

ESTÁNDAR IEEE 754

Conversión de bases

Métodos numéricos

Universidad San Buenaventura Cali

ESTÁNDAR IEEE 754.

El estándar IEEE 754 o IEEE Standard for Binary Floating-Point Arithmetic, ha sido definido por el Instituto de Ingenieros Eléctricos y Electrónicos (Institute of Electrical and Electronics Engineers, IEEE).

Es un estándar para las computaciones en coma flotante. especifica cuatro formatos para la representación de valores en coma flotante de los cuales mostraremos dos en este documento los cuales son: precisión simple (32 bits), precisión doble (64 bits).

CONVERSIÓN DE NUMERO EN BASE DECIMAL EN UNA MEMORIA DE 32 BITS.

Este formato se almacena en una palabra de 32 Bits organizándose de la siguiente manera:



- El **signo** del número almacenado donde puede ser:
 - 0= positivo
 - 1= negativo.
- El **exponente** se guarda en el espacio de 8 Bits.
- La **mantisa** se guarda en el espacio de 23 Bits.

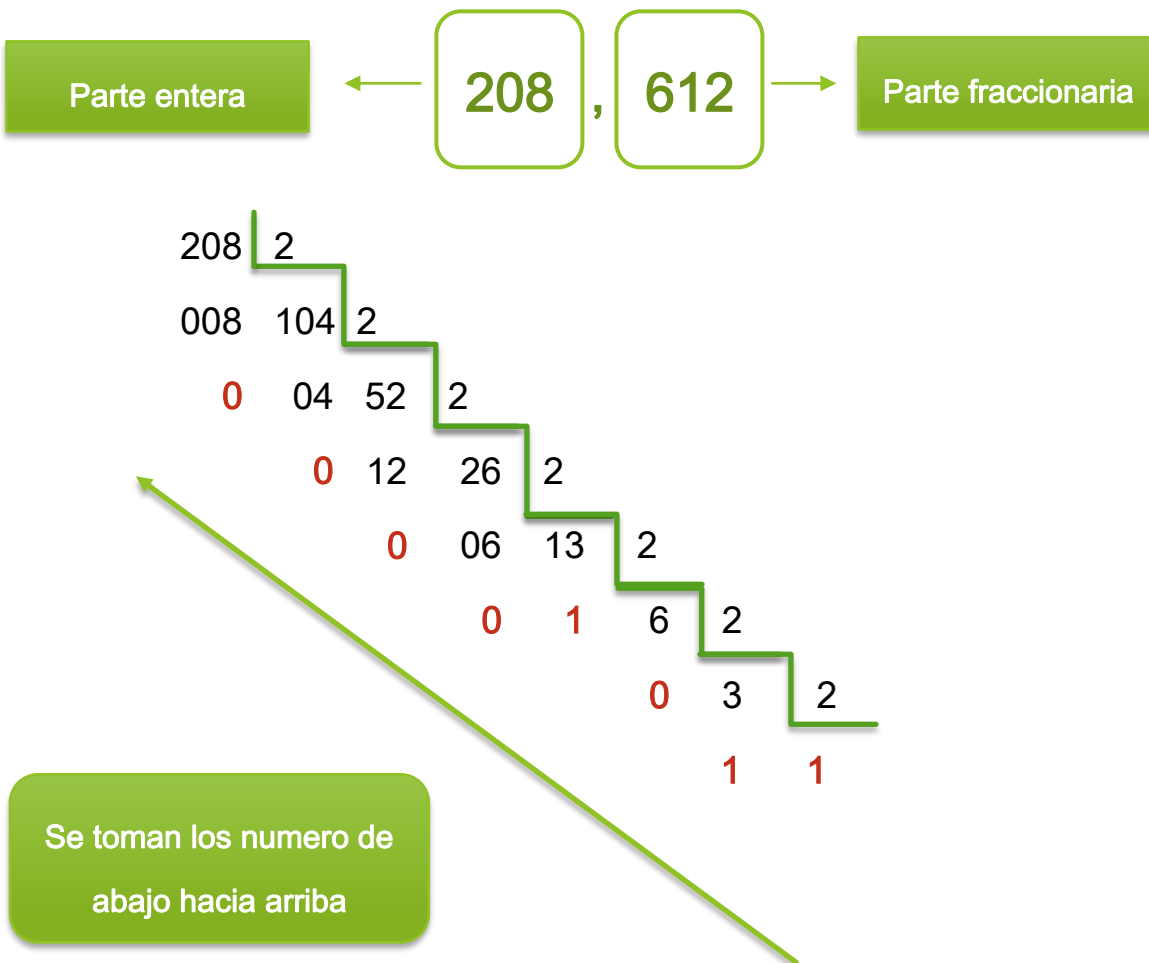
Ejemplo:

Escribir el número 208,612 en una memoria de 32 bits de acuerdo al estándar IEEE-754.

