La velocidad no lleva a ninguna parte si no se va en la dirección correcta.

Proverbio Americano

Punto Flotante

Elaborado por Prof. Ricardo González
A partir de Materiales de las Profesoras
Angela Di Serio
María Blanca Ibañez

Representacion en Punto Flotante

Además de los enteros con y sin signo, existen problemas que deseamos modelar en el computador que requieren de números fraccionarios, que en matemática se le llama números Reales.

3,14159265 …… (pi)
2,71828 …….. (e)
0,000000001 o 1,0 x 10^{-9} (segundos en un nanosegundo)
3.155.760.000 o 3,15576 x 10^{9} (segundos en un siglo)

A la notación de la derecha de los dos últimos números se le llama notación científica. Que tiene un solo digito a la izquierda del punto decimal.

Representación en Punto Flotante

Fundamentos:

Supongamos que tenemos 62 digitos para almacenar enteros y decidimos tener:

- 34 dígitos a la izquierda del punto decimal.
- 28 dígitos a la dcrccha del punto decimal.

Ejemplos:

Masa del electron (9×10^{-28} gr):

0000000000000000000000000000000000.0000000000000000000000000009

Masa del Sol (2×10^{33} gr):

2000000000000000000000000000000000.0000000000000000000000000000

- La cantidad de dígitos significativos es 1 y no 62!
- Necesitamos un sistema en el que el intervalo de los números que puedan expresarse sea independiente del número de dígitos.

Representacion en Punto Flotante
Notación Científica

$$n = f \times 10^{e}$$

- **f** es la fracción o mantisa

- **e** es un entero positivo o negativo llamado exponente .

A la versión en una computadora de este tipo de notación se le denomina **Punto Flotante**.

Ejemplos:

$3.14 = 3.14 \times 10^{0}$

$0.000001 = 1.0 \times 10^{-6}$

$1941 = 1.941 \times 10^{3}$

Representacion en Punto Flotante

Número en punto flotante = $d_0 . d_1 d_2 d_3 \ldots . d_k \times b^{exp}$

Donde $d_0 . d_1 d_2 d_3 \ldots . d_k$ es la mantisa, b la base y exp el exponente.

¿Qué se necesita para representar un número en punto Flotante?

- El signo del número
- El signo del exponente
- Dígitos para el exponente
- Dígitos para la Mantisa

Números normalizados

Un número que no va precedido por ceros se dice que está normalizado

Normalizado	No están Normalizados
1.0×10^{-8}	0.1×10^{-9} o 10.0×10^{-10}

Un número punto flotante puede expresarse de diferentes formas que son equivalentes, es necesario establecer una única representación por lo que se trabaja con números normalizados. Decimos que un número esta normalizado si el dígito a la izquierda del punto coma está entre 0 y la base

(0 < dígito a la izquierda del punto < b)

En particular decimos que un número binario está normalizado si el dígito a la izquierda del punto es igual a 1.

Notación normalizada

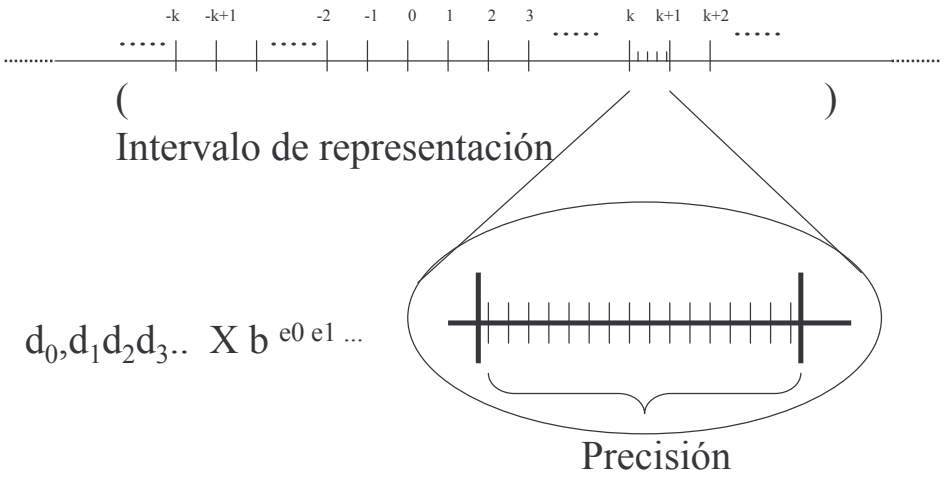
Por lo general los números en punto flotantes serán almacenados en notación normalizada, donde el punto se encuentra a la derecha del primer dígito significativo del número

