bases de partida y de llegada son una potencia de la otra: ($s = r^k$ ó $r = s^k$)

- Binario a octal: se agrupan los dígitos binarios en grupos de tres
- Binario a hexadecimal: se agrupan los dígitos binarios en grupos de cuatro
- Octal a binario: Cada dígito octal se sustituye por su equivalente binario
- Hexadecimal a binario: Cada dígito hexadecimal se sustituye por su equivalente binario

1.2 Conversión entre bases: ejemplos

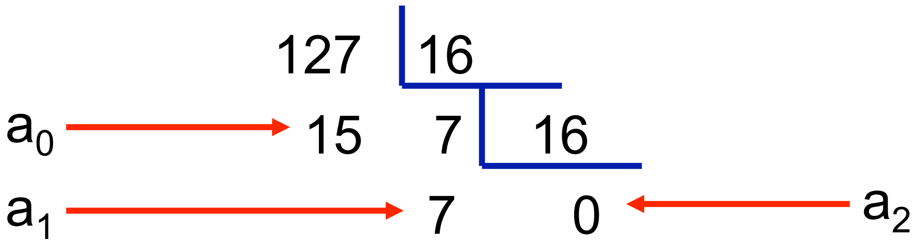
Convertir de base 10 a cualquier base (s)

Ejemplo: convertir el número $(19)_{10}$ a binario.



$(19)_{10} = (10011)_2$

Ejemplo: convertir el número $(127)_{10}$ a hexadecimal.



$(127)_{10} = (07(15))_{16} = (07F)_{16}$

1.2 Conversión entre bases: ejemplos

Convertir de base 10 a cualquier base (s)

Ejemplo: $(0{,}1285)_{10}$ a base 4.

		$0{,}1285 \times 4$
a_{-1} →	0	$0{,}5140 \times 4$
a_{-2} →	2	$0{,}0560 \times 4$
a_{-3} →	0	$0{,}2240 \times 4$
a_{-4} →	0	$0{,}8960 \times 4$
a_{-5} →	3	$0{,}5840 \times 4$
a_{-6} →	2	$0{,}3360 \times 4$
	...	...

$$(0{,}1285)_{10} = (0{,}020032\ldots)_4$$

Ejemplo: $(0{,}3)_{10}$ a binario.

		$0{,}3 \times 2$
a_{-1} →	0	$0{,}6 \times 2$
a_{-2} →	1	$0{,}2 \times 2$
a_{-3} →	0	$0{,}4 \times 2$
a_{-4} →	0	$0{,}8 \times 2$
a_{-5} →	1	$0{,}6 \times 2$
a_{-6} →	1	$0{,}2 \times 2$
a_{-7} →	0	$0{,}4 \times 2$
a_{-8} →	0	$0{,}8 \times 2$
a_{-9} →	1	$0{,}6 \times 2$
	...	...

$$(0{,}3)_{10} = (0{,}010011001\ldots)_2$$

1.2 Conversión entre bases: ejemplos

Bases de partida y de llegada son una potencia de la otra

- Binario a Octal:

$$(1101001{,}11101)_2 = (\underline{001}\ \underline{101}\ \underline{001},\ \underline{111}\ \underline{010})_2 = (151{,}72)_8$$
$$= (\ 1\quad 5\quad 1\ ,\ 7\quad 2)_8$$

- Binario a Hexadecimal:

$$(1111011{,}10101)_2 = (\underline{0111}\ \underline{1011},\ \underline{1010}\ \underline{1000})_2 = (7B{,}A8)_{16}$$
$$= (\ 7\quad B\ ,\ A\quad 8\ )_{16}$$

- Octal a binario:

$$(17{,}4)_8 = (\ \underline{1}\quad \underline{7}\ ,\ \underline{4}\ )_8$$
$$(\ 001\quad 111,\quad 100\ )_2 = (001111{,}100)_2$$

- Hexadecimal a binario:

$$(69{,}E)_{16} = (\ \underline{6}\quad \underline{9}\ ,\ \underline{E}\ )_{16}$$
$$(\ 0110\ 1001,\ 1110\ )_2 = (1101001{,}111)_2$$

2. Códigos binarios: conceptos básicos

Rango de un sistema de representación:

Es el intervalo comprendido entre el menor y el mayor número representable.

