

# Representación de la información numérica

## Contenidos:

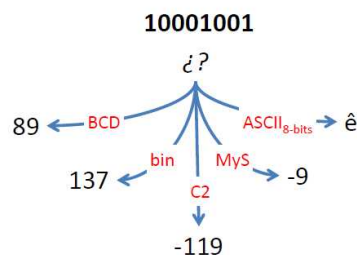
### Representación de información numérica

1. Sistemas de numeración
  1. Binario
  2. Hexadecimal
  3. Octal
2. Conversión entre sistemas de numeración
3. Números enteros con signo y sin signo: representación y operaciones
  1. Signo magnitud
  2. Complemento a 1
  3. Complemento a 2
4. Adición y resta binaria: acarreo y desbordamiento
5. Operaciones lógicas: AND, OR, XOR, NOT, NAND, NOR
6. Operaciones a nivel de bit: SHIFT-LEFT, SHIFT-RIGHT y versiones aritméticas
7. Números decimales: coma flotante IEEE 754

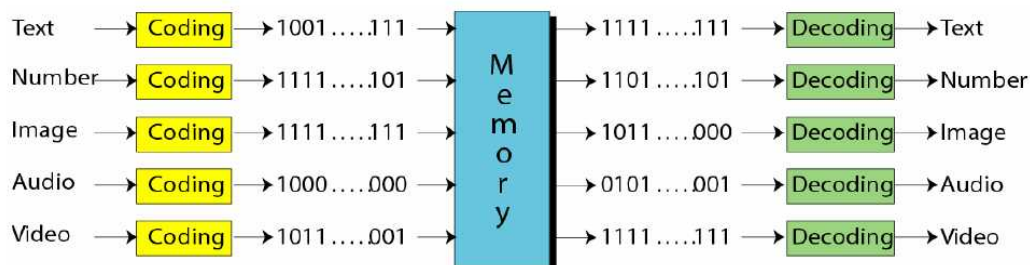
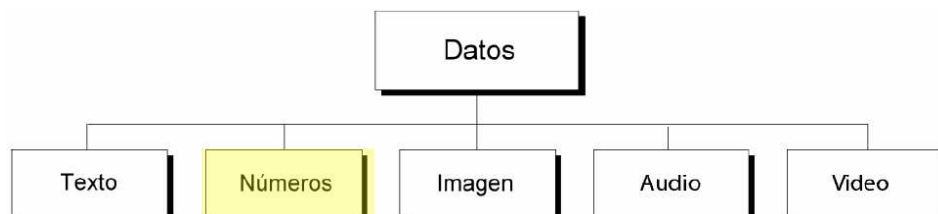
No olvidar



Una cadena de bits por sí misma no significa nada



es la codificación usada la que le da sentido



## 1. Sistemas de numeración

- Binario
  - Generar una tabla de todas las combinaciones de n (n=2, 3, 4, 5) bits
  - Contar en binario: sucesor y antecesor de un número
- Hexadecimal
  - Contar en hexadecimal: sucesor y antecesor de un número
  - Saber crear una tabla de correspondencia entre los 16 dígitos hexadecimales y sus correspondientes 4 bits
- Octal
  - Saber crear una tabla de correspondencia entre los 8 dígitos octales y sus correspondientes 4 bits
  - Contar en octal: sucesor y antecesor de un número

Para más información consultar:

- 01-02-Representacion de informacion numerica-REFERENCIA-01-02-A.pdf
- 01-02-Representacion de informacion numerica-REFERENCIA-01-02-B.pdf

## 2. Conversión entre sistemas de numeración

Conversión de binario a decimal y de decimal a binario

Conversión de octal a binario y de binario a octal

Conversión de hexadecimal a binario y de binario a octal

Para más información consultar:

- 01-02-Representacion de informacion numerica-REFERENCIA-01-02-A.pdf
- 01-02-Representacion de informacion numerica-REFERENCIA-01-02-B.pdf

### 3. Números enteros con signo y sin signo: representación

Artículo de la Wikipedia: [Representación de números con signo](#)