Ejemplos:

$3450000_{10} \Rightarrow 3.45 \times 10^{6}$

$0.1001_{2} \Rightarrow 1.001 \times 2^{-1}$

Cálculo de Intervalo y Precisión



El intervalo esta determinado por el número de dígitos del exponente.
La precisión esta determinada por el número de dígitos de la fracción o mantisa

Cálculo de Intervalo y Precisión

Ejemplo

5 dígitos y dos signos para representar magnitudes entre:

$+ 1.00 \times 10^{-99}$ y $9.99 \times 10^{+99}$

Fracción de tres dígitos y signo dentro del intervalo:

$1 <= | f | < 10$

Exponente de dos dígito con signo:

$-99 <= e <= 99$

Del número infinito de reales en el intervalo:

$[+1.00 \times 10^{-99} , 9.99 \times 10^{+99}]$ sólo se pueden expresar:

179.100 números positivos

179.100 números negativos

y El 0

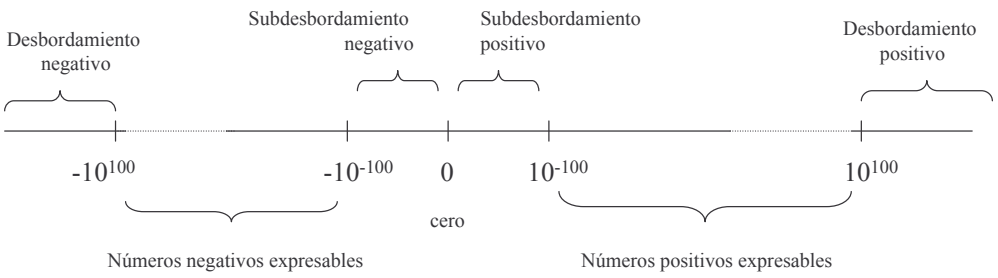
358.201

$9*10*10* (99 + 1 + 99) = 179.100$

Diferencias entre Punto Flotante y Numeros Reales

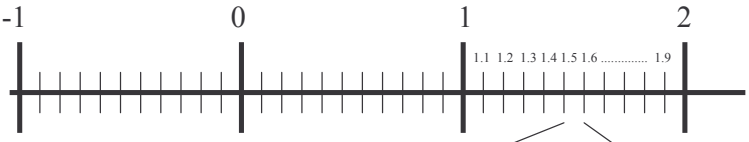
5 dígitos y dos signos para representar magnitudes entre:

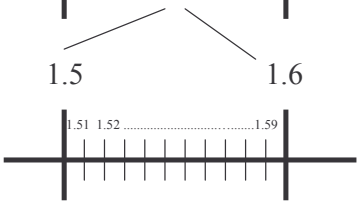
$+ 1.00 \times 10^{-99}$ y $9.99 \times 10^{+99}$