Por ejemplo, en binario puro con representación entera:

- con n bits hay 2^n elementos $\Rightarrow$ rango desde el 000...0 hasta el 111...1 $\Rightarrow$ desde 0 hasta 2^n-1 en enteros.
- con 2 bits $\Rightarrow$ desde 0 (00) hasta 3 (11) $\Rightarrow$ desde 0 hasta 2^2-1.

Resolución de un sistema de representación:

Es la diferencia existente entre dos elementos representables consecutivos.

Por ejemplo, en binario puro con representación entera:

- con n bits hay 2^n elementos $\Rightarrow$ resolución de 1 entero.
- con 2 bits $\Rightarrow$ {0,1,2,3} $\Rightarrow$ resolución de 1 entero.

2. Códigos binarios

Números positivos

Binario natural

Números negativos

Bit de signo + magnitud

S	MMMMMMM

Complemento a 1

Complemento a 2

Números reales

Estándares IEEE (IEEE 754)

2. Representación de números en coma fija

Sistemas de representación en **coma fija**:

- Sistema de representación sin signo: **binario puro**
- Sistemas de representación con signo:
  - **Magnitud y signo** (signo-magnitud o módulo y signo)
  - Complemento a la base (en binario: **complemento a 2**)
  - Complemento restringido a la base (en binario: **complemento a 1**)

2.1 Binario puro

1. **Definición:** divisiones y productos sucesivos
2. **Conversión a base 10**

$$A = \sum_{i=0}^{n-1} a_i \cdot 2^i$$

3. **Representación de números**
   - **Sólo nº positivos**
4. **Rango: [0, 2^n-1]**
5. **Cambio de signo:** representación de nº sin signo
6. **Extensión de signo:** representación de nº sin signo

2.2 Magnitud y signo

1. Definición: bit de **signo** y la **magnitud** del número.

S	MMMMMMM

2. Conversión a base 10

$$A = (1 - 2 \cdot a_{n-1}) \cdot \sum_{i=0}^{n-2} a_i \cdot 2^i$$

3. Representación de números

- **nº positivos:**

0	MMMMMMM

- **nº negativos:**

1	MMMMMMM

4. Rango: $[-(2^{n-1}-1), 2^{n-1}-1]$.

¡Ambigüedad en el cero: 0000 – 1000!

5. Cambio de signo: cambiar el bit de signo

6. Extensión de signo: se desplaza a la izquierda el bit de signo, y el hueco en el destino se rellena con bits a 0.

2.3 Complemento a 2

1. Definición: $C_2(N) = 2^n$ - N

Si n = 4, $C_2(1010)$ = 10000 - 1010 = 0110
Si n = 5, $C_2(10100) = 2^5$ - 10100 = 01100

$$\begin{array}{r} 100000 \\ -\quad 10100 \\ \hline 01100 \end{array}$$

Cálculo del C_2 se procede de derecha a izquierda de la siguiente manera:

- Copiar todos los bits de N hasta el primer 1 inclusive.
- El resto de los bits se obtienen cambiando 1s por 0s y 0s por 1s.

2. Conversión a base 10

$$A = -a_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} a_i \cdot 2^i$$

$A_{\lfloor C2} = 00011101_{\lfloor C2}$, $A = 1x2^0 + 1x2^2 + 1x2^3 + 1x2^4 = 29_{\lfloor 10}$

$B_{\lfloor C2} = 11001011_{\lfloor C2}$, $B = 1x2^0 + 1x2^1 + 1x2^3 + 1x2^6 - 1x2^7 = -53_{\lfloor 10}$

2.3 Complemento a 2

3. Representación de números

- **nº positivos:** Igual que en magnitud y signo

0	MMMMMMM

- **nº negativos: C_2 (N). Complemento a la base** del número positivo N.

Ejemplo: representar $A = 29_{\lfloor 10}$ con n = 8 bits

$A_{\lfloor C2} = A_{\lfloor MS} = 00011101_{\lfloor C2}$

Ejemplo: representar $B = -53\ _{\lfloor 10}$ con n = 8 bits

Primero lo representamos en positivo: $-B_{\lfloor C2} = 00110101_{\lfloor C2}$

Ahora calculamos el complemento: $B_{\lfloor C2} = C2(-B_{\lfloor C2}) = 11001011_{\lfloor C2}$

También podemos calcularlo directamente y después pasarlo a binario:

$B_{\lfloor C2} = 2^8\text{-}53 = 256\text{-}53 = 203 = 11001011_{\lfloor C2}$

2.3 Complemento a 2

4. Rango: $[-2^{n-1}, 2^{n-1}-1]$

Ejemplo: Para un número en base 2 (r=2) representado con 4 bits (n=4), el rango es $-2^3 \leq x \leq 2^3-1 \Rightarrow -8 \leq x \leq 7$.

