Representación digital de los números

Fundamentos de Computadores

Escuela Politécnica Superior. U.A.M

Índice de la Unidad 6

Escuela Politécnica Superior

Representación de números enteros, positivos y negativos

Recuerda: Un número binario natural, entero sin signo, utiliza un sistema numérico posicional.

- Con un número de n bits: $b_{n-1}\, b_{n-2} \ldots b_1 b_{1 0}$, se pueden representar **2^n** números diferentes en el rango **$[0, 2^n-1]$**.

$$\boxed{N = \sum_{i=0}^{n-1} \mathrm{b}_i \times 2^i}$$

¿Cómo se representa en binario un entero con signo?

¿Cómo se representa en binario un número real?

Representación de números enteros, positivos y negativos

Signo-Magnitud

- Para un número de **n** bits (b_{n-1}, b_{n-2} … b_1, b_0), el bit más significativo (msb) señala el signo, los **n-1** bits restantes la magnitud.
  - Si msb = '0', el número es positivo.
  - Si msb = '1', el número es negativo.

$$\boxed{N_{sm} = (-1)^{b_{n-1}} x \sum_{i=0}^{n-2} b_i \times 2^i}$$

- Ejemplo: escribir con 6 bits y representación signo-magnitud los números decimales +6, -6, +12, -24, +32 y -40.

+6 =

+12 =

+32 =

- 6 =

- 24 =

- 40 =

Escuela Politécnica Superior

Representación de números enteros, positivos y negativos

Signo-Magnitud

- Rango de representación de enteros en signo-magnitud:

$$-(2^{n-1} - 1) \text{ a } (2^{n-1} - 1)$$

- Problemas de la representación de enteros en signo-magnitud:
  - ✓ Dos representaciones para el cero "±0" (000..00 y 100..00)
  - ✓ La extensión en bits del número no es igual para ambos signos
  - ✓ La suma de números con distinto signo (resta) no funciona bien
    - ✓ Ejem: (-6) + (+6) =>

```
   10000110
+  00000110
   --------
   10001100 (-12, ¡error!)
```

Escuela Politécnica Superior

Representación de números enteros, positivos y negativos

Complemento a 2

- **CODIFICACIÓN** en Complemento a 2. Representa el valor de un número en un sistema binario posicional, en un número de **n** bits (b_{n-1}, b_{n-2} … b_1, b_0), el bit más significativo (msb) tiene el valor $\mathbf{-2^{n-1}}$
  - Si msb = '0', el número es positivo.
  - Si msb = '1', el número es negativo.

$$\boxed{N_{c2} = b_{n-1} x(-2^{n-1}) + \sum_{i=0}^{n-2} \mathrm{b}_i \times 2^i}$$

- **OPERACIÓN** de Complemento a 2 de un número entero. Equivale a la inversion de signo de un número

  1. Invertir todos los bits
  2. Sumar 1 al resultado

NO confundir CODIFICACIÓN con OPERACIÓN

Escuela Politécnica Superior

Representación de números enteros, positivos y negativos

Complemento a 2

- Ejemplo: escribir con 8 bits y representación c2 los números enteros dados en decimal:

+6 =	+89 =	+112 =	-125 =
- 6 =	- 24 =	- 75 =	- 142 =

- Ejemplo: calcular el valor en decimal de los números escritos en binario con 8 bits y representación en c2:

01110000 =	11010011 =
11110000 =	01000100 =

- Ejemplo: calcular el valor en decimal de los números escritos en hexadecimal y representación en c2:

0xC5 =	0x7F =
0x6D =	0x84 =

Escuela Politécnica Superior

Representación de números enteros, positivos y negativos

Complemento a 2

- Rango de la representación de enteros en Complemento a 2:

$$(-2^{n-1}) \text{ a } (2^{n-1}-1)$$

- La extensión de signo no modifica el valor del número:
  - Ejemplo 4=>8 bits, nº positivo: (+5) = 0101 = 00000101
  - Ejemplo (4=>8 bits, nº negativo: (-5) = 1011 = 11111011
- La operación de restar es equivalente a sumar el minuendo con el c2 del sustraendo:

  ✓ Ejem: (+6) + (-6) =>

```
   00000110
+  11111010
-----------
   00000000 (¡correcto!)
```

Suma de números en binario

- Suma decimal

```
      11  ← carries
     3734
+    5168
---------
     8902
```

- Suma Binaria

```
      11  ← carries
     1011
+    0011
---------
     1110
```

Ejemplos

0001

```
   1001
+  0101
-------
   1110
```

1110

```
   1011
+  0110
-------
   0001
```

0110

```
   0111
+  0110
-------
   1101
```

Problema de *overflow* (desbordamiento) en los sistemas digitales

- Los sistemas digitales operan con un número fijo de bits.
- El resultado de una operación (suma) puede sobrepasar el rango de representación de los bits utilizados. XOR de los dos últimos acarreos, si es 0 el resultado es correcto

Operaciones en complemento a 2

Ejemplos: Utilizando números de 8 bits en la notación de c2, realizar las operaciones:

- ✓ Sumar (+45) + (+32)
- ✓ Restar (+45) - (+32)
- ✓ Sumar (-35) + (-27)
- ✓ Restar (+35) - (-27)
- ✓ Sumar (100) + (-12)
- ✓ Restar (-128) - (+64)

Escuela Politécnica Superior

Sumador binario

- ✓ **Semisumador (1 bit):** Circuito combinacional para la suma aritmética de los dos bits de la entrada (a_i y b_i), obteniendo a la salida un bit para la suma y un bit para el acarreo (s_i y c_{i+1})

Tabla de verdad:

Bit1	Bit2	Suma	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

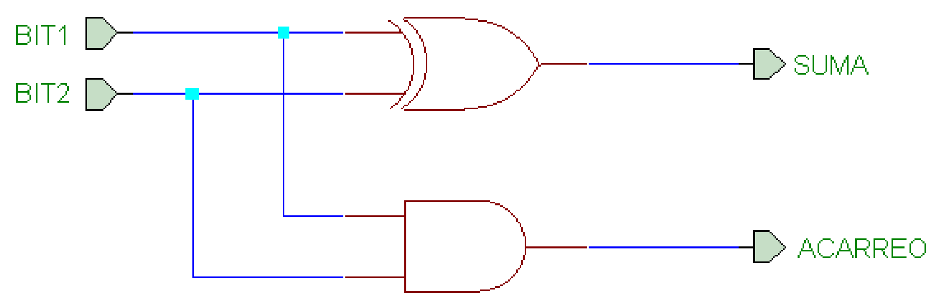
La ecuación para el bit de suma corresponde a una operación XOR:

$$\boxed{Suma = Bit1 \oplus Bit2}$$

La ecuación para el bit de acarreo corresponde a una AND:

$$\boxed{C_{out} = Bit1 \cdot Bit2}$$

Circuito:



Sumador binario

- ✓ **Sumador completo (1 bit):** Circuito combinacional para la suma aritmética de los dos bits de la entrada mas el acarreo del bit anterior (a_i, b_i y c_i), obteniendo a la salida un bit para la suma y un bit para el acarreo (s_i y c_{i+1})

Tabla de verdad:

C_{in}	Bit1	Bit2	Suma	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Donde:

C_{in}: Acarreo de entrada

C_{out}: Acarreo de salida

La ecuación para el bit de suma:

$$\boxed{Suma = (Bit1 \oplus Bit2) \oplus C_{in}}$$

La ecuación para el bit de acarreo:

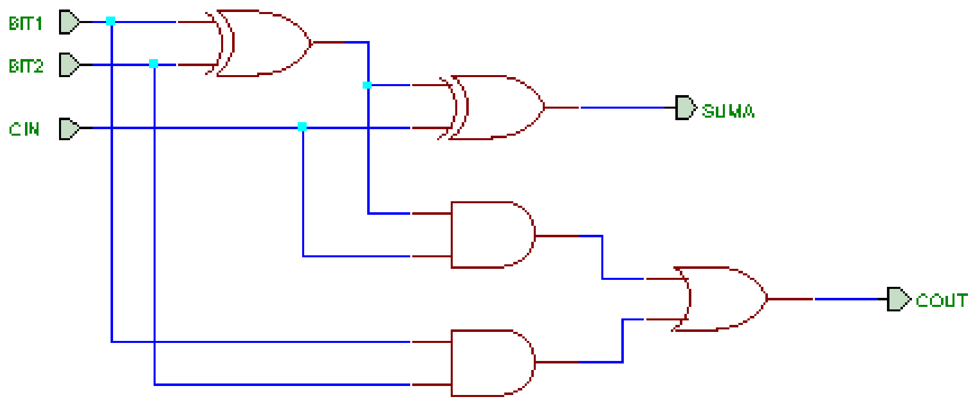
$$\boxed{C_{out} = Bit1 \cdot Bit2 + (Bit1 \oplus Bit2) \cdot C_{in}}$$

