# Organización de Computadoras 2014

## *Apunte 2: Sistemas de Numeración: Punto Flotante*

La coma o punto flotante surge de la necesidad de representar números reales y enteros con un rango de representación mayor que el que nos ofrece punto fijo. En la representación en coma flotante, se dividen los n bits disponibles para representar un dato, en 2 partes llamadas mantisa M y exponente E. Considerando que la mantisa tiene una longitud de p bits y que el exponente la tiene de q bits, se cumple que n = p + q.

La mantisa contiene los dígitos significativos del dato, en tanto que el exponente indica el factor de escala, en forma de una potencia de base r. Por todo ello, el valor del número viene dado por:

$$V(X) = M * r^E$$

El número X viene representado por la cadena:

$$X = (m_{p-1}, ..., m_1, m_{0,} e_{q-1}, ... , e_1, e_0)$$

Cuya representación gráfica es:

| Mantisa (p bits) | Exponente (q bits) |
|---|---|
| n-1 | q q-1 0 |

La mantisa M y el exponente E se representan en alguno de los sistemas de punto fijo.

El rango de representación estará dado por:

Mínimo número negativo: $-(\text{mantisa máxima}) * \text{base}^{\text{máximo exponente positivo}}$
Máximo número negativo: $-(\text{mantisa mínima}) * \text{base}^{\text{máximo exponente negativo}}$
Mínimo número positivo: $(\text{mantisa mínima}) * \text{base}^{\text{máximo exponente negativo}}$
Máximo número positivo: $(\text{mantisa máxima}) * \text{base}^{\text{máximo exponente positivo}}$

Ejemplo:

Supongamos que tenemos un sistema de numeración con las siguientes características:
Mantisa de 5 bits, expresada en BSS.
Exponente de 3 bits, expresada en BCS.
Base 2.
Entonces, la cadena 10110 101 representará a $22 * 2^{-1} = 11$

Notar que, si la mantisa M representa el valor 0, el valor del número representado será el 0 independientemente del valor de exponente que se disponga.
Para el ejemplo anterior la cadena
00000 001 representará el valor $0* 2^1 = 0$
00000 101 representará el valor $0* 2^{-1} = 0$
00000 **eee** representará el valor $0* 2^e = 0$ para cualquier valor de **e**

O sea poseeremos tantas representaciones del valor 0 como exponentes distintos dispongamos.

### Mantisa fraccionaria

En este caso, la mantisa en lugar de interpretarse como un número entero, se toma como un número real con el punto decimal implícito a la izquierda de sus bits.

Siguiendo con el ejemplo anterior de mantisa de 5 bits en BSS, exponente de 3 bits en BCS y base 2.

Entonces, la cadena 10110 101
representará a $(1/2 + 1/8 + 1/16) * 2^{-1} = 0.6875 * 2^{-1} = 0{,}34375$

y la cadena 00101 011
representará a $(1/8 + 1/32) * 2^3 = (5/32) * 2^3 = 5/4 = 1{,}25$

este valor coincide con el de la cadena
01010 010 => (1/4 + 1/16) * $2^2$ = (5/16) * $2^2$ = 1,25

y con el de la cadena
10100 001 => (1/2 + 1/8) $2^1$ = (5/8) * $2^1$ = 1,25

Para evitar tener múltiples representaciones de un mismo número, se puede representar la mantisa en forma ***normalizada***. La normalización consiste en hacer que su primer dígito tenga el valor 1. De esta manera, un número no puede tener mas de una representación.

En el ejemplo anterior, en un sistema con mantisa normalizada, de las 3 representaciones posibles para el número 1,25 solo sería válida 10100 001, ya que en ella la mantisa empieza con un 1.

La desventaja de este sistema de representación es que no se puede representar el número 0.