0110 = 6	1110 = –2
0000 = 0	1000 = –8

5. Cambio de signo: se lleva a cabo mediante la complementación.

Ejemplo: cambiar de signo el número $A_{\lfloor C2} = 00011101_{\lfloor C2}$, n = 8

$-A_{\lfloor C2} = C2(A_{\lfloor C2}) = 11100011_{\lfloor C2}$

Ejemplo: cambiar de signo el número $B_{\lfloor C2} = 11001011_{\lfloor C2}$, n = 8, q = 0

$-B_{\lfloor C2} = C2(B_{\lfloor C2}) = 00110101_{\lfloor C2}$

2.3 Complemento a 2

6. Extensión de signo: se replica el bit de signo hacia la izquierda.

Ejemplo: extender X = $100110_{|C2}$ de 6 a 8 bits $\underline{1}00110$

11100110

Extender X = $010011_{|C2}$ de 6 a 8 bits $\underline{0}10011$

00010011

2.4 Complemento a 1

1. Definición: $C_1N = 2^n - 1 - N$

Si n = 4, $C_1(1010) = 2^4 - 1010 - 0001 = 10000 - 1010 - 0001 = 0101$

Cálculo del C_1 : complementar todos los bits del número, es decir, a cambiar 1s por 0s y 0s por 1s.

El cálculo del C_1 se puede considerar un paso intermedio para el cálculo del C_2. Así, para obtener el C_2 de un número podemos realizar estos dos pasos:

a) Calcular el C_1

b) Sumar un 1 al dígito menos significativo

2. Conversión a base 10

$$A = (-1)^{a_{n-1}} \sum_{i=0}^{n-2} (a_i \oplus a_{n-1}) \cdot 2^i$$

2.4 Complemento a 1

3. Representación de números

- **nº positivos:** Igual que en magnitud y signo

0	MMMMMMMM

- **nº negativos: C_1 (N). Complemento restringido a la base** del número positivo N.

4. Rango: $[-(2^{n-1}- 1), 2^{n-1}- 1]$

Ambigüedad en el cero: 0000 – 1111

5. Cambio de signo: se lleva a cabo mediante la complementación

6. Extensión de signo: se replica el bit de signo hacia la izquierda.

Resumen

	Binario Puro	Magnitud-Signo	C2	C1
Definición	Divisiones sucesivas	S | Magnitud	$C_2N=2^n-N$ De dcha a izqda: copiar hasta el primer 1 y luego 1s por 0s y 0s por 1s	$C_1N=2^n-N-1$ 1s por 0s y 0s por 1s
Nº Positivos		0 | Magnitud	0 | Magnitud	0 | Magnitud
Nº Negativos	Sin signo	1 | Magnitud	C_2N	C_1N
Rango	$[0, 2^n -1]$	$[-(2^{n-1} -1), 2^{n-1} -1]$	$[-2^{n-1} , 2^{n-1} -1]$	$[-(2^{n-1} -1), 2^{n-1} -1]$
Cambio de Signo	Sin signo	Cambiar bit de signo	C_2N	C_1N
Conversión a base 10	$A=\sum_{i=0}^{n-1} a_i \cdot 2^i$	$A=(1-2a_{n-1})\sum_{i=0}^{n-2} a_i \cdot 2^i$	$A=-a_{n-1}\cdot 2^{n-1}+\sum_{i=0}^{n-2} a_i \cdot 2^i$	$A=(-1)^{a_{n-1}}\sum_{i=0}^{n-2}(a_i \oplus a_{n-1})\cdot 2^i$
Extensión del signo	Sin signo	Se copian los bits de signo y magnitud y los que faltan ceros	Se replica el bit de signo	Se replica el bit de signo
0	Único: 0	0000 - 1000	Único:0000	0000 - 1111

3. Aritmética binaria en coma fija

Estudiaremos las reglas básicas de la aritmética según los distintos sistemas de representación numérica en coma fija estudiados:

- ➲ Binario puro
- ➲ Magnitud y signo
- ➲ Complemento a 2
- ➲ Complemento a 1

3. Aritmética en base 2

SUMA	Acarreo
0 + 0 = 0	0
0 + 1 = 1	0
1 + 0 = 1	0
1 + 1 = 0	1

RESTA	Acarreo
0 - 0 = 0	0
0 - 1 = 1	1
1 - 0 = 1	0
1 - 1 = 0	0

Suma binaria

```
  1     acarreos    1 1 1
 ----              --------
   9                1 0 0 1
+ 15             +  1 1 1 1
 ----              --------
  24              1 1 0 0 0
```

Resta binaria

```
   83        1 0 1 0 0 1 1   minuendo
-  21    -       1 0 1 0 1   sustraendo
 ----        -------------
   62        1 1 1 1         acarreos
             -------------
             0 1 1 1 1 1 0   diferencia
```

3. Aritmética en base 2

PRODUCTO BINARIO

$0 \times 0 = 0$
$0 \times 1 = 0$
$1 \times 0 = 0$
$1 \times 1 = 1$

$$
\begin{array}{rr}
 & 12 \\
\times & 6 \\
\hline
 & 72
\end{array}
$$

$$
\begin{array}{rccccccl}
 & & & & 1 & 1 & 0 & 0 \quad \text{multiplicando} \\
\times & & & & & 1 & 1 & 0 \quad \text{multiplicador} \\
\hline
 & & & & 0 & 0 & 0 & 0 \\
 & & & 1 & 1 & 0 & 0 & \quad \text{Productos parciales} \\
 & & 1 & 1 & 0 & 0 & & \\
\hline
 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \quad \text{resultado}
\end{array}
$$

3. Aritmética en base 2

División binaria

La división binaria se puede realizar igual que la decimal.

En el caso de la binaria es más sencillo porque **se simplifica la elección de cada dígito del cociente** ya que sólo pueden ser 0 ó 1.

➲ Si el dividendo parcial es mayor o igual que el divisor, el siguiente dígito del cociente es 1, si no es 0.

```
112 | 8
    |____
     14

dividendo  1 1 1 0 0 0 0 | 1 0 0 0   divisor
         - 1 0 0 0       |________
           ---------       1 1 1 0   cociente
           0 1 1 0 0
           - 1 0 0 0
             ---------
             0 1 0 0 0
             - 1 0 0 0
               ---------
               0 0 0 0 0
               - 0 0 0 0
                 ---------
                 0 0 0 0   resto
```

3. Aritmética en base 2

Multiplicación y división de un número N por una potencia de la base r (r^m):

$$N = a_{p-1} \cdot r^{p-1} + \ldots\ldots + a_0 \cdot r^0 + a_{-1} \cdot r^{-1} + \ldots\ldots + a_{-q} \cdot r^{-q}$$

$$N \cdot r^m = a_{n-1} \cdot r^{n-1+m} + \ldots\ldots + a_0 \cdot r^{0+m} + a_{-1} \cdot r^{-1+m} + \ldots\ldots + a_{-p} \cdot r^{-p+m}$$

La coma aparece a la derecha del dígito a_i que cumple $i+m = 0 \Rightarrow i = -m$, es decir detrás del dígito que originalmente era a_{-m}

- **si m > 0 (multiplicación)** se desplaza la coma p lugares **a la derecha.**
- **si m < 0 (división)** se desplaza la coma p lugares **a la izquierda.**

Ejemplo: $(1101001,111)_2 \times 2^3 = (1101001111,0)_2$

$(1101001,111)_2 \times 2^{-4} = (110,1001111)_2$

$(10,53)_{10} \times 10^4 = (105300,0)_{10}$

3. Aritmética binaria en binario puro

Sus reglas son las de la aritmética binaria ya estudiada, con la limitación del tamaño de los operandos (**n = p+q**).

Desbordamiento: puede darse al realizar sumas, restas, multiplicaciones y divisiones.

- **Suma**: el resultado puede tener n+1 bits (**acarreo superior C = 1**)

$$\begin{array}{r} 1101 \\ +\ 1111 \\ \hline \boxed{1}\,1100 \end{array} \qquad \begin{array}{r} 13 \\ +15 \\ \hline 28 \end{array}$$

acarreo

DESBORDAMIENTO POSITIVO

- **Resta**: el resultado puede ser negativo (**acarreo superior C = 1**)

$$\begin{array}{r} 1101 \\ -\ 1111 \\ \hline \boxed{1}\,1110 \end{array} \qquad \begin{array}{r} 13 \\ -\ 15 \\ \hline -\ 2 \end{array}$$

acarreo

DESBORDAMIENTO NEGATIVO: sustraendo mayor que minuendo

- **Producto**: al multiplicar números de n bits el resultado puede necesitar hasta 2n bits (¡puede salirse de rango!).
- **División**: hay desbordamiento si el divisor es 0.

