Clase 2.

1. Fracciones binarias y representación computacional de los números fraccionarios.

Al final de la clase anterior llegamos a que un número puede ser expresado como fracción binaria de la siguiente manera:

$$x = \pm(a_n 2^n + a_{n-1} 2^{n-1} + \ldots + a_1 2^1 + a_0 2^0 + b^1 2^{-1} + \ldots + b^m 2^{-m})$$

Por ejemplo:

$x = -101{,}011 \quad a_2 = 1, a_1 = 0, a_0 = 1, b_1 = 0, b_2 = 1, b_3 = 1$

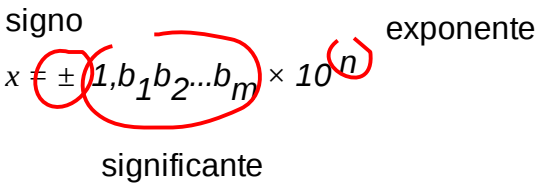
$x = 0{,}01101 \quad a_0 = 0, b_1 = 0, b_2 = 1, b_3 = 0, b_4 = 1$

Esta representación puede "condensarse" si extraemos la potencia de dos como factor común de manera tal que la expresión fraccionaria tiene a 1 como el único dígito binario en su parte entera:

$x = -1{,}01011 \times 2^2$

$x = 1{,}101 \times 2^{-2}$

Es decir que en general podemos escribir un número en fracción binaria como:



donde tenemos el signo (+ ó -), los dígitos propiamente dichos del número o "significante" (que como notamos recién siempre comienza en 1) y el exponente.

Esta representación es importante a efectos de entender como las computadoras almacenan los números llamados de "punto flotante", que es el término informático que denota a esta representación de los números fraccionarios. Básicamente, necesitamos almacenar el signo del número, su significante y el exponente. El término "punto flotante" denota el hecho de que la coma no está en una posición fija, sino que depende del exponente:

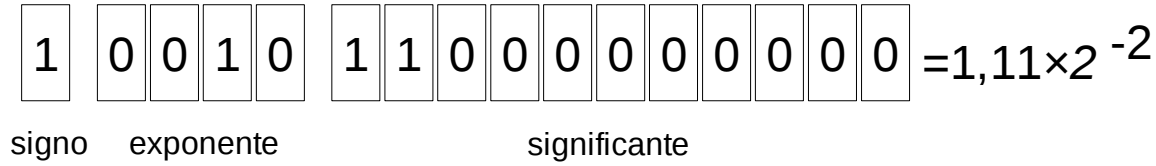
$x = -1{,}01011 \times 2^2 = -101{,}011$ (la coma aparece luego del tercer dígito)

$x = 1{,}01 \times 2^5 = 101000{,}0$ (la coma aparece luego del quinto dígito)

Un número de punto flotante tendrá asignado un número determinado de bits para su almacenamiento en la memoria de la computadora Por ejemplo: 16, 32 o 64 bits. Si tenemos 16 bits, entonces una posible manera de utilizar estos bits para almacenar las tres partes de un número fraccionario (signo, significante y exponente) son las siguientes:

El primer bit es destinado al signo, siguiento la convención 1 = + y 0 = -. Los 4 siguientes bits son utilizados para el exponente, con el primero de ellos indicando el signo. En ejemplo, el exponente es 3 (positivo), que en binario se escribe como 11, por lo que los 4 bits para el exponente son 1011 (el primer 1 indica que el exponente es positivo). Finalmente, el significante es 00101 (no hace falta

especificar el primer 1 porque siempre está presente). Otro número, por ejemplo:



Pueden contestar las siguientes preguntas y explicar el porqué:

a. Cual es el número positivo más grande que puede ser expresado en esta representación?

b. Cual es el número positivo más pequeño que puede ser expresado en esta representación?

2. Representación de un número en diferentes bases.

Consideremos las representaciones en binario, decimal y hexadecimal de un número entero positivo:

$$x_2 = a_n 2^n + a_{n-1} 2^{n-1} + \ldots + a_1 2^1 + a_0 2^0$$
$$x_{10} = b_m 10^m + b_{m-1} 10^{m-1} + \ldots + b_1 10^1 + b_0 10^0$$
$$x_{16} = c_r 16^r + c_{r-1} 16^{r-1} + \ldots + c_1 16^1 + c_0 16^0$$

Recordemos que la manera de obtener los dígitos binarios a a partir de la representación decimal es dividir progresivamente por 2 y utilizar lor restos de las divisiones como los dígitos binarios buscados.

a. Escribir un programa (en el lenguaje de elección con el que se sientan más familiares) que a partir de un número arbitrario en representación decimal, genere los dígitos de su representación hexadecimal.

b. Y otro programa que tome una representación binaria (una cadena de 1's y 0's) y genere el número correspondiente en decimal.