Números fraccionarios

Números fraccionarios

Así como hemos visto distintas formas de representar números en $\mathbb{Z}$, veremos las **principales formas de representar números en** $\mathbb{R}$.

- Notación de **punto fijo (fixed point) Signo-Magnitud**.
- Notación de **punto fijo con Complemento a la Base**.
- Estándar **IEEE 754 de punto flotante (floating point)**.

Punto fijo (no signado)

Como habíamos visto antes, al almacenar un número en memoria, la computadora no puede representar el punto decimal.

El formato de **punto fijo** soluciona el problema utilizando **cantidades fijas de bits** tanto para la **parte entera** como para la **parte fraccionaria** del número a representar.

Por ejemplo, si utilizamos 8 bits para cada parte, 73, 3203125 se representa como:

Exponente:	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8
Bit:	0	1	0	0	1	0	0	1	0	1	0	1	0	0	1	0

$2^6 + 2^3 + 2^0 + 2^{-2} + 2^{-4} + 2^{-7} = 64 + 8 + 1 + 0{,}25 + 0{,}0625 + 0.0078125 =$
$73{,}3203125$

Punto fijo (no signado)

Ya vimos antes el proceso para convertir la parte entera de decimal a binario. Para convertir la **parte fraccionaria**, debemos **realizar sucesivas multiplicaciones por dos**, y **de cada multiplicación extraer la parte entera como un bit del resultado**, y realizar la siguiente iteración **solo con la parte fraccionaria** del mismo.

0,3203125	x 2 = **0**,640625
0,640625	x 2 = **1**,28125
0,28125	x 2 = **0**,5625
0,5625	x 2 = **1**,125
0,125	x 2 = **0**,25
0,25	x 2 = **0**,5
0,5	x 2 = **1**

Punto fijo (no signado)

El algoritmo **termina cuando sucede alguna de las siguientes**:

- Obtenemos **1 como resultado** (sin decimales). El resultado **es exacto**.
- Realizamos **tantos pasos como bits decimales tenemos disponibles**. El resultado **no es exacto** pues está truncado.

Finalmente, los **bits de la parte fraccionaria** estarán conformados por las **partes enteras de los resultados** anteriores, **en el orden en el que se obtuvieron**.

Entonces, $73{,}3203125_{10} = 01001001{,}01010010_2$.

Punto fijo (signado)

Para **representar números con signo**, podemos hacer **lo mismo que hacíamos con números enteros**:

- **Signo-Magnitud: reservamos el MSb de la parte entera para el signo**. Seguimos teniendo doble representación del cero y dificultades para las operaciones aritméticas.
- **Complemento a la dos: expresamos el valor absoluto** del número como si se tratase de uno no signado, y luego **calculamos el complemento a dos** de la misma manera que lo hacíamos con los números enteros, **“olvidándonos de la coma”**.

Ejercicio 8

Expresar el número -73,3203125 utilizando Signo-Magnitud y Complemento a la Base, utilizando 8 bits para ambas partes.

Solución

Signo-Magnitud:

11001001,01010010

Complemento a dos:

01001001,01010010

↓

10110110,1010111O

Punto flotante

El sistema de punto flotante es **muy similar a la notación científica**, donde **la coma no está** en una posición **fija**, sino que ésta **depende del exponente** que se utilice.

Ejemplos:

- El número 135351,98 se expresa como $1,3535198 \times 10^{5}$.
- El número 0,000000000912 se expresa como $9,12 \times 10^{-10}$.

Es importante notar que la **parte entera** de un número en notación científica debe ser de **un solo dígito**, el cual **no puede ser cero**.

Por ejemplo, $0,9142 \times 10^{4}$ y $91,42 \times 10^{2}$ son representaciones incorrectas del número $9142 = 9,142 \times 10^{3}$.

Punto flotante

Es el **sistema más utilizado actualmente** ya que ofrece muchas **ventajas frente al de punto fijo**:

- Permite **representar números muy pequeños y muy grandes**.
- La representación es muy **compacta y eficiente**.
- **Permite** realizar **operaciones aritméticas**.
- Tiene una **distribución** más **uniforme** de números fraccionarios.

IEEE 754 32 bits

El formato IEEE 754 es el estándar de punto flotante.