3. Aritmética binaria en magnitud y signo

Es preciso tratar por separado signos y magnitudes.

Suma de R=A+B: casos posibles

- ➲ Signo(A) = Signo(B):
  - Signo (R) = signo(A) = signo(B)
  - |R| = |A| + |B|
- ➲ Signo (A) ≠Signo(B):
  - signo(R) = signo (mayor(|A|,|B|)
  - |R| =(mayor(|A|,|B|) - (menor(|A|,|B|)

Resta: A-B = A + (-B)

Por tanto, al sumar o restar con módulo y signo se debe hacer lo siguiente:

1. Observar los signos y decidir qué operación se va a realizar.
2. Ordenar los módulos si hay que restar.
3. Operar con los módulos y detectar el posible desbordamiento.
4. Colocar el signo al resultado.

3. Aritmética binaria en magnitud y signo

Producto:

1. Se separan el signo y el módulo del multiplicando y del multiplicador.
2. Se multiplican los módulos (da un resultado de hasta 2n-2 bits).
3. Si los signos del multiplicando y el multiplicador son iguales, el resultado es positivo, y si no es negativo.

División:

1. Se separan el signo y el módulo del dividendo y del divisor.
2. Se dividen los módulos.
3. Si los signos del dividendo y divisor son iguales, el cociente es positivo, y si no es negativo.
4. El signo del resto será siempre igual que el del dividendo.

Desbordamiento: se detecta al operar con los módulos.

3. Aritmética en complemento a 2

➲**El bit de acarreo superior siempre se desprecia,** se realiza la suma directamente con las reglas de la **aritmética binaria**

Ejemplo: A = 4, B = 2

0100	4
+ 0010	+ 2
0110	6

Ejemplo : A = 6, B = -4

0110	6
+ 1100	- 4
1 0010	2

Se desprecia el bit de acarreo

Ejemplo : A = 4, B = -6

0100	4
+ 1010	- 6
1110	- 2

Ejemplo : A = -2, B = -3

1110	-2
+ 1101	-3
1 1011	-5

Se desprecia el bit de acarreo

3. Aritmética en complemento a 2

Resta en complemento a 2: **$A-B = A + C_2(B)$**

Ejemplo: $A = 6_{\lfloor 10} = 0110_{\lfloor C2}$, $B = 4_{\lfloor 10} = 0100_{\lfloor C2}$, $A-B = 2_{\lfloor 10}$, $n = 4$

Primero: complementar el sustraendo $-B_{\lfloor C2} = C2(B_{\lfloor C2}) = 1100_{\lfloor C2}$

Segundo: sumar A+(-B)

El acarreo superior se desprecia,

y el resultado es positivo

$$\begin{array}{r} 0110 \\ +\ 1100 \\ \hline \boxed{1}0010 \end{array}$$

Ejemplo: $A = -7_{\lfloor 10} = 1001_{\lfloor C2}$, $B = -3_{\lfloor 10} = 1101_{\lfloor C2}$, $A-B = -4_{\lfloor 10}$, $n = 4$

Primero: complementar el sustraendo $-B_{\lfloor C2} = C2(B_{\lfloor C2}) = 0011_{\lfloor C2}$

Segundo: sumar A+(-B)

$$\begin{array}{r} 1001 \\ +\ 0011 \\ \hline 1100 \end{array}$$

3. Aritmética en complemento a 2

➲ **En sumas y restas en complemento a 2, el bit de acarreo superior siempre se desprecia, y el resultado obtenido siempre es correcto (salvo que se produzca desbordamiento).**

➲ **Desbordamiento en sumas y restas:** se detecta porque el resultado presenta un signo erróneo.

Puede producirse desbordamiento al sumar dos números de igual signo o al restar dos números de distinto signo.

Nunca puede haber desbordamiento al sumar números de distinto signo o al restar números de igual signo.

El posible acarreo superior resultante en una suma o una resta no indica desbordamiento.

También puede producirse desbordamiento en productos y divisiones.