- Representación signo-magnitud
- Representación complemento a 1
- Representación complemento a 2

Para más información consultar:

- 01-02-Representacion de informacion numerica-REFERENCIA-01-02-A.pdf
- 01-02-Representacion de informacion numerica-REFERENCIA-01-02-B.pdf

#### 4. Adición y resta binaria: acarreo y desbordamiento

SUMA BINARIA					RESTA BINARIA				
SIN ACARREO		CON ACARREO			SIN ACARREO		CON ACARREO		
<i>A</i>	<i>B</i>	<i>S</i>	<i>C</i>		<i>A</i>	<i>B</i>	<i>D</i>	<i>C</i>	
0	0	0	0		0	0	0	0	
0	1	1	0		0	1	1	1	
1	0	1	0		1	0	1	0	
1	1	0	1		1	1	0	0	

<i>A<sub>i</sub></i>	<i>B<sub>i</sub></i>	<i>C<sub>i</sub></i>	<i>S<sub>i</sub></i>	<i>C<sub>i+1</sub></i>	<i>A<sub>i</sub></i>	<i>B<sub>i</sub></i>	<i>C<sub>i</sub></i>	<i>D<sub>i</sub></i>	<i>C<sub>i+1</sub></i>
0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1	1	0
0	1	0	1	0	1	0	1	0	0
0	1	1	0	1	1	1	0	0	1
1	0	0	1	0	1	0	1	0	0
1	0	1	0	1	1	1	0	0	1
1	1	0	0	1	1	1	0	0	1
1	1	1	1	1	1	1	1	1	1

La resta en los microprocesadores realmente se hace en complemento a 2 (Ca2): si, por ejemplo, queremos hacer la resta 34 - 25, se realiza la operación 34 + (-25) en binario; estando el -25 expresado en complemento a 2.

Tenemos que tener en cuenta que cuando operamos en complemento a 2 (Ca2) puede aparecer el **overflow/desbordamiento** (o el underflow)

#### EJEMPLOS SENCILLOS DE SUMAS (O RESTAS) EN COMPLEMENTO A 2

En fondo rosa aparecen los bits de signo:

- cuando los operandos tienen el mismo signo y el resultado signo distinto, se ha producido **overflow** (**desbordamiento**) y el resultado no expresa el valor real de la operación.
- entre paréntesis el valor del resultado si considerasemos el bit de acarreo como parte de la representación del número con 4 bits en vez de 3

0000	0	<b>ancho palabra de 3 bits</b>		-	011	+3	-	010	+2	-	011	+3	-	101	-3	-	100	-4
0001	1				010	+2	-	011	+3	-	101	-3	-	101	-3	-	001	1
0010	2				011	+3		010	+2		011	+3		101	-3		100	-4
0011	3				110	-2		101	-3		011	+3		011	+3		111	-1
0100	4	000	0	C=1	001	+1	C=0	111	-1	C=0	110	-2	(6)	000	0	C=1	011	3
0101	5	001	1	V = 0			V = 0			V = 1				000	0	V = 1		(-5)
0110	6	010	2	Z = 0			Z = 0			Z = 0				001	1	Z = 0		
0111	7	011	3	C = 1			C = 0			C = 0				011	-1	C = 1		
				S = 0			S = 1			S = 1				111	-1	S = 0		
1000	-8	100	-4		001	+1		111	-1		100	-4		101	-3		100	-4
1001	-7	101	-3		111	-1	-	001	+1	-	101	-3		100	-4	-	100	-4
1010	-6	110	-2		001	+1		111	-1		100	-4		101	-3		100	-4
1011	-5	111	-1		001	+1		111	-1		011	+3		100	-4		100	-4
1100	-4			C=0	010	+2	C=1	110	-2	C=0	111	-1		001	1	C=1	000	0
1101	-3			V = 0			V = 0			V = 0				001	1	V = 1		(-8)
1110	-2			Z = 0			Z = 0			Z = 0				000	0	Z = 1		
1111	-1			C = 0			C = 1			C = 0				000	0	C = 1		
				S = 0			S = 1			S = 1				000	0	S = 1		