Diferencias entre Punto Flotante y Numeros Reales

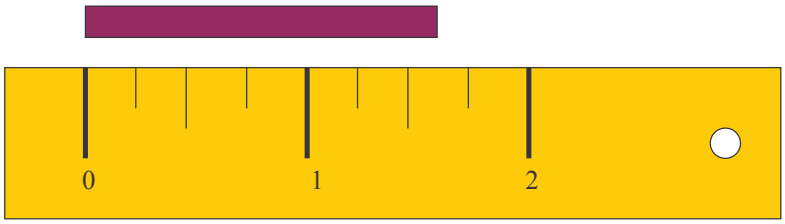
Densidad





$+ 1.00 \times 10^{-99}$ y $9.99 \times 10^{+99}$

Aquí no puedo representar 1.584 ni 1.58257



Ejemplo de Suma Punto Flotante en Base Decimal

Sumar $9.999 \times 10^{1} + 1.610 \times 10^{-1}$

Asumir que podemos almacenar

- Cuatro dígitos para la mantisa
- Dos dígitos para el exponente

Para operarlos debemos hacer que ambos exponentes sean iguales, por lo que aquél cuyo exponente sea el menor lo llevaremos al del mayor exponente.

$$1.610 \times 10^{-1} = 0.1610 \times 10^{0} = 0.01610 \times 10^{1}$$

Sumar/Restar las mantisas: $9.999 + 0.016 = 10.015$

El resultado es: 10.015×10^{1}

Normalizar:

$$10.015 \times 10^{1} = 1.0015 \times 10^{2}$$

Al redondear el número nos queda:

$$1.002 \times 10^{2}$$

$$\begin{array}{r} 9.999 \\ +\ 0.016 \\ \hline 10.015 \end{array}$$

Conversión de un número en Punto Flotante Decimal a Binario

Un número *Numb= d0.d1d2d3...* en base representa podemos reescribirlo de la siguiente forma:

$$Num_{10} = d_0\, b^0 + d_1\, b^{-1} + d_2\, b^{-2} + d_3\, b^{-3} + \ldots\ldots + d_{n-1}\, b^{-n+1} + d_n\, b^{-n}$$

$$Num_{10} = \quad d_0 + d_1\, b^{-1} + d_2\, b^{-2} + d_3\, b^{-3} + \ldots\ldots + d_{n-1}\, b^{-n+1} + d_n\, b^{-n}$$

$$Num_{10} = \quad d_0 + b^{-1} *(d_1 + d_2\, b^{-1} + d_3\, b^{-2} \ldots\ldots + d_{n-1}\, b^{-n} + d_n\, b^{-n-1})$$

$$Num_{10} = d_0 + b^{-1} *(d_1 + b^{-1} *(d_2 + d_3\, b^{-1} *(\ldots\ldots + b^{-1} * (d_{n-1} + d_n\, b^{-1}))))$$

$$Num_{10} = \quad d_0 + b^{-1}\, q_1$$

$$\text{donde } q_1 = \quad d_1 + b^{-1} * (d_2 + d_3\, b^{-1} *(\ldots\ldots + b^{-1} * (d_{n-1} + d_n\, b^{-1})\ldots))$$

Conversión de un número en Punto Flotante Decimal a Binario

De la última expresión podemos deducir el algoritmo de conversión de punto flotante decimal a cualquier base

Dado un número Num_{10} en punto flotante decimal y una base *b*

d0= parte entera(Num_{10})

$Num_{10} = (Num_{10} - d_0) * b$

i=1

Repetir desde i=1 hasta N

d_i=parte entera (Num_{10})

$Num_{10}=(Num_{10} - d_i) * b$

Conversión de un número en Punto Flotante Decimal a Binario (un ejemplo)

Convertir 3.75_{10} a binario y hallar su representación en IEEE precisión simple

i	Num_{10}	d_i=parte entera(Num_{10})	NuevoNum_{10}=(Num_{10} - d_i) * b
0	3.75	d_0=3	1.50 = (3.75-3) * 2
1	1.50	d_1=1	1.00 = (1.50-1) * 2
2	1.00	d_2=1	0.00 = (1.00-1) * 2