Dado que las mantisas normalizadas siempre empiezan con un 1, podemos no almacenar este dígito, con lo cual la mantisa tiene un bit de mas para almacenar. Por lo tanto,

$$p + q = n + 1$$

Este bit implícito se agrega en el momento de operar. A este sistema se lo llama ***mantisa fraccionaria normalizada con bit implicito.***

Dado que no siempre podemos representar exactamente el número que queremos, definimos *error absoluto* y *error relativo* de un número en un sistema de la siguiente forma:

$$EA(x) = | x' - x |$$
$$ER(x) = EA(x) / x$$

donde x' es el número representable del sistema más próximo a x.

Para que los límites de las distintas formas de representación en punto flotante sean comparables utilizaremos como "mantisa mínima" el valor de mantisa M distinta de cero mas pequeña.

## Ejercicios

***1. Mantisa entera en BSS de 8 bits.***
***Exponente en Ca2 en 4 bits.***
***Donde los primeros 8 bits desde la izquierda representan la mantisa, y los siguientes 4 al exponente.***

Queremos obtener:
a) La representación del número 0,40625:
0,40625 * (2 * 2 * 2 * 2 * 2) = 13
0,40625 * $2^5$ = 13
0,40625 = 13/$2^5$
0,40625 = 13 * $2^{-5}$
0,40625 => 00001101 1011

b) El número máximo:
mayor mantisa por mayor exponente: 11111111 0111 => 255 * $2^7$ = 32640

c) El número mínimo (con mantisa distinta de cero):
menor mantisa por menor exponente: 00000001 1000 => 1 * $2^{-8}$ = $2^{-8}$ = 1/256

d ) Resolución en el máximo:
mayor número representable: 11111111 0111 => 32640
anterior número representable: 11111110 0111 => 32512
diferencia: = 128

e ) Resolución en el mínimo:
menor número representable (M≠0): 00000001 1000 => 1/256
siguiente número representable: 00000010 1000 => 2/256
diferencia: = 1/256

***2. Mantisa fraccionaria en BCS de 6 bits.***
***Exponente en Ex2 en 3 bits.***
***Donde el primer bit de la izquierdarepresenta el signo, los 5 siguientes representan la mantisa, y los siguientes 3 al exponente***.

Queremos obtener:
a) La representación del número 4,75:
4,75 * (2 * 2) = 19
4,75 = 19 / 4
tenemos que lograr que la mantisa sea una fracción, por tanto, multiplicamos y dividimos por 8
4,75 = (19 * 8)/(4 * 8)
reagrupando obtenemos
4,75 = (19 / 32) * 8
4,75 = (19 / 32 )* $2^3$
4,75 => 010011 111

b) El número máximo positivo:
mayor mantisa positiva por mayor exponente positivo: 011111 111 => (31/32) * $2^3$ = 7,75

c) El número mínimo positivo (distinto de 0):
menor mantisa positiva por mayor exponente negativo: 000001 000 => (1/32) * $2^{-4}$ = $2^{-5}$ * $2^{-4}$ = $2^{-9}$

d) El número máximo negativo (distinto de 0):
menor mantisa negativa por menor exponente negativo: 100001 000 => -(1/32) * $2^{-4}$ = $-2^{-5}$ * $2^{-4}$ = $-2^{-9}$

e) El número mínimo negativo:
mayor mantisa negativa por mayor exponente positivo: 111111 111 => -(31/32) * $2^3$ = -7,75

f ) Resolución en el máximo positivo:
mayor número representable: 011111 111 => (31/32) * $2^3$ = 7,75
anterior número representable: 011110 111 => (30/32) * $2^3$ = 7,50
diferencia: = 0,25

g ) Resolución en el mínimo positivo:
menor número representable positivo (M≠0): 000001 000 => (1/32) * $2^{-4}$
siguiente número representable: 000010 000 => (2/32) * $2^{-4}$
diferencia: = (1/32) * $2^{-4}$ = $2^{-9}$

h ) Resolución en el máximo negativo:
menor número representable negativo (M≠0): 100001 000=> -(1/32) * $2^{-4}$
siguiente número representable: 100010 000=> -(2/32) * $2^{-4}$
diferencia: = (1/32) * $2^{-4}$ = $2^{-9}$

i ) Resolución en el mínimo negativo:
mayor número representable negativo: 111111 111 => (31/32) * $2^3$ = -7,75
anterior número representable: 111110 111 => (30/32) * $2^3$ = -7,50
diferencia: = 0,25