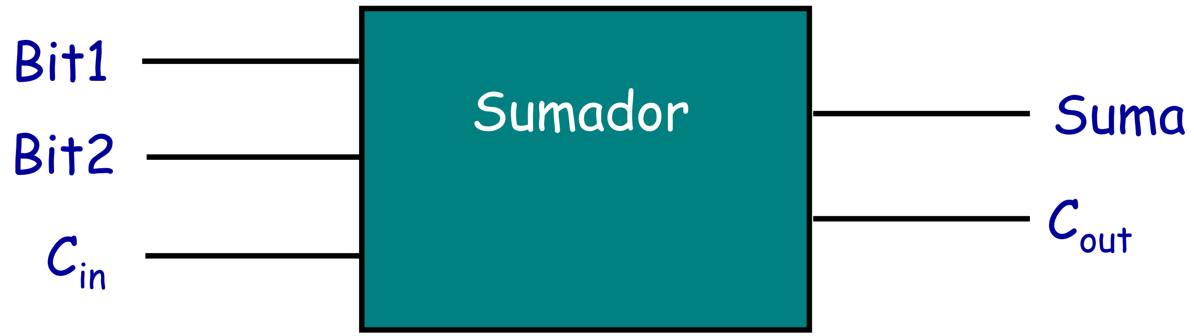
Escuela Politécnica Superior

Sumador binario

✓ **Sumador completo.** Circuito

Circuito esquemático



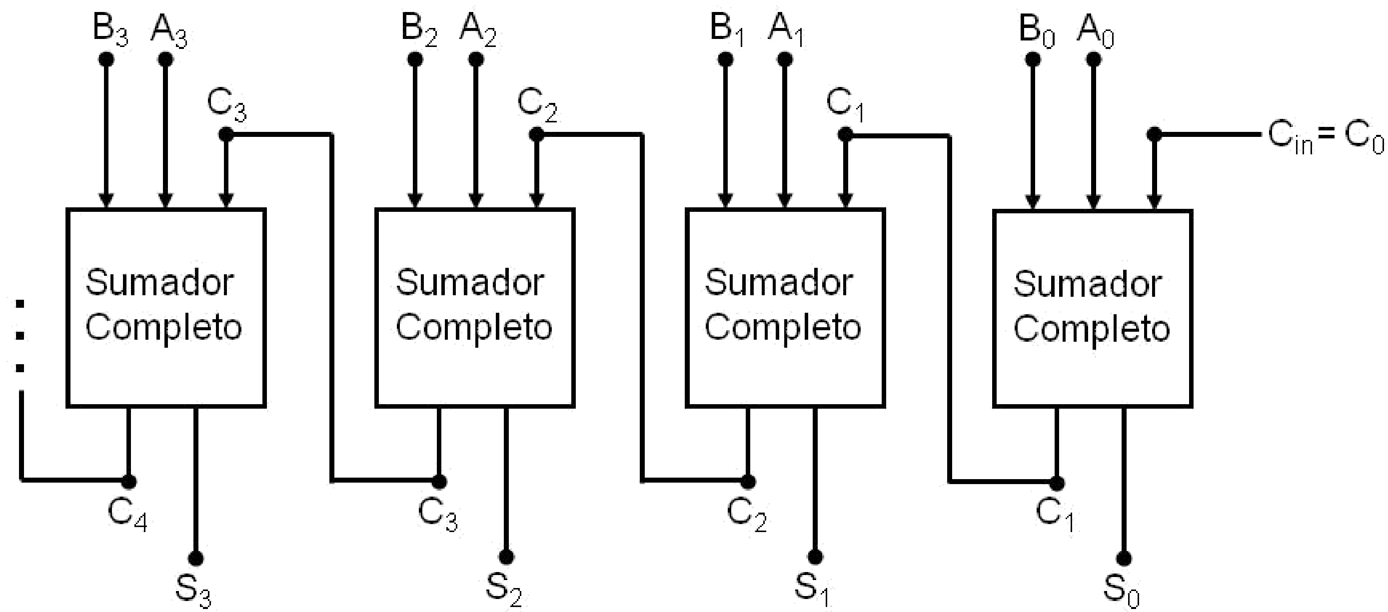


Esquema de bloque

Escuela Politécnica Superior

Sumador binario

- ✓ **Circuito sumador para n bits en paralelo con acarreo en serie.**
  - ✓ Un circuito sumador elemental para cada bit.
  - ✓ Los bits del mismo peso se suman dos a dos
  - ✓ Para obtener cada suma parcial se necesita el acarro que se produce en la suma precedente.



Representación de números reales

✓ Representación en coma fija

Es una representación binaria numérico-posicional en la cual el número de bits dedicados a la parte entera y a la parte fraccionaria es fijo. Los números negativos se representan en complemento a 2:

$$\boxed{N = \mathrm{b}_{n-1} x(-2^{n-1}) + \sum_{i=-k}^{n-2} \mathrm{b}_i \times 2^i}$$

Ejemplos (n = 8; k = 2)

$30{,}3125_{10} =$

$0{,}375_{10} =$

$-0{,}1875_{10} =$

Escuela Politécnica Superior

¡¡¡ Existe un problema de precisión !!!

Representación de números reales

✓ Coma flotante, estándar IEEE-754 (32b)

Un número N se representa con tres grupos de bits:

$$N = (-1)^s \times M \times 2^e$$

Signo	Exponente	Mantisa

- Notación similar a la notación científica en decimal
- <u>Signo</u> (s) (1 bit): indica el signo del número, "0" si positivo y "1" si negativo.
- <u>Exponente</u> (n=8 bits). Con notación sesgada o en exceso ($2^{n-1} - 1$), indica el exponente de una potencia en base 2.
  - Exponente = e + exponente en exceso = $e+(2^7 - 1)=e+127$
- <u>Mantisa</u> (m) (23 bits): multiplicado por la potencia binaria, indica el valor absoluto del número.