Este formato consiste en **1 bit de signo (S)**, un **exponente (E)** y una **mantisa (M)**, cada uno de los cuales explicaremos a continuación.

El número que representamos se obtiene mediante:

$$N = (-1)^S \times 1.M \times 2^{E-127}$$

Signo (1 bit)	Exponente (8 bits)	Mantisa (23 bits)

Mantisa

La mantisa representa la **parte fraccionaria del número expresado en notación científica**. Debido a que **la parte entera no puede ser cero** y a que estamos trabajando en binario, la misma **es, indefectiblemente, un uno**. Por esto, **obviamos dicho uno**, y utilizamos los 23 bits de la **mantisa únicamente para los decimales**.

Por ejemplo, para representar el número -73,3203125, primero lo representamos en punto fijo no signado: 1001001,0101001 ; y luego en notación científica: **1,0010010101001 2^6**. Finalmente, la mantisa será: 0010 0101 0100 1000 0000 000 (**rellenamos con ceros**).

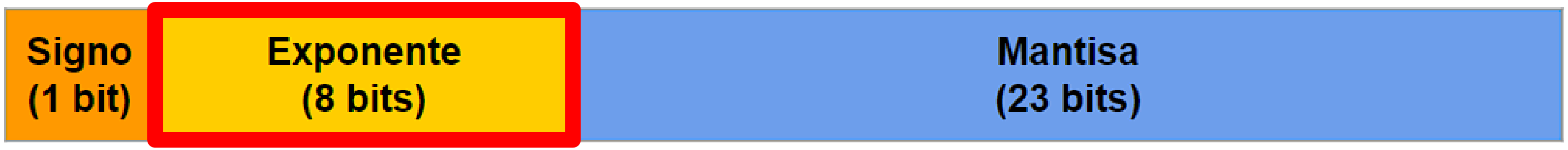
Signo (1 bit)	Exponente (8 bits)	Mantisa (23 bits)

Exponente (sesgo 127)

El exponente es un número **entero no signado** que **indica la magnitud del número** representado. En muchas oportunidades, es posible **comparar dos números** en formato IEEE 754 simplemente **comparando los exponentes** (si éstos fueran iguales, se debe comparar la mantisa).

Debido a que es no signado, el número **presenta un "sesgo"**, está **incrementado en 127**, para poder contemplar exponentes negativos. Por ejemplo, si fuera -6, en este campo se coloca 121.

De esta forma, podemos tener **exponentes entre -126 y 127**. Los dos faltantes (0000 0000 y 1111 1111) se usan para casos especiales.



Exponente (sesgo 127)

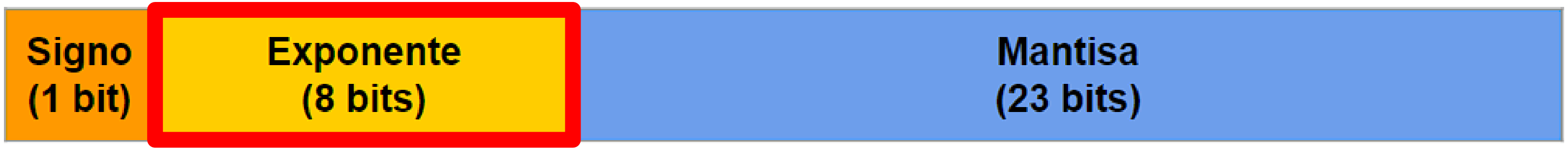
En el ejemplo anterior, habíamos expresado el número a representar en **punto fijo no signado**, y **luego en notación científica**.

$$001001{,}0101001 = 1{,}0010010101001\ 2^{6}$$

De la representación en notación científica, obtenemos el exponente, que en este caso es 6.

Luego, **le sumamos 127**, quedando 133, **y lo expresamos en binario como un entero no signado**. Entonces, en el campo del exponente colocaremos:

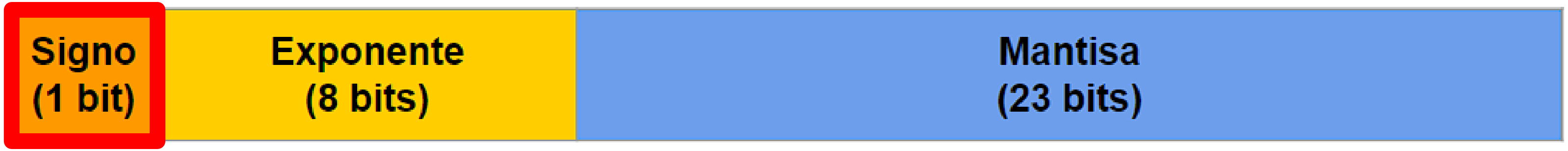
1000 0101



Signo

El bit de signo es igual que en los demás sistemas que vimos: **0 si el número es positivo, o 1 si es negativo**.