***3. Mantisa fraccionaria normalizada en BCS de 6 bits.***
***Exponente en Ex2 en 3 bits.***
***Donde el primer bit de la izquierda representa el signo, los 5 siguientes representan la mantisa, y los siguientes 3 al exponente.***

Queremos obtener:
a) La representación del número -0,140625:
-0,140625 * (2 * 2 * 2 * 2 * 2 * 2) = -9
-0,140625 * $2^6$ = -9
-0,140625 = -9 / $2^6$
-0,140625 = (-9 / $2^{5)}$ * $2^{-1}$
-0,140625 => 101001 011
como no está normalizada la mantisa, hacemos un corrimiento
-0,140625 => 101001 011 = 110010 010

b) El número máximo positivo:
mayor mantisa positiva por mayor exponente positivo: 011111 111 => (31/32) * $2^3$ = 7,75

c) El número mínimo positivo (distinto de 0):
menor mantisa positiva por mayor exponente negativo: 010000 000 => (1/2) * $2^{-4}$ = 1/32

d) El número máximo negativo (distinto de 0):
menor mantisa negativa por menor exponente negativo: 110000 000 => -(1/2) * $2^{-4}$ = -1/32

e) El número mínimo negativo:
mayor mantisa negativa por mayor exponente positivo: 111111 111 => -(31/32) * $2^3$ = -7,75

f ) Resolución en el máximo positivo:
mayor número representable positivo: 011111 111 => 7,75
anterior número representable: 011110 111 => 7,5
diferencia: = 0,25

g ) Resolución en el mínimo positivo:
menor número representable positivo (M≠0): 010000 000 => (1/2) * $2^{-4}$
siguiente número representable: 010001 000 => (17/32) * $2^{-4}$
diferencia: = (1/32) * $2^{-4}$ = $2^{-9}$

h ) Resolución en el máximo negativo:
menor número representable negativo (M≠0): 110000 000 => -(1/2) * $2^{-4}$
siguiente número representable: 110001 000 => -(17/32) * $2^{-4}$
diferencia: = (1/32) * $2^{-4}$ = $2^{-9}$

i ) Resolución en el mínimo negativo:
mayor número representable negativo: 111111 111 => -7,75
anterior número representable: 111110 111 => -7,5
diferencia: = 0,25

***4. Mantisa fraccionaria normalizada con bit implícito en BCS de 6 bits.***
***Exponente en Ex2 en 3 bits.***
***Donde el primer bit de la izquierda representa el signo, los 5 siguientes representan la mantisa, y los siguientes 3 al exponente.***

Queremos obtener:
a) La representación del número 1,96875:
1,96875 * (2 * 2 * 2 * 2 * 2) = 63
1,96875 * $2^5$ = 63
1,96875 = 63 / $2^5$
1,96875 = (63 / $2^5$) * $2^0$ = (63 / $2^6$) * $2^1$
1,96875 => 011111 101

b) El número máximo positivo:
mayor mantisa positiva por mayor exponente positivo: 011111 111 => (63/64) * $2^3$ = 7,875

c) El número mínimo positivo:
menor mantisa positiva por mayor exponente negativo: 000000 000 => (1/2) * $2^{-4}$ = 1/32

d) El número máximo negativo:
menor mantisa negativa por menor exponente negativo: 100000 000 => -(1/2) * $2^{-4}$ = -1/32

e) El número mínimo negativo:
mayor mantisa negativa por mayor exponente positivo: 111111 111 => -(63/64) * $2^3$ = -7,875

f ) Resolución en el máximo positivo:
mayor número representable positivo: 011111 111 => 7,875
anterior número representable: 011110 111 => 7,75
diferencia: = 0,125

