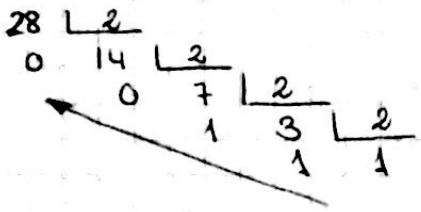
SISTEMAS DE REPRESENTACIÓN

1. BINARIO PURO

Los números escritos en binario puro solo pueden ser positivos, ya que no se contempla la representación de los negativos. Por tanto, el rango de representación es: $[0, 2^n - 1]$.

Para pasar la parte entera de un número decimal a binario hay que dividir entre 2 sucesivamente. Ejemplo



$28 = 11100$

Para pasar la parte fraccionaria multiplicamos sucesivamente hasta tener la unidad.

$0,75 \cdot 2 = 1,50$ ↓
$0,50 \cdot 2 = 1,00$ ↓

$0,75 = 0,11$

De manera general, si una secuencia de n dígitos binarios $a_{n-1} a_{n-2} \cdots a_1 a_0$ es interpretada como un entero sin signo A, su valor es:

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

2. SIGNO - MAGNITUD

Representación que emplea un bit de signo. En una palabra de n bits, los n-1 bits de la derecha representan la magnitud del entero. Por ejemplo.

$+18 = 0001\,0010$
$-18 = 1001\,0010$

En caso general, puede expresarse: $A = \begin{cases} \sum_{i=0}^{n-2} 2^i a_i & \text{si } a_{n-1} = 0 \\ -\sum_{i=0}^{n-2} 2^i a_i & \text{si } a_{n-1} = 1 \end{cases}$

La representación signo-magnitud posee varias limitaciones:

1. La suma y la resta tienen que tener en cuenta los signos de los números además de sus magnitudes relativas.

2. Existen dos representaciones del número 0. Esto supone un inconveniente ya que es más difícil comprobar el número 0.

Debido a estas limitaciones, cada vez se usa la representación signo-magnitud para implementar en la ALU las operaciones con enteros. En su lugar, el esquema más común es la representación en complemento a dos.

3. Complemento a dos (C2)

Al igual que la de signo-magnitud, se usa el bit más significativo como bit de signo, facilitando la comprobación de si es entero o no.

El rango de esta representación es $[-2^{n-1}, 2^{n-1}-1]$

Considerando **A** un número entero de n bits positivo tenemos que los a_{n-1} bits representan la magnitud del número de la misma forma que en la representación signo-magnitud es decir:

$$A = \sum_{i=0}^{n-2} 2^i a_i \quad \text{para } A \geq 0$$

Mientras que para valores negativos quedaría:

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

Una forma de hallar el opuesto de un número binario positivo en complemento a dos es comenzar por la derecha (el dígito menos significativo), copiando el número original (de derecha a izquierda) hasta encontrar el primer 1, después de haber copiado el 1, se niegan (complementan) los dígitos restantes.

Otra forma es negar todos los dígitos (se halla el complemento a 1) y después sumar 1 al resultado.

Ejemplos:

Nº decimal	binario	C2
7	0111	0111
-3	0011	1101

4. Complemento a uno (C1)

Se obtiene cambiando cada uno de los bits del número binario por su complementario (cambiar unos por ceros).

Ejemplos:

Nº decimal	binario	C1
5	0101	0101
-7	0111	1000

5. Representación en exceso a M ($EX-M_n$)

Consiste en hacer corresponder los códigos del conjunto origen con los valores representados más un cierto exceso M.

El exceso M, en principio, es arbitrario aunque normalmente se elige de forma que la aritmética se simplifique o adquiera alguna característica deseada

El rango no presenta discontinuidades: $[-M, 2^n - 1 - M]$

Un valor muy común de M es 2^{n-1}

6. Rango en exceso a 2^{n-1} $(EX\text{-}2^{n-1}{}_n)$

Rango de representación: $\text{rango}(EX\text{-}2^{n-1}) = [-2^{n-1}, 2^{n-1}-1]$

El rango no es simétrico y cuenta con una representación para el 0.

El bit más significativo de las representaciones de enteros positivos es 1 mientras que las representaciones de los enteros negativos siempre comienzan por 0.

El bit más significativo indica el signo pero no es el bit de signo ya que codifica valor también.