$3.75_{10} = d_0 . d_1 d_2 = 11.11_2 = 1.111 \times 2^1$

3.75_{10}

(3.75-3) * 2 = 1.50 d0=3

(1.50-1) * 2 = 1.00 d_1=1

(1.00-1) * 2 = 0.00 d_2=1

$3.75_{10} = 11.11_2 = 1.111 \times 2^1$

Conversión de un número en Punto Flotante Decimal a Binario (otro ejemplo)

Convertir 0.3_{10} a binario y hallar su representación en IEEE precisión simple

0.3

(0.3-0) * 2 = 0.6 d_0=0

(0.6-0) * 2 = 1.2 d_1=0

(1.2-1) * 2 = 0.4 d_2=1

(0.4-0) * 2 = 0.8 d_3=0

(0.8-0) * 2 = 1.6 d_4=0

(1.6-1) * 2 = 1.2 d_5=1

(1.2-1) * 2 = 0.4 d_6=1

(0.4-0) * 2 = 0.8 d_7=0

$0.3_{10} = 0.01001101001..._2 = 1.001101001... \times 2^{-2}$

Representación binaria del exponente

¿ Qué representación es más conveniente usar para el exponente?

Si utilizamos Complemento a Dos, los exponentes negativos aparecerán como mayores que los exponentes positivos al usar la circuitería de enteros.

C2(-1) = 1111 1111

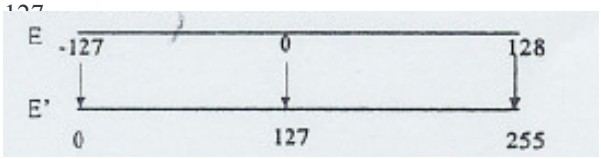
C2(0) = 0000 0000

C2(1) = 0000 0001

Para evitar este inconveniente, se utiliza una representación en exceso *N* de forma que el exponente más negativo posible quede en 0000 0001 y el más grande de los positivos en 1111 1110.

Supongamos que tenemos 8 bits para el exponente, entonces:

- Podemos representar 256 números: 0 -> 255,
- Representaremos los exponentes en el inlervalo: -127 --> 128
- Trabajeramos con la notación exceso $127_{10} = (01111111)_2$, y haremos la siguiente correspondencia: E'= E +



Representación del exponente

Ejemplo

Exponente: $10100_2 = 20_{10}$

Exceso 127: $127_{10} + 20_{10} = 147_{10} = 10010011_2$

Exponente: $-10100_2 = -20_{10}$

Exceso 127: $127_{10} - 20_{10} = 107_{10} = 01101011_2$

Estándar de Punto Flotante IEEE 754

Este estándar:

- Ha sido desarrollado para facilitar la portabilidad de los programas de un procesador a otro y para alentar el desarrollo de programas numéricos sofisticados.
- Ha sido ampliamente adoptado y se utiliza prácticamente en todos los procesadores y coprocesadores aritméticos actuales.
- Define el formato para precisión simple de 32 bits y para precisión doble de 64 bits.

Estándar de Punto Flotante IEEE 754

Precision Sencilla o Simple (32 bits)

- 1 bit para el signo del número en su totalidad.
- 8 bits para el exponente -exceso en 127-.los valores del exponente se mueven en el rango de -127 a +128.
- 23 bits para la fracción o mantisa.

Doble Precision (64 bits)

- 1 bit para el signo del número en su totalidad.
- 11 bits para el exponente -exceso en 1023-
- 52 bits para la fracción o mantisa.