  - La mantisa se representa como M = 1,m; donde m (23 bits) es la parte que se guarda. El valor absoluto se mantiene a través de la modificación del valor del exponente

Escuela Politécnica Superior

Representación de números reales

✓ Coma flotante, estándar IEEE-754 (32b)

- Precisión simple (*single-precision*):
  - 32-bits en total
  - 1 bit de signo, 8 de exponente y 23 de mantisa
  - sesgo = 127
- Precisión doble (*double-precision*):
  - 64-bits en total
  - 1 bit de signo, 11 de exponente y 52 de mantisa
  - sesgo = 1023

Representación de números reales

✓ Coma flotante, estándar IEEE-754 (32b)

Ejemplo: representar 228_{10} en CF en el estándar IEEE-754 (32bits)

- Convertir el número decimal a binario:

$$228_{10} = 11100100_2 = 1{,}11001 \times 2^7$$

- Rellenar cada parte del número (s):

Bit de signo positivo (0)

1 bit	8 bits	23 bits
0		
Sign	**Exponent**	**Mantissa**

Representación de números reales

✓ Coma flotante, estándar IEEE-754 (32b)

(continuación): representar 228_{10} en CF en el estándar IEEE-754 (32bits)

- Convertir el número decimal a binario:

$$228_{10} = 11100100_2 = 1{,}11001 \times 2^7$$

- Rellenar cada parte del número (e):
  - Exponente con sesgo = sesgo + exponente
  - Sesgo => 127
  - El exponente, 7, se almacena como: $127+7 = 134 = 10000110_2$

1 bit	8 bits	23 bits
0	**10000110**	
Sign	**Exponent**	**Mantissa**

Representación de números reales

✓ Coma flotante, estándar IEEE-754 (32b)

(continuación): representar 228_{10} en CF en el estándar IEEE-754 (32bits)

- Convertir el número decimal a binario:
  - $228_{10} = 11100100_2 = 1{,}11001 \times 2^7$
- Rellenar cada parte del número (m):
  - El primer bit de la mantisa siempre es 1:
    - guardar ese 1, llamado el 1 implícito, es redundante.
    - no se almacena, hay más sitio para la parte fraccionaria.