3. Aritmética en complemento a 2

Ejemplos de sumas con desbordamiento

$A = 6_{\lfloor 10} = 0110_{\lfloor C2}$, $B = 3_{\lfloor 10} = 0011_{\lfloor C2}$, $A+B = 9_{\lfloor 10}$, $n = 4$, $q = 0$

$$\begin{array}{rr} 0110 & 6 \\ +\ 0011 & +\ 3 \\ \hline 1001 & ¡¡-7!! \end{array}$$

La suma de dos números positivos no puede producir un número negativo: V = 1

$A = -3_{\lfloor 10} = 1101_{\lfloor C2}$, $B = -7_{\lfloor 10} = 1001_{\lfloor C2}$, $A+B = -10_{\lfloor 10}$, $n = 4$, $q = 0$

$$\begin{array}{rr} 1101 & -3 \\ +\ 1001 & +\ \ -7 \\ \hline \boxed{1}0110 & ¡¡6!! \end{array}$$

La suma de dos números negativos no puede producir un número positivo: V = 1

El acarreo superior se desprecia, y el resultado es positivo

3. Aritmética en complemento a 2

Es posible utilizar la resta en complemento a 2 para realizar más fácilmente la resta en binario puro.

➲Precaución: hay que invertir el bit de acarreo superior una vez realizada la operación de resta en complemento a 2.

Ejemplo: $A=0111_{\lfloor 2}$, $B = 0011_{\lfloor 2}$, $A\text{-}B = 4_{\lfloor 10}$, $n = 4$, $q = 0$

Operación en binario puro

$$\begin{array}{r} 0111 \\ -\ 0011 \\ \hline 0100 \end{array}$$

Acarreo superior erróneo:
hay que invertirlo **C=0**

Operación en complemento a 2

$$\begin{array}{r} 0111 \\ +\ 1101 \\ \hline \boxed{\mathbf{1}}0100 \end{array}$$

3. Aritmética en complemento a 2

Aunque hay algoritmos para multiplicar y dividir directamente números en complemento a 2, no los vamos a estudiar todavía.

De momento, para multiplicar y para dividir haremos lo siguiente:

- Pasamos los operandos a positivos.
- Operamos en binario puro.
- Si el análisis de los signos de los operandos revela que el resultado (o el cociente o el resto) debe ser negativo, se complementa el dato obtenido.

De forma análoga a la aritmética estudiada para la representación en complemento a la base, se puede estudiar la aritmética para la representación en complemento restringido a la base.

- El acarreo superior siempre se desprecia.
- Problema: si en sumas o restas el bit de acarreo superior vale 1, es preciso sumar 1 al resultado.

3. Aritmética en complemento a 1

De forma análoga a la aritmética estudiada para la representación en complemento a 2, se puede estudiar la aritmética para la representación en complemento a 1.

- ➲ El acarreo superior siempre se desprecia.
- ➲ Si en sumas o restas el bit de acarreo superior vale 1, es preciso sumar 1 al resultado.

4. Códigos BCD (Binary-Coded Decimal)

Codificación en BCD:

Es la codificación decimal más sencilla y representa a los diez dígitos decimales asignándoles el código binario de su representación binaria pura con 4 bits. Con esa representación un número decimal se evalúa mediante la expresión:

$$b_3 \cdot 2^3 + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0 = b_3 \cdot \mathbf{8} + b_2 \cdot \mathbf{4} + b_1 \cdot \mathbf{2} + b_0 \cdot \mathbf{1}$$

Por esta razón al código BCD se le conoce también como **código 8-4-2-1.**

Equivalencia entre dígitos decimales y código BCD 8-4-2-1:

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

Es importante **no confundir la representación de un dígito decimal en BCD con un número binario**, ya que son representaciones distintas.

Ejemplo: En BCD, el número decimal de dos dígitos 56 se escribe (5) y (6), es decir 0101 0110, mientras que en binario puro se escribe como 111000.

4. Aritmética en BCD

La **suma** de dos dígitos representados en BCD proporciona un dígito correcto representado en BCD, a no ser que:

- El dígito resultante sea mayor que 9.
- Se produzca un acarreo superior.

Corrección: sumar 6 al dígito resultante y dar un acarreo superior a la siguiente pareja de dígitos BCD.

Ejemplos:

	0100		4
	+ 0011		+ 3
	0111		7

		1000	8
		+ 0011	+ 3
Incorrecto BCD		1011	
		+ 0110	+ 6
Correcto BCD	1	0001	11

		1001	9
		+ 1001	+ 9
Incorrecto BCD	1	0010	
		+ 0110	+ 6
Correcto BCD	1	1000	18

Resta en BCD es una suma haciendo el complemento a 10 del sustraendo.