## 5. Operaciones lógicas: AND, OR, XOR, NOT, NAND, NOR

Son operaciones bit a bit:

- cuando son operaciones con 2 operandos: se pone un operando A sobre el operando B, de forma que los bits en las mismas posiciones de ambos coinciden unos encima de otros, y se va realizando la operación aplicando la tabla de verdad (ver a continuación).
  - la acción sobre un par de bits no afecta a ningún otro par (a diferencia de la operación de adición o suma, donde puede existir acarreo/carry).
- cuando es una operación monaria (1 operando): se realiza bit a bit.

Se utilizan mucho en programación, por ejemplo, para crear máscaras (masks) para activar o desactivar bits, con direcciones IP para calcular cuál es la dirección de la red y cuál la de host, ...

### FUNCIONES LÓGICAS BÁSICAS

NOMBRE	AND - Y	OR - O	XOR O-exclusiva	NOT Inversor	NAND	NOR																																																																																	
SÍMBOLO																																																																																							
SÍMBOLO																																																																																							
TABLA DE VERDAD	<table><tr><th>a</th><th>b</th><th>z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	a	b	z	0	0	0	0	1	0	1	0	0	1	1	1	<table><tr><th>a</th><th>b</th><th>z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	a	b	z	0	0	0	0	1	1	1	0	1	1	1	1	<table><tr><th>a</th><th>b</th><th>z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	a	b	z	0	0	0	0	1	1	1	0	1	1	1	0	<table><tr><th>a</th><th>z</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	a	z	0	1	1	0	<table><tr><th>a</th><th>b</th><th>z</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	a	b	z	0	0	1	0	1	1	1	0	1	1	1	0	<table><tr><th>a</th><th>b</th><th>z</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	a	b	z	0	0	1	0	1	0	1	0	0	1	1	0
a	b	z																																																																																					
0	0	0																																																																																					
0	1	0																																																																																					
1	0	0																																																																																					
1	1	1																																																																																					
a	b	z																																																																																					
0	0	0																																																																																					
0	1	1																																																																																					
1	0	1																																																																																					
1	1	1																																																																																					
a	b	z																																																																																					
0	0	0																																																																																					
0	1	1																																																																																					
1	0	1																																																																																					
1	1	0																																																																																					
a	z																																																																																						
0	1																																																																																						
1	0																																																																																						
a	b	z																																																																																					
0	0	1																																																																																					
0	1	1																																																																																					
1	0	1																																																																																					
1	1	0																																																																																					
a	b	z																																																																																					
0	0	1																																																																																					
0	1	0																																																																																					
1	0	0																																																																																					
1	1	0																																																																																					
EQUIVALENTE EN CONTACTOS																																																																																							
AXIOMA	$z = a \cdot b$	$z = a + b$	$z = \bar{a} \cdot b + a \cdot \bar{b}$	$z = \bar{a}$	$z = \overline{a \cdot b}$	$z = \overline{a + b}$																																																																																	

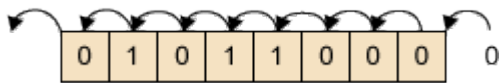
## 6. Operaciones a nivel de bit

Ahora veremos el funcionamiento de las instrucciones lógicas de desplazamiento y rotación:

**a) Desplazamiento lógico a la izquierda (SHL).** Se considera el primer operando como un valor sin signo. Se desplazan los bits a la izquierda tantas posiciones como indica el segundo operando; el bit de más a la izquierda en determinadas arquitecturas se pierde y en otras se copia sobre el bit de transporte y se añaden ceros por la derecha.