1 bit	8 bits	23 bits
0	10000110	11001000000000000000000
Sign	**Exponent**	**Mantissa**

- En hexadecimal: 0x43640000

Representación de números reales

✓ Coma flotante, estándar IEEE-754 (32b)

- Ejemplo: representar $-58{,}25_{10}$ en el estándar IEEE 754 de 32-bits.
- Primero, convertir de decimal a binario y notación científica:
  - $58{,}25_{10} = 111010{,}01_2 = 1{,}1101001 \times 2^5$

  Después, hallar los tres campos del número:
  - Bit de signo: 1 (negativo)
  - Exponente sesgado en 8 bits: $(127 + 5) = 132 = 10000100_2$
  - 23 bits de mantisa: 110 1001 0000 0000 0000 0000

1 bit	8 bits	23 bits
1	100 0010 0	110 1001 0000 0000 0000 0000
Sign	Exponent	Fraction

- En hexadecimal: 0xC2690000

Representación de números reales

✓ Coma flotante, IEEE-754 (32b). Casos especiales

- El estándar IEEE 754 incluye casos especiales para números o valores díficiles de codificar. Por ejemplo, el 0 no tiene un bit 1 para el 1 implícito y tiene dos representaciones.

Número	Signo	Exponente	Mantisa
0	X	00000000	00000000000000000000000
∞	0	11111111	00000000000000000000000
- ∞	1	11111111	00000000000000000000000
NaN	X	11111111	Distinto de cero

NaN (*Not a Number*) se usa para números que no existen, como por ejemplo √-1 o log(-5).

Representación de números reales

Ejemplos de números en coma flotante (CF, FP)

- Precisión simple (32 bits): 1/8/23.

  Ejemplos:

  - $30{,}3125_{10} = (-1)^0\ 1{,}11100101 \times 2^4$.

  (**s e m**) 0 10000011 11100101000000000000000 ($41F28000_{16}$)

  - $0{,}375_{10} = (-1)^0\ 1{,}1 \times 2^{-2}$

  (**s e m**) 0 01111101 10000000000000000000000 ($3EC00000_{16}$)

  - $-0{,}1875_{10} = (-1)^1\ 1{,}1 \times 2^{-3}$

  (**s e m**) 1 01111100 10000000000000000000000 ($BE400000_{16}$)

- Precisión doble (64 bits): 1/11/52

Escuela Politécnica Superior

Otros códigos binarios

Código decimal binario (BCD)

Es una representación para números enteros sin signo, en la que cada dígito decimal (0, 1, …, 9) tiene su equivalente binario en 4 bits.

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

Ejemplos de números BCD

$30_{10} =$

$375_{10} =$

$1875_{10} =$

Escuela Politécnica Superior

Otros códigos binarios

Representación de texto. El estándar ASCII

ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) es un código binario que se utiliza para representar caracteres. El código ASCII extendido utiliza 8 bits para identificar caracteres adicionales a un alfabeto tradicional.

Carácter	ASCII	Carácter	ASCII	Carácter	ASCII
0	30_{16}	A	41_{16}	"espacio"	20_{16}
1	31_{16}	B	42_{16}	%	25_{16}
2	32_{16}	C	43_{16}	~	$7E_{16}$
…	…	…	…	/	$2F_{16}$
7	37_{16}	a	61_{16}	ñ	$A4_{16}$
8	38_{16}	b	62_{16}	à	$A0_{16}$
9	39_{16}	c	63_{16}	@	64_{16}