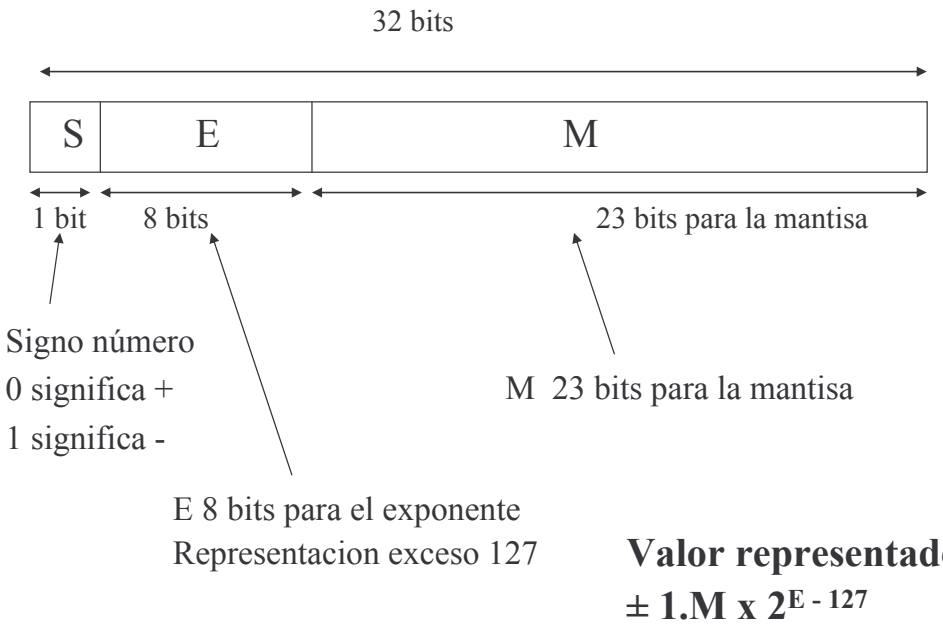
Precision Extendida (80 bits)

- Se usa tan solo en las unidades aritméticas de punto flotante.

Observaciones

- Se trabaja con números normalizados, i.e. de la forma: ± l,bb...b x $2^{\pm e}$,
  - Esto implica que el bit a la izquierda del punto es siempre 1, por lo tanto no se almacena.
  - Al no almacenarse ese 1 entonces realmente usamos 23 bits pero el número tiene un bit extra, es decir, 24 bits.
- Para el formato Simple: se suma 127 al exponente original para almacenarlo en el campo del exponente.
- Para el formato Doble: se suma 1023 al exponente original para almacenarlo en el campo del exponente.
- La base es 2.

Precisión Simple



Valor representado:
$\pm$ 1.M x $2^{E - 127}$

Precisión Simple

Signo se encuentra en el bit más significativo, de esta manera podemos usar la misma circuitería (de enteros) para llevar a cabo comparaciones con respecto al cero.

Mantisa. Está formada por el resto de los bits en la palabra (23). Como los números se representan de manera normalizada entonces siempre tendremos un 1 a la izquierda del punto. Por lo tanto este dígito no es necesario almacenarlo en la palabra y se tiene de manera implícita que la mantisa consiste realmente en 24 bits de precisión.

Precisión Simple

Exponente con signo. Está conformado por los siguientes 8 bits. Esta ubicación del exponente en la palabra facilita las comparaciones de números. Si los números se encuentran normalizados, comparamos los exponentes. Si son iguales pasamos a comparar las mantisas. Pero, ¿ qué representación es más conveniente usar para el exponente?. Si utilizamos Complemento a Dos, los exponentes negativos aparecerán como mayores que los exponentes positivos al usar la circuitería de enteros.

Para evitar este inconveniente, se utiliza una representación en exceso *N* de forma que el exponente más negativo posible quede en 0000 0001 y el más grande de los positivos en 1111 1110. El estándar IEEE 754 usa como exceso 127 para precisión simple.