Solución:

1. El número 208,612 es positivo, por lo cual el signo en el primer bit será 0.
2. Convertiremos el número a binario:

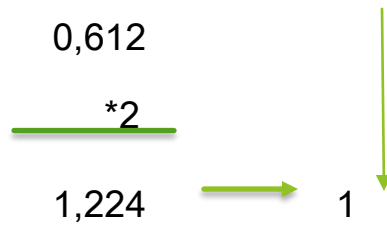
1. primero tomamos la parte entera del número y la pasamos a binaria:



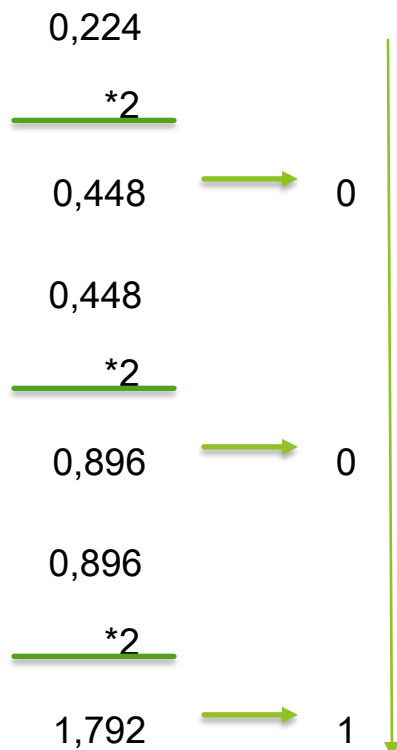
Así tenemos que el número 208 en binarios es $11010000_{(2)}$

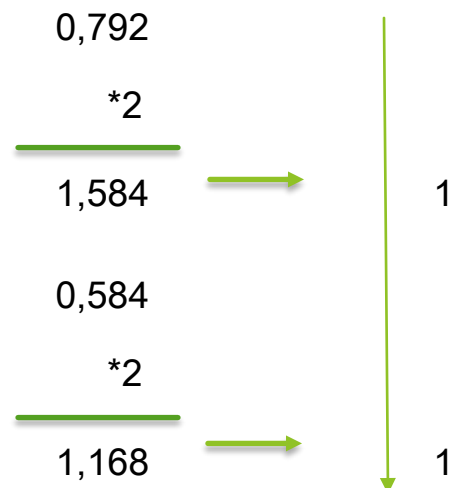
2. Ahora transformamos la parte fraccionaria 0,612:

Multiplicamos la parte fraccionaria y del resultado guardamos el número entero

$$\begin{array}{r} 0,612 \\ \times 2 \\ \hline 1,224 \end{array} \rightarrow 1$$


Después le restamos al resultado el número entero y volvemos a operar. La cantidad de operaciones son dependiendo de cuantas decimas sean necesarias.

$$\begin{array}{r} 0,224 \\ \times 2 \\ \hline 0,448 \end{array} \rightarrow 0$$
$$\begin{array}{r} 0,448 \\ \times 2 \\ \hline 0,896 \end{array} \rightarrow 0$$
$$\begin{array}{r} 0,896 \\ \times 2 \\ \hline 1,792 \end{array} \rightarrow 1$$


$$\begin{array}{r} 0,792 \\ \times 2 \\ \hline 1,584 \end{array} \rightarrow 1$$
$$\begin{array}{r} 0,584 \\ \times 2 \\ \hline 1,168 \end{array} \rightarrow 1$$


Así tenemos que el número 0,612 en binarios es $0,1001110_{(2)}$

Ahora que sabemos que el número $208,612 = 11010000,1001110_{(2)}$

podemos empezar a convertirlo en la memoria de 32 Bits con los siguientes pasos:

1. Se normaliza el numero:

- Se corre la coma hasta tener un solo 1 a la izquierda de la coma.
Se cuentan los números que se desplazaron después de la coma,
y esta cantidad es el exponente al que se eleva la base.

$$\begin{array}{c} 11010000,1001110_{(2)} \\ \text{↑↑↑↑↑} \\ 7 \ 6 \ 5 \ 4 \ 3 \ 2 \ 1 \end{array}$$
$$1, \underbrace{10100001001110}_{\text{Mantisa}} \times 2^7$$

2. Después se transforma el exponente sumándole 127 (el cual es un número fijo) al exponente el cual en este caso es 7:

$$127 + 7 = \underline{\underline{134}}$$

los números que van después de la coma serian la mantisa y se llenaria con ceros para completar los 23 Bits

Se registra así:

1 Bit	8 Bits	23 Bits
0	10000110	10100001001110000000000

CONVERSIÓN DE NUMERO EN BASE DECIMAL EN UNA MEMORIA DE 64 BITS.