Escuela Politécnica Superior

Códigos para tratamiento de errores

Necesidad del tratamiento de errores

- Posibilidad de cometer errores
  - En un sistema informático la información circula entre diferentes elementos digitales y se almacena en otros dispositivos también digitales.
  - Puede haber errores debido a:
    - Ruidos en las comunicaciones
    - Defectos en las superficies de los discos, etc...
  - Los errores consisten en la modificación de la información desde que se emite (o almacena) hasta que se recibe (o se recupera).
    - Cambio de valor de algunos bits (0 ⇔ 1)

Escuela Politécnica Superior

Códigos para tratamiento de errores

- Códigos de paridad
  - **VRC** (**V**ertical **R**edundancy **C**hecking)
    - La información se coloca en bloques de longitud fija
    - A los bloques se les añade un bit llamado de paridad y que, normalmente, precede a la información

Criterios para la paridad

- Paridad par:
  - Nº total de "1" (en datos) par: Bit de paridad = 0
  - Nº total de "1" (en datos) impar: Bit de paridad = 1
- Paridad impar:
  - Nº total de "1" (en datos) par: Bit de paridad = 1
  - Nº total de "1" (en datos) impar: Bit de paridad = 0

Códigos para tratamiento de errores

Comprobación de paridad

Completar el bit de paridad con criterio:

- ✓ (1) paridad impar
- ✓ (2) paridad par

1	2	Información						
		1	0	0	0	0	0	1
		0	1	0	1	1	1	1
		1	1	0	1	0	0	0
		1	1	1	0	1	1	1
		1	0	1	0	0	0	1
		0	1	1	1	1	1	1
		0	1	1	1	0	0	1
		0	0	0	1	1	0	1

Escuela Politécnica Superior

Códigos para tratamiento de errores

- **Paridad vertical, longitudinal y cruzada**

  La información se coloca en grupos de (m) bloques de longitud fija (k) como matriz k x m o m x k

  Ejemplo: Se quiere enviar la información "IBM" en ASCII (7 bits), es decir: $49_{16}\ 42_{16}\ 4D_{16}$ = $1001001\ 1000010\ 1001101_2$ (m=3, k=7)

  Si paridad "par", se añade:
  - ➢ Bit para VRC criterio par (verde, primera columna)
  - ➢ Bit para LRC criterio par (azul, última fila)
  - ➢ Bit de paridad cruzada criterio par (rojo)

VCR

Paridad Cruzada

LCR

1	1	0	0	1	0	0	1
0	1	0	0	0	0	1	0
0	1	0	0	1	1	0	1
1	1	0	0	0	1	1	0

Se transmite: $C9424DC6_{16}$

Códigos para tratamiento de errores

- Paridad vertical, longitudinal y cruzada
  - Detección y corrección de un error simple

0	1	0	0	0	1	1	1
0	1	0	0	0	0	0	1
0	1	0	1	0	0	0	0
0	1	0	1	0	1	1	0

Datos originales

0	1	0	0	0	1	1	1
0	1	0	0	0	0	0	1
0	1	0	1	1	0	0	0
0	1	0	1	0	1	1	0

Error en recepción

0	1	0	0	0	1	1	1
0	1	0	0	0	0	0	1
0	1	0	1	1	0	0	0
0	1	0	1	0	1	1	0

Error en paridad

0	1	0	0	0	1	1	1
0	1	0	0	0	0	0	1
0	1	0	1	1	0	0	0
0	1	0	1	0	1	1	0

Bit sospechoso

Códigos para tratamiento de errores

- **Checksum**
  - Añade uno o más bytes a la información para alcanzar un resultado conocido en la suma total.

  Ej.: Dato: 37 4A. Cheksum: 7F (37+4A+7F = 1**00**)

  Información transmitida: 37 4A 7F
- **Códigos polinómicos o de redundancia cíclica (CRC)**
  - Añade bits a la información para alcanzar una división exacta por un polinomio conocido.

  Ej.: $G(x)=(x^3+x+1)$. Dato: 11000011. {11000011 mod 1011 =1000}

  Información transmitida: 110000111000
- **Códigos i en n**
  - Utiliza códigos con el mismo número de bits de valor '1'.

  Ej: código 5043210 (2 en 7): 0->0100001; 1->0100010; 2->0100100…

  …7->1000100; 8->1001000; 9->1010000