Exponente más negativo representable:

x + 127 = 0000 0001

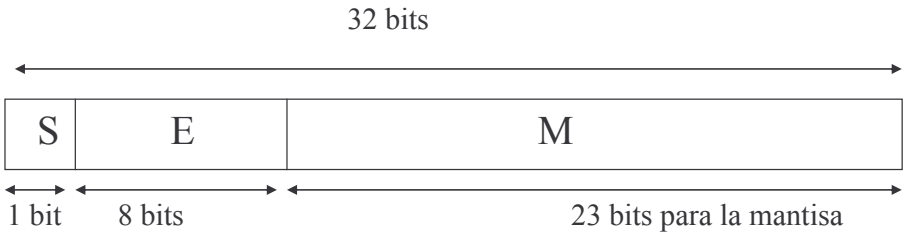
x = -126

Exponente más grande representable

x + 127 = 1111 1110

x = 127

Precisión Simple (un ejemplo)



$7_{10} = 111_2$

Normalizamos el número y nos queda $1.11_2 \times 2^2$

El signo es positivo por lo que el campo de S queda con valor 0

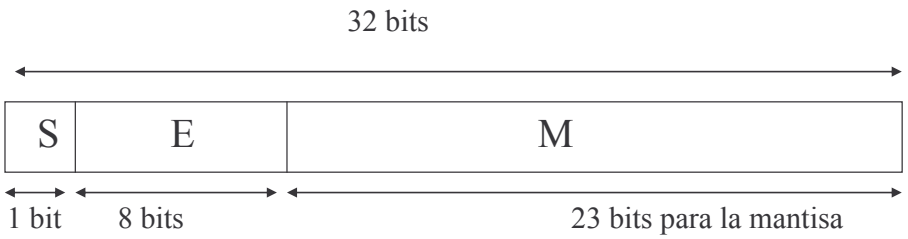
Calculamos el exponente con exceso 127

$2 + 127 = 129 = 1000\ 0001_2$

El número 7_{10} en el estándar IEEE es representado como:

0	1000 0001	11000000000000000000000

Precisión Simple (otro ejemplo)



$21_{10} = 10101_2$

Normalizamos el número y nos queda $1.0101_2 \times 2^4$

El signo es positivo por lo que el campo de S queda con valor 0

Calculamos el exponente con exceso 127

$4 + 127 = 131 = 1000\ 0011_2$

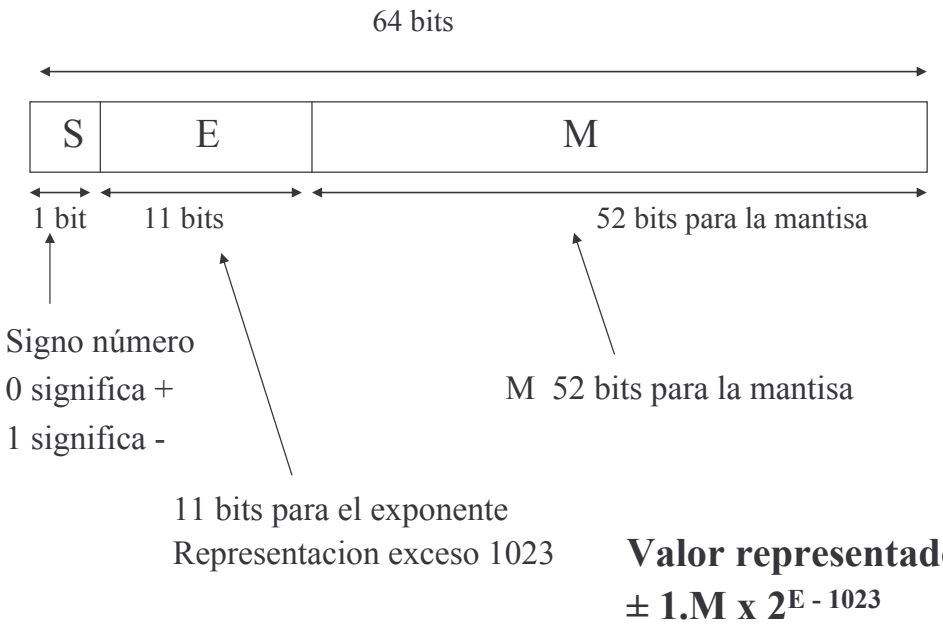
El número 21_{10} en el estándar IEEE es representado como:

0	10000011	01010000000000000000000

Rango de Valores

- Para un número de 32 bits (intervalo)
  - 8 bit de exponente
  - +/- $2^{256} = 1.6 \times 10^{77}$
- Precisión
  - 23 bits de mantisa $2^{-23} = 1.192 \times 10^{-7}$
  - Un dígito entero y alrededor de 7 decimales

Precisión Doble



Valor representado:
$\pm$ 1.M x $2^{E - 1023}$

Precisión Doble

La representación de un número en precisión doble en el formato IEEE-754 consta de las siguientes partes:

- *Signo* se encuentra en el bit más significativo.
- *Exponente en exceso*. Está conformado por los siguientes 11 bits. Se utiliza una representación en exceso 1023 de forma que el exponente más negativo posible quede en 000 0000 0001 y el más grande de los positivos en 111 1111 1110.
- *Mantisa.* Está formada por 52 bits más el bit implícito (53).

Casos especiales

Para valores de exponente desde 1 hasta 254 en el formato simple y desde 1 a hasta 2046 en el formato doble, se representan números en punto fijo normalizados. El exponente está en exceso, siendo el rango del exponente de -126 a +127 en el formato simple y de –1022 a +1023 en el doble.

Un número normalizado debe contener un bit 1 a la izquierda del punto binario; este bit está implícito, dando una mantisa efectiva de 24 bits para precisión simple o 53 bits para precisión doble.

Un exponente cero junto con una parte fraccionaria cero representa el cero positivo o negativo, dependiendo del bit de signo. Es útil tener una representación del valor 0 exacto.

Casos especiales

Precisión Simple

Exponente en exceso	Mantisa	Valor
0	0	Cero
0	<>0	Número no normalizado (0. + Mantisa) x 2^{-126}
1 .. 254		(1. + Mantisa) x $2^{exp-127}$
255	0	Infinito
255	<>0	Not a Number

Valor representado:
$\pm$ 1.M x $2^{E - 127}$

Casos especiales

Precisión Doble

Exponente en exceso	Mantisa	Valor
0	0	Cero
0	<>0	Número no normalizado (0. + Mantisa) x 2^{-1022}
1 .. 2046		(1. + Mantisa) x $2^{exp-1023}$
2047	0	Infinito
2047	<>0	Not a Number

Valor representado:
$\pm$ 1.M x $2^{E - 1023}$

Conversión de un decimal a binario enformato IEEE 754

Convertir 3.75_{10} a binario y hallar su representación en IEEE precisión simple

3.75_{10}

(3.75-3) * 2 = 1.50 d0=3

(1.50-1) * 2 = 1.00 d_1=1

(1.00-1) * 2 = 0.00 d_2=1

$3.75_{10} = 11.11_2 = 1.111 \times 2^1$

signo	exponente con signo	Mantisa
1	8	23

exponente en exceso = $1 + 127 = 128_{10} = 1000\ 0000_2$

Signo positivo = 0

0	1000 0000	11100000000000000000000
signo	exponente en exceso	mantisa

Conversión de un decimal a binario enformato IEEE 754

Convertir 0.3_{10} a binario y hallar su representación en IEEE precisión simple

0.3

(0.3-0) * 2 = 0.6 d_0=0

(0.6-0) * 2 = 1.2 d_1=0

(1.2-1) * 2 = 0.4 d_2=1

(0.4-0) * 2 = 0.8 d_3=0

(0.8-0) * 2 = 1.6 d_4=0

(1.6-1) * 2 = 1.2 d_5=1

(1.2-1) * 2 = 0.4 d_6=1

(0.4-0) * 2 = 0.8 d_7=0

signo	exponente con signo	Mantisa
1	8	23

$0.3_{10} = 0.010011001100110011001100110011001100..._2 = 1.001101100110011... \times 2^{-2}$

