Sistemas Digitales

Tomás Agustín Hernández

1. Introducción a los sistemas de representación

Magnitud

Llamamos magnitud al tamaño de algo, dicho en una medida específica. Es representada a través de un sistema que cumple 3 conceptos fundamentales:

- Finito: Debe haber una cantidad finita de elementos.
- Composicional: El conjunto de elementos atómicos deben ser fáciles de implementar y componer.
- Posicional: La posición de cada dígito determina en qué proporción modifica su valor a la magnitud total del número.

Algunos de los sistemas de representación más utilizados son: binario, octal, decimal y hexadecimal.

Bases

Una base nos indica la cantidad de símbolos que podemos utilizar para poder representar determinada magnitud.

Base	**Símbolos disponibles**
2 (binario)	0, 1
8 (octal)	0, 1, 2, 3, 4, 5, 6, 7
10 (decimal)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
16 (hexadecimal)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Tabla 1: Bases más utilizadas

La tabla anterior representa los símbolos disponibles para las bases 2, 8, 10 y 16.

Consideremos por un momento que estamos en binario; ¿sería correcto que 1 + 1 = 2? ¡No! Porque 2 no es un símbolo válido en base 2.

Para indicar la base en la que está escrito un número, se coloca la base entre paréntesis en la esquina inferior derecha.

$1024_{(10)}$: 1024 representado en base 10 (decimal)

Digítos/Bits

Sea $n \in \mathbb{Z}$, cuando decimos que tenemos n bits es lo mismo que decir que tenemos n dígitos.

- 0001: Representa el número 1 en binario, en 4 bits/dígitos.
- 0010: Representa el número 2 en binario, en 4 bits/dígitos.

Teorema de división

Es una manera de poder realizar un cambio de base de un número decimal a otra base. La representación en la otra base es el resto visto desde abajo hacia arriba.

$$a = k * d + r \ con \ 0 \leq r < |d|$$

donde:

- k = cociente
- d = divisor.
- r = resto de la división de a por d.

Pasaje del número $128_{(10)}$ a $128_{(2)}$ en 8 bits

$$128 = 64 * 2 + 0$$

$$64 = 32 * 2 + 0$$

$$32 = 16 * 2 + 0$$

$$16 = 8 * 2 + 0$$

$$8 = 4 * 2 + 0$$
$$4 = 2 * 2 + 0$$
$$2 = 1 * 2 + 0$$
$$1 = 0 * 2 + 1$$

Luego, $128_{(2)} = 1000\ 0000$

Bit más significativo / menos significativo

El bit más significativo en un número es el que se encuentra a la izquierda, mientras que el menos significativo es el que se encuentra a la derecha.

$$\mathbf{1}0000000\mathbf{0}_{(2)}$$

Tipos numéricos

Representemos números naturales y enteros a partir de la representación en base 2 (binario)

Sin signo: Representa únicamente números positivos. No se pueden utilizar los símbolos de resta (**-**) ni tampoco coma (**,**)

$$1_{(10)} = 01_{(2)}$$
$$128_{(10)} = 10000000_{(2)}$$

Signo + Magnitud: Nos permite representar números negativos en binario.
El bit más significativo indica el signo

- 0: número positivo
- 1: número negativo.

$$18_{(10)} = \mathbf{0}0010010_{(2)}$$
$$-18_{(10)} = \mathbf{1}0010010_{(2)}$$

Representar números en S+M suele traer problemas porque el 0 puede representarse de dos maneras

$$+0_{(10)} = \mathbf{0}0000000_{(2)}$$
$$-0_{(10)} = \mathbf{1}0000000_{(2)}$$

Para solucionar este problema, las CPU utilizan la notación Complemento a 2 (C_2)

Exceso m: Sea $m \in \mathbb{Z}$, decimos que un número n está con exceso m unidades cuando $m > 0$

$$n_0 = n + m$$
$$n = 1 \land m = 10 \longrightarrow n_0 = -9$$

Nota: n_0 indica el valor original de n antes de ser excedido m unidades.