Este formato se almacena en una palabra de 64 Bits organizándose de la siguiente manera:



Como en el anterior:

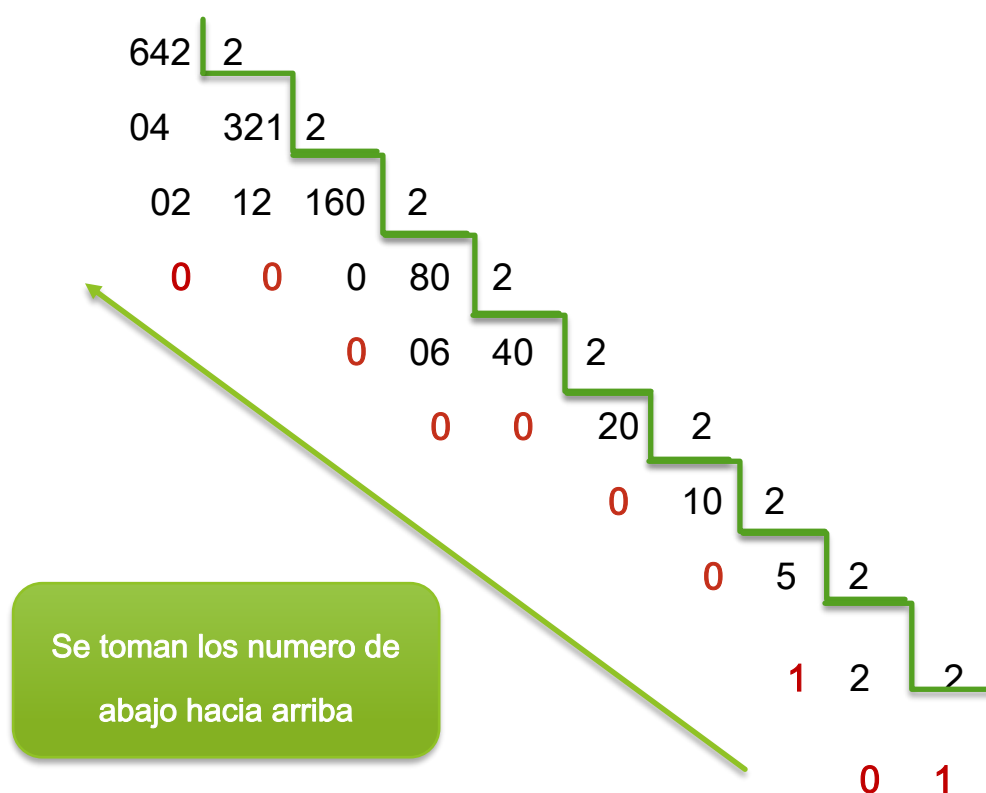
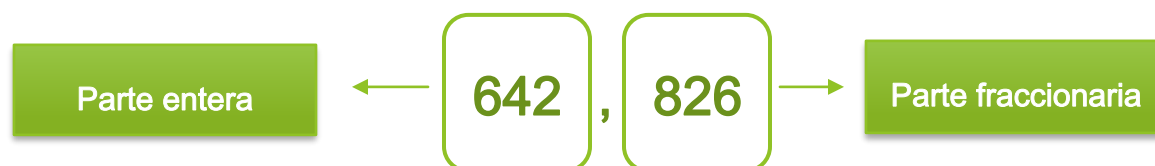
- el **signo**:
 - 0= positivo
 - 1= negativo.
- El **exponente** se guarda en el espacio de 8 Bits.
- La **mantisa** se guarda en el espacio de 23 Bits.

Ejemplo:

Escribir el número -642,826 en una memoria de 64 bits de acuerdo al estándar IEEE-754.

3. El número 642,826 es negativo, por lo cual el signo en el primer bit será 0.
4. Convertiremos el número a binario:

3. primero tomamos la parte entera del número y la pasamos a binaria:



Así tenemos que el número 642 en binarios es $1010000010_{(2)}$

4. Ahora transformamos la parte fraccionaria 0,826:

Multiplicamos la parte fraccionaria y del resultado guardamos el número entero:

0,826

 *2

1,652  1

0,652

 *2

1,304  1

0,304

 *2

0,608  0

0,608

 *2

1,216  1

0,216

 *2

0,432  0

Ahora que sabemos que el número 642,826 en binarios es 1010000010,11010⁽²⁾ podemos empezar a convertirlo en la memoria de 64 Bits con los siguientes pasos:

$$1, \underbrace{01000001011010}_{\text{Mantisa}} \times 2^9$$
$$1023 + 9 = \underline{1032}$$
[illegible]