exponente en exceso = $-2 + 127 = 125_{10} = 0111\ 1101_2$

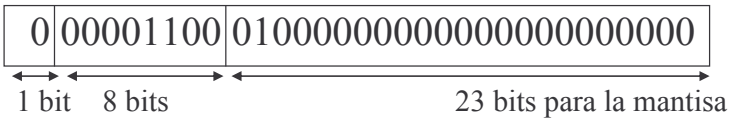
Signo positivo = 0

0	0111 1101	00110100100100100100100
Signo	Exponente en exceso	Mantisa

Conversión de un binario enformato IEEE 754 a decimal

¿Qué número decimal representa el siguiente patrón de bits en IEEE precisión simple?

0 00001100 01000000000000000000000



Calculamos el exponente que va a formar parte del número decimal, restando el valor del exponente menos el exceso de 127.

00001100_2 exponente en exceso = 12_{10} = exponente + 127_{10} =>

exponente = 12 - 127 = -115_{10}

Mantisa = 1.01000000000000000000000

Los dígitos que están en la mantisa van a formar parte del número decimal, y por tanto el número representado es

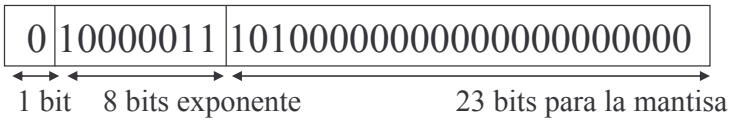
$1.01_2 \times 2^{-115} = (1. + 0*2^{-1} + 1* 2^{-2}) \times 2^{-115} = (1. + 0.25) \times 2^{-115}$

$= 1.25_{10} \times 2^{-115} = 1.25_{10} \times 2.40741 \times 10^{-35} = 3.00927 \times 10^{-35}$

Conversión de un binario enformato IEEE 754 a decimal

¿Qué número decimal representa el siguiente patrón de bits en IEEE precisión simple?

0 10000011 10100000000000000000000



Exponente en exceso = 131

Exponente = 131 - 127 = 4

Mantisa = 10100000000000000000000

$1.1010 \times 2^4 = 11010 = 26_{10}$

2^4	2^3	2^2	2^1	2^0
1	1	0	1	0
16	8	4	2	1

16 + 8 +2 = 26

Underflow o pérdida de significancia

El underflow o Subdesbordamiento ocurre cuando el número con el que se está trabajando es tan pequeño que no puede ser representado en la notación punto flotante y entonces su valor es aproximado al valor de Cero (0).

Dicha situación puede ocasionar errores graves en cálculos numéricos dentro del computador.

Microsoft Excel - puntoflotante.xls

File Edit View Insert Format Tools Data Window Help

E9 =

	A	B	C	D	E	F	G	H
1							d	
2		a	b	c	a+b	a+b-a	(a+b-a)/b	c/d
3	Computador	10000000000	1E-10	123	10000000000	0	0	#DIV/0!
4								
5	Calculo Real	10000000000	1E-10	123	suma(10000000000.0000000001)	1E-10	1	123

Underflow o pérdida de significancia

¿Por que ocurre el underflow?

Supongamos que nuestra mantisa nos permite almacenar sólo 10 digitos

$1.0000000000 \times 2^{15}$

$1.0000000000 \times 2^{-10}$

Al normalizar

$1.0000000000 \times 2^{15}$

$+ \ 0.0000000000000000000000001 \times 2^{15}$

$1.0000000000000000000000001 \times 2^{15}$

Pero este resultado no cabe en una mantisa de 10 dígitos por lo que se almacena la parte más significativa del número y se descarta el resto.

$1.0000000000000000000000001 \times 2^{15}$

1.00000000×2^{15}