g ) Resolución en el mínimo positivo:

menor número representable positivo: 000000 000 => $(1/2) * 2^{-4}$
siguiente número representable: 000001 000 => $(33/64) * 2^{-4}$
diferencia: $= (1/64) * 2^{-4} = 2^{-10}$

h ) Resolución en el máximo negativo:

menor número representable negativo: 100000 000 => -1/32
siguiente número representable: 100001 000 => $-(33/64) * 2^{-4}$
diferencia: $= (1/64) * 2^{-4} = 2^{-10}$

i) Resolución en el mínimo negativo:

mayor número representable negativo: 111111 111 => -7,875
anterior número representable: 111110 111 => -7,75
diferencia: = 0,125

***5. Mantisa fraccionaria entera en Ca1 de 8 bits.***
***Exponente en BSS en 4 bits.***
***Donde los primeros 8 bits representan la mantisa, y los siguientes 4 al exponente.***

Dadas las siguientes representaciones:

p = 00011010 1101
q = 00001001 1111

Queremos hallar el resultado de la suma de las mismas.

Para esto, tenemos que igualar los exponentes. Dado que el exponente de p es menor que el del q, o bien subimos el exponente de p, o bien bajamos el de q. Subir el exponente de un número implica hacer un corrimiento a derecha de la mantisa; bajar el exponente implica hacer un corrimiento a izquierda de la mantisa.

- Subir el exponente de p:

Subiendo en 1 el exponente de p obtenemos: 00001101 1110
Todavía el exponente sigue siendo menor que el de q, con lo cual tendríamos que hacer otro corrimiento. Pero dado que el ultimo bit de la mantisa es un 1, no lo podemos hacer.

- Bajar el exponente de q:

Bajando en 1 el exponente de q obtenemos: 00010010 1110
Todavía el exponente sigue siendo mayor que el de p, con lo cual tendríamos que hacer otro corrimiento.
Bajando en 1 el exponente obtenemos: 00100100 1101
Con lo cual conseguimos igualar los exponentes.

Ahora realizamos la suma de las mantisas:

00011010
<u>00100100</u>
00111110

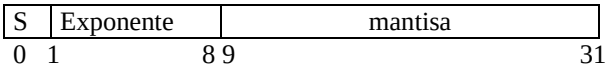
Por tanto, el resultado de la suma es: 00111110 1101

# IEEE 754

El IEEE 754 es un estándar de aritmética en coma flotante. Este estándar especifica como deben representarse los números en coma flotante con simple precisión (32 bits) o doble precisión (64 bits), y también cómo deben realizarse las operaciones aritméticas con ellos.

Emplea mantisa fraccionaria, normalizada y en representación signo magnitud, sin almacenar el primer dígito, que es igual a 1. El exponente se representa en exceso, que en este caso no se toma como $2^{n-1}$, sino como $2^{n-1}$ -1

**Simple Precisión**
El estándar IEEE-754 para la representación en simple precisión de números en coma flotante exige una cadena de 32 bits. El primer bit es el bit de signo (S), los siguientes 8 son los bits del exponente (E) y los restantes 23 son la mantisa (M):




En los formatos IEEE no todas lasa cadenas se interpretan de la misma manera. Algunas combinaciones se utilizan para representar valores especiales.

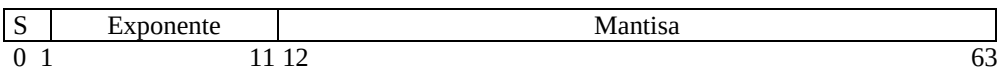
El valor V representado por esta cadena puede ser determinado como sigue:

| Exponente | Mantisa | Signo | Valor |
|---|---|---|---|
| 0 < E < 255 | Cualquiera | 1 | $-1 * 2^{E-127} * 1.M$ donde "1.M" se emplea para representar el número binario creado por la anteposición a M de un 1 y un punto binario |
| 0 < E < 255 | Cualquiera | 0 | $2^{E-127} * 1.M$ donde "1.M" se emplea para representar el número binario creado por la anteposición a M de un 1 y un punto binario |
| 255 | No nulo | Cualq. | NaN ("Not a number"). Se aplica para señalar varias condiciones de error. |
| 255 | 0 | 1 | -Infinito |
| 255 | 0 | 0 | Infinito |
| 0 | No nulo | 1 | $-(2^{-126})* (0.M)$ (Números sin normalizar) |
| 0 | No nulo | 0 | $2^{-126} * (0.M)$ (Números sin normalizar) |
| 0 | 0 | 1 | -0 |
| 0 | 0 | 0 | 0 |

En particular,

```
0 00000000 00000000000000000000000 => 0
1 00000000 00000000000000000000000 => -0

0 11111111 00000000000000000000000 => Infinito
1 11111111 00000000000000000000000 => -Infinito

0 11111111 00000100000000000000000 => NaN
1 11111111 00100010001001010101010 => NaN

0 10000000 00000000000000000000000 => 2^(128-127) * 1.0 = 2
0 10000001 10100000000000000000000 => 2^(129-127) * 1.101 = 6,5
1 10000001 10100000000000000000000 => -1 * 2^(129-127) * 1.101 = -6,5

0 00000001 00000000000000000000000 => 2^(1-127) * 1.0 = 2^(-126)
0 00000000 10000000000000000000000 => 2^(-126) * 0.1  = 2^(-127)
```

**Doble precisión**
El estándar IEEE-754 para la representación en doble precisión de números en coma flotante exige una cadena de 64 bits. El primer bit es el bit de signo (S), los siguientes 11 son los bits del exponente (E) y los restantes 52 son la mantisa (M):




El valor V representado por esta cadena puede ser determinado como sigue:

| Exponente | Mantisa | Signo | Valor |
|---|---|---|---|
| 0 < E < 2047 | Cualquiera | 1 | $-1 * 2^{E-1023} * 1.M$ donde "1.M" se emplea para representar el número binario creado por la anteposición a M de un 1 y un punto binario |
| 0 < E < 2047 | Cualquiera | 0 | $2^{E-1023} * 1.M$ donde "1.M" se emplea para representar el número binario creado por la anteposición a M de un 1 y un punto binario |
| 2047 | No nulo | Cualq. | NaN ("Not a number"). Se aplica para señalar varias condiciones de error. |
| 2047 | 0 | 1 | -Infinito |
| 2047 | 0 | 0 | Infinito |
| 0 | No nulo | 1 | $-(2^{-1022})* (0.M)$ (Números sin normalizar) |
| 0 | No nulo | 0 | $2^{-1022} * (0.M)$ (Números sin normalizar) |
| 0 | 0 | 1 | -0 |
| 0 | 0 | 0 | 0 |

En particular,

```
0 00000000000 0000000000000000000000000000000000000000000000000000 => 0
1 00000000000 0000000000000000000000000000000000000000000000000000 => -0

0 11111111111 0000000000000000000000000000000000000000000000000000 => Infinito

1 11111111111 0000000000000000000000000000000000000000000000000000 => -Infinito

0 11111111111 0000010000000000000000000000000000000000000000000000 => NaN
1 11111111111 0010001000100101010101000000000000000000000000000000 => NaN

0 10000000000 0000000000000000000000000000000000000000000000000000
                                                    => 2^(1024-1023) * 1.0 = 2
0 10000000001 1010000000000000000000000000000000000000000000000000
                                                    => 2^(1025-1023) * 1.101 = 6,5
1 10000000001 1010000000000000000000000000000000000000000000000000
                                                => -1 * 2^(1025-1023) * 1.101 = -6,5
0 00000000001 0000000000000000000000000000000000000000000000000000
                                                     => 2^(1-1023) * 1.0 = 2^(-1022)
0 00000000000 1000000000000000000000000000000000000000000000000000
                                                     => 2^(-1022) * 0.1 = 2^(-1023)
```