En este caso de ejemplo, el número es negativo, por lo que el bit de signo será 1.



Conversión decimal a IEEE754

Con los pasos anteriores ya hechos, juntamos todos los datos en 32 bits, y luego abreviamos el binario en hexadecimal.

SIGNO EXPONENTE MANTISA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	1	0	0	0	0	1	0	1	0	0	1	0	0	1	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0

Finalmente, abreviado en hexadecimal, queda **0xC292A400** (–73,3203125).

Casos especiales

Existen 4 casos especiales en el estándar IEEE 754:

- **NaN (Not a Number).**
  Exponente 1111 1111, mantisa ≠ 0.
- **Infinito.**
  Exponente 1111 1111, mantisa = 0. Tiene signo.
- **Números subnormales / underflow.**
  Exponente 0000 0000, mantisa ≠ 0.
- **Cero.**
  Exponente 0000 0000, mantisa = 0.

Ver que como el número está expresado en notación científica, **el uno** que obviamos **en la parte entera impediría representar el cero** de la forma en la que representamos cualquier otro número.

Ejercicio 9

Convertir los siguientes números fraccionarios expresados en base decimal al formato IEEE 754 de 32 bits. Dar el resultado abreviado en hexadecimal.

- -3,125
- 14,0625
- -37,75
- -103,828125

Solución

- -3.125 => 0xC0480000
- 14,0625 => 0x41610000
- -37,75 => 0xC2170000
- -103,828125 => 0xC2CFA800

Conversión IEEE754 a decimal

Para realizar la **conversión inversa**, lo que debemos hacer es **interpretar el binario** en formato IEEE 754 para **separar la información de cada una de sus partes**. Tomemos el mismo ejemplo:

Lo primero que notamos, del **MSb**, es que el número es **negativo**.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	1	0	0	0	0	1	0	1	0	0	1	0	0	1	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0

Conversión IEEE754 a decimal

De los bits 30 a 23 inclusive, obtenemos el exponente. Recordemos que está incrementado en 127, por lo que debemos **restarle 127 para obtener el verdadero exponente de la notación científica**.

En este caso, en el **campo del exponente** tenemos 1000 0101. Recordemos que es **no signado**, por lo que representa el 133.

Al restarle 127 obtenemos el exponente real: 133 – 127 = 6.

30	29	28	27	26	25	24	23
1	0	0	0	0	1	0	1

Conversión IEEE754 a decimal

De los **bits 22 al 0 inclusive**, obtenemos la **mantisa**. Recordar que **se trata únicamente de la parte fraccionaria**, por lo que debemos **agregarle un 1 de parte entera** antes de la coma.

Podemos **obviar los ceros que están demás en la parte menos significativa**, marcados con rojo. Entonces tenemos:

Mantisa: **0010010101001**

Finalmente tenemos: **1,0010010101001**

22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	1	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0

Conversión IEEE754 a decimal

Juntando toda la información que obtuvimos, formamos el número en notación científica:

$$-1{,}0010010101001 \times 2^{6}$$

En punto fijo, esto equivale a:

$$-1001001{,}0101001$$

Convertimos el número a decimal como vimos antes:

$$2^{6} + 2^{3} + 2^{0} + 2^{-2} + 2^{-4} + 2^{-7} =$$

$$= 64 + 8 + 1 + 0{,}25 + 0{,}0625 + 0.0078125 =$$

$$= 73{,}3203125$$

Y, como en el primer paso vimos que el número era negativo:

$$= \mathbf{-73{,}3203125}$$

Ejercicio 10

Convertir los siguientes números expresados según el estándar IEEE 754 de 32 bits a base decimal.

- 0x44BF5000
- 0xC2FC0100
- 0x00000000
- 0x3C000000

Solución

- 1530.5
- -126.001953125
- 0
- 0.0078125