* Definición del operador $EX\text{-}2^{n-1}{}_n\ (Z, n)$

Sea $Z \in \mathbb{Z}$ y $b = 2$. El operador $EX\text{-}2^{n-1}\ (Z, n)$ devuelve:

$$EX\text{-}2^{n-1}(Z,n) = \begin{cases} \text{BIN}\ (Z + 2^{n-1}, n) = Z_{EX\text{-}2^{n-1}, n} \\ \text{des} = '0' \text{ si } Z \in \text{rango}\ (EX\text{-}2^{n-1}{}_n) \\ \text{des} = '1' \text{ si } Z \notin \text{rango}\ (EX\text{-}2^{n-1}{}_n) \end{cases}$$

Por otro lado, para calcular el valor en base 10 de un número entero (N):

$$N_{EX} = \left[\left(\sum_{i=0}^{n-1} a_i \cdot 2^i\right) - 2^{n-1}\right]_{10}$$

- $0000\ 0000\ 0100\ 1001_{\text{EX. a } 32768} = \left((1\cdot 2^6 + 1\cdot 2^3 + 1\cdot 2^0) - 2^{15}\right)_{10} = \left((64+8+1) - 32768\right)$
$= -32\,695_{10}$

- Para $n = 8$ en exceso $2^{n-1} - 1 = 2^{8-1} - 1 = 127$

Valor 128 $\xrightarrow{\text{binario}}$ 1000 0000 $\xrightarrow{\text{en exceso}}$ 1111 1111
Valor −125 $\xrightarrow{\text{binario}}$ 0111 1101 $\xrightarrow{\text{en exceso}}$ 0000 0010

7. Representación en coma fija

Con esta notación es posible representar un rango de enteros positivos y negativos centrado en el cero. Asumiendo una coma binaria fija, dicho formato permite también representar números con parte fraccionaria.

Esto conlleva ciertos problemas:

1. Los números muy grandes no pueden representarse pero tampoco las fracciones muy pequeñas.

2. La división de dos números grandes puede perderse la parte fraccionaria del cociente.

Precisión constante $1/2^9$

8. REPRESENTACIÓN EN COMA FLOTANTE

Para números decimales, la limitación de la coma fija se supera utilizando la notación científica donde lo que se hace es mover dinámicamente la coma decimal a una posición conveniente y utilizar el exponente para mantener registrada la posición de la coma. Esto permite representar un rango de números muy grandes y muy pequeños con solo unos cuántos dígitos.

Esa misma técnica puede aplicarse a los números binarios. Podemos representar un número en la forma:

$$\pm S \times B^{\pm E}$$

$\pm E \rightarrow$ exponente

$S \rightarrow$ parte significativa o mantisa

$B \rightarrow$ base implícita

Ejemplos: $1,1010001 \times 2^{10100} = 0,1001\ 0011\ \$1010001000000000000000 =$
$= 1,6328125 \cdot 2^{20}$

- Mantisa normalizada

La mantisa será la parte de mayor peso (la que situemos a la derecha de la coma) donde encontraremos una cifra significativa.

¿Cuándo será una cifra significativa? Cuando cumpla $0, d_{n-1}\ d_{n-2} \Rightarrow d_{n-1} = \overline{d_{n-2}}$

SM → $\pm$ 0, 1 [redacted]

C2 / C1 / $EX2^{n-1}$ } 0, [redacted] → ⊕ 0,01 ⊖ 0,10

VENTAJAS: Aporta un bit más de representación
DESVENTAJA: No puedo representar el 0

RANGO_COMA_FLOTANTE $\in$ [rango_neg] $\cup$ [rango_pos]

$$[M_{inf-} \times r^{E_{sup}}, M_{sup-} \times r^{E_{inf}}] \cup [M_{inf+} \times r^{E_{inf}}, M_{sup+} \times r^{E_{sup}}]$$

PRECISIÓN $f(q, E) = \frac{1}{2^q} \times r^E$

Ejemplos: Representar el número real 7,625 en coma flotante SM, C2 con 8 bits y exponente con exceso 2^{n-1} sobre 8 bits.

7,625 → binario → 0111, 1010

SM: $0,111\ 101 \cdot 2^3 \rightarrow 0,111\ 1010\quad 1000\ 0011$

C2: $0,0111\ 1010 \cdot 2^4 \longrightarrow 0,0111\ 1010\quad 1000\ 0100$