**Equivale a ir multiplicando el número por 2**

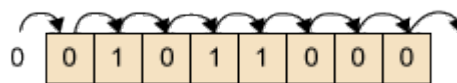
Ejemplo de desplazamiento de un bit a la izquierda



**b) Desplazamiento lógico a la derecha (SHR).** Se considera el primer operando como un valor sin signo; se desplazan los bits a la derecha tantas posiciones como indica el segundo operando, el bit de la derecha en determinadas arquitecturas se pierde y en otras se copia sobre el bit de transporte y se añaden ceros por la izquierda.

**Equivale a ir dividiendo el número sin signo por 2**

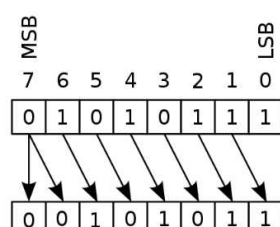
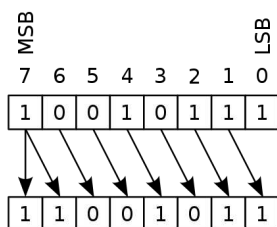
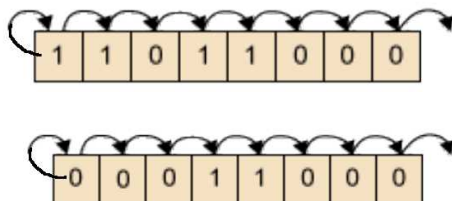
Ejemplo de desplazamiento de un bit a la derecha



**c) Desplazamiento aritmético a la derecha (SAR).** Se considera el primer operando como un valor con signo expresado en complemento a 2. El bit de más a la izquierda, bit de signo, se conserva y se va copiando sobre los bits que se van desplazando a la derecha. Se desplaza a la derecha tantas posiciones como indica el segundo operando y los bits de la derecha en determinadas arquitecturas se pierden y en otras se copian sobre el bit de transporte.

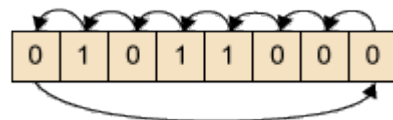
Nota

Ejemplos de desplazamiento aritmético de un bit a la derecha



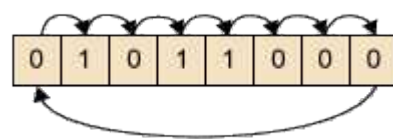
**d) Rotación a la izquierda (ROL).** Se considera el primer operando como un valor sin signo; se desplazan los bits a la izquierda tantas posiciones como indica el segundo operando, el bit de más a la izquierda se pasa a la posición menos significativa del operando.

Ejemplo de rotación de un bit a la izquierda



**f) Rotación a la derecha (ROR).** Se considera el primer operando como un valor sin signo; se desplazan los bits a la derecha tantas posiciones como indica el segundo operando, el bit de más a la derecha se pasa a la posición más significativa del operando.

Ejemplo de rotación de un bit a la derecha



## 7. Números decimales: coma flotante IEEE 754

Ver las webs:

- [IEEE-754 Floating Point Converter](#)
- [Base Convert: IEEE 754 Floating Point](#)
- [IEEE-754 Floating-Point Conversion](#): formato antiguo
- [IEEE-754 Analysis](#) (el más moderno)

La norma IEEE 754 define los siguientes formatos:

- Precisión simple Rango aproximado  $[-10^{38}, 10^{38}]$

$$\text{valor} = (-1)^{\text{signo}} \times 1.\text{mantisa} \times 2^{\text{exponente}-127}$$

signo	exponente	mantisa
1 bit	8 bits	23 bits

- Precisión doble Rango aproximado  $[-10^{308}, 10^{308}]$

$$\text{valor} = (-1)^{\text{signo}} \times 1.\text{mantisa} \times 2^{\text{exponente}-1023}$$

signo	exponente	mantisa
1 bit	11 bits	52 bits

- Precisión extendida Rango aproximado  $[-10^{4932}, 10^{4932}]$

$$\text{valor} = (-1)^{\text{signo}} \times 1.\text{mantisa} \times 2^{\text{exponente}-16383}$$

signo	exponente	mantisa
1 bit	15 bits	64 bits