Complemento a 2: Los positivos se representan igual.
El bit más significativo indica el signo, facilitando saber si el número es positivo o negativo. Cosas a tener en cuenta

- **Rango**: -2^{n-1} *hasta* $2^{n-1} - 1$
- **Cantidad** de representaciones del cero: Una sola
- **Negación**: Invierto el número en representación binaria positiva y le sumo uno.
- **Extender número a más bits**: Se rellena a la izquierda con el valor del bit del signo.
- **Regla de Desbordamiento**: Si se suman dos números con el mismo signo, solo se produce desbordamiento cuando el resultado tiene signo opuesto.

Overflow / Desbordamiento

Hablamos de overflow/desbordamiento cuando

- El número a representar en una base dada, excede la cantidad de bits que tenemos disponibles.
- Si estamos en notación C_2 al sumar dos números cambia el signo.

Acarreo / Carry

Ocurre cuando realizamos una suma de números binarios y el resultado tiene más bits que los números originales que estamos sumando

Suma entre números binarios

Se hace exactamente igual que una suma común y corriente.
Es importante prestar atención a la cantidad de dígitos que nos piden para representarlo, y en caso de estar en C_2 que el signo no cambie.

Hagamos sumas en C_2 (sin límite de bits)

$$\begin{array}{r} 0010 = 2 \\ +1001 = -7 \\ \hline 1011 = -5 \end{array} \qquad \begin{array}{r} 0101 = 5 \\ +1110 = -2 \\ \hline 10011 = 3 \end{array} \qquad \begin{array}{r} 1011 = -5 \\ +1110 = -2 \\ \hline 11001 = -7 \end{array} \qquad \begin{array}{r} 0111 = 7 \\ +0111 = 7 \\ \hline 1110 = \text{Overflow} \end{array} \qquad \begin{array}{r} 1010 = -6 \\ +1100 = -4 \\ \hline 10110 = \text{Overflow} \end{array}$$

Nota: El color azul indica el carry; El rojo indica qué es lo que produce overflow (cambio de signo).

Hagamos sumas en C_2 (límite de bits: 4)

$$\begin{array}{r} 0010 = 2 \\ +1001 = -7 \\ \hline 1011 = -5 \end{array} \qquad \begin{array}{r} 0101 = 5 \\ +1110 = -2 \\ \hline 10011 = Overflow \end{array} \qquad \begin{array}{r} 1011 = -5 \\ +1110 = -2 \\ \hline 11001 = Overflow \end{array} \qquad \begin{array}{r} 0111 = 7 \\ +0111 = 7 \\ \hline 1110 = \text{Overflow} \end{array} \qquad \begin{array}{r} 1010 = -6 \\ +1100 = -4 \\ \hline 10110 = \text{Overflow} \end{array}$$

Nota: Al tener un límite de 4 bits, en las sumas que tenemos carry terminamos teniendo overflow.

Rango de valores representables en n bits

Sean $n, m \in \mathbb{Z}$ decimos que el rango de representación en base n y m bits acepta el rango de valores de: $[-n^m, n^m - 1]$
¿Es posible representar el 1024 en binario y 4 bits? No.

- $2^4 = 16 \implies [-16, 15]$
- Pero, $1024 \notin [-16, 15]$
- Por lo tanto, 1024 no es representable en 4 bits.

Pasar número binario a decimal

Si tenemos el mismo número todo el tiempo podemos usar la serie geométrica

¿Qué número decimal representa el número $1111111111_{(2)}$?

$$\sum_{i=0}^{j-1} 1 \cdot n^i = \frac{q^{n+1} - 1}{q - 1}$$

$$\sum_{i=0}^{9} 1 \cdot 2^i = 2^{10} - 1$$

Si no tenemos el mismo número todo el tiempo podemos multiplicar cada dígito por la base y el exponente es la posición del bit.

$10_{(2)} = 1 * 2^1 + 0 * 2^0 = 2$