**IEEE punto flotante**

**De Wikipedia, la enciclopedia libre**

"<http://es.wikipedia.org/wiki/IEEE_punto_flotante>"

El **estándar de la** [**IEEE**](http://es.wikipedia.org/wiki/IEEE) **para aritmética en coma flotante** (**IEEE 754**) es el estándar más extendido para las computaciones en [punto flotante](http://es.wikipedia.org/wiki/Punto_flotante), y es seguido por muchas de las mejoras de [CPU](http://es.wikipedia.org/wiki/CPU) y [FPU](http://es.wikipedia.org/wiki/FPU). El estándar define formatos para la representación de [números en punto flotante](http://es.wikipedia.org/wiki/Punto_flotante) (incluyendo el cero) y valores desnormalizados, así como valores especiales como [infinito](http://es.wikipedia.org/wiki/Infinito) y [NaN](http://es.wikipedia.org/wiki/NaN), con un conjunto de **operaciones en** [**punto flotante**](http://es.wikipedia.org/wiki/Punto_flotante) que trabaja sobre estos valores. También especifica cuatro modos de redondeo y cinco excepciones (incluyendo cuando dichas excepciones ocurren, y que sucede en dichos momentos).

IEEE 754 especifica cuatro formatos para la representación de valores en punto flotante: precisión simple (32 bits), precisión doble (64 bits), precisión simple extendida (>= 43 bits, no usada normalmente) y precisión doble extendida (>= 79 bits, usualmente implementada con 80 bits). Sólo los valores de 32 bits son requeridos por el estándar, los otros son opcionales. Muchos lenguajes especifican qué formatos y aritmética de la IEEE implementan, a pesar de que a veces son opcionales. Por ejemplo, el [lenguaje de programación C](http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_C), ahora permite pero no requiere la aritmética de la IEEE (el tipo de C float es típicamente usado para la precisión simple de la IEEE y el tipo double usa la precisión doble del la IEEE).

El título completo del estándar es **IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Std 754-1985)**, y también es conocido por **IEC 60559:1989, Binary floating-point arithmetic for microprocessor systems** (originalmente el número de referencia era IEC 559:1989). [[1]](http://www.opengroup.org/onlinepubs/009695399/frontmatter/refdocs.html)

|  |
| --- |
| **Contenido**   * [1 Anatomía de un número en punto flotante](http://lsi.vc.ehu.es/asignaturas/FdIc/labs/a1/htm/IEEE_punto_flotante.htm#Anatom.C3.ADa_de_un_n.C3.BAmero_en_punto_flotante)   + [1.1 Convenciones de Bit usadas en este artículo](http://lsi.vc.ehu.es/asignaturas/FdIc/labs/a1/htm/IEEE_punto_flotante.htm#Convenciones_de_Bit_usadas_en_este_art.C3.ADculo)   + [1.2 Precisión simple 32-bits](http://lsi.vc.ehu.es/asignaturas/FdIc/labs/a1/htm/IEEE_punto_flotante.htm#Precisi.C3.B3n_simple_32-bits)   + [1.3 Un Ejemplo](http://lsi.vc.ehu.es/asignaturas/FdIc/labs/a1/htm/IEEE_punto_flotante.htm#Un_Ejemplo) * [2 Precisión doble 64-bits](http://lsi.vc.ehu.es/asignaturas/FdIc/labs/a1/htm/IEEE_punto_flotante.htm#Precisi.C3.B3n_doble_64-bits)   + [2.1 Comparación de números en punto flotante](http://lsi.vc.ehu.es/asignaturas/FdIc/labs/a1/htm/IEEE_punto_flotante.htm#Comparaci.C3.B3n_de_n.C3.BAmeros_en_punto_flotante)   + [2.2 Redondeo de números en punto flotante](http://lsi.vc.ehu.es/asignaturas/FdIc/labs/a1/htm/IEEE_punto_flotante.htm#Redondeo_de_n.C3.BAmeros_en_punto_flotante) * [3 Revisión del estándar](http://lsi.vc.ehu.es/asignaturas/FdIc/labs/a1/htm/IEEE_punto_flotante.htm#Revisi.C3.B3n_del_est.C3.A1ndar) * [4 Enlaces externos](http://lsi.vc.ehu.es/asignaturas/FdIc/labs/a1/htm/IEEE_punto_flotante.htm#Enlaces_externos) |

**Anatomía de un número en punto flotante**

**Convenciones de Bit usadas en este artículo**

Los [Bits](http://es.wikipedia.org/wiki/Bit) dentro de una palabra de tamaño W están indexados por [enteros](http://es.wikipedia.org/wiki/Entero) en el rango 0 a W-1 inclusive. El bit cuyo índice es 0 se sitúa a la derecha. El menor bit indexado es normalmente el menos significativo.

**Precisión simple 32-bits**

Un número en punto flotante de precisión simple se almacena en una palabra de 32 bits.

1 8 23 <-- tamaño en bits

+-+--------+-----------------------+

|S| Exp | Mantisa |

+-+--------+-----------------------+

31 30 23 22 0 <-- índice del bit (0 a la derecha)

desplazado +127

donde S es el bit de signo y Exp es el campo exponente. (Para el signo: 0=Positivo ; 1= Negativo).

El exponente es desplazado en el un número en precisión simple, un exponente en el rango -126 a +127 es desplazado mediante la suma de 127 para obtener un valor en el rango 1 a 254 (0 y 255 tienen valores especiales descritos más adelante). Cuando se interpreta el valor en punto flotante, el número es desplazado de nuevo para obtener el exponente real.

El conjunto de valores posibles pueden ser divididos en los siguientes:

* ceros
* números normalizados
* números desnormalizados
* infinitos
* NaN (no E, no es un número, como por ejemplo, la raíz cuadrada de un número negativo)

Las clases se distinguen principalmente por el valor del campo Exp, siendo modificada ésta por el campo fracción. Considera Exp y Fracción como campos de números binarios sin signo (Exp se encuentra en el rango 0-255):

|  |  |  |
| --- | --- | --- |
| **Clase** | **Exp** | **Fracción** |
| Ceros | 0 | 0 |
| Números desnormalizados | 0 | distinto de 0 |
| Números normalizados | 1-254 | cualquiera |
| Infinitos | 255 | 0 |
| NaN (Not a Number) | 255 | distinto de 0 |

Para números normalizados, los más comunes, Exp es el exponente desplazado y Fracción es la parte fraccional de la [mantisa](http://es.wikipedia.org/wiki/Mantisa) (o significando). El número tiene valor v:

v = s Ã— 2e Ã— m

Donde

s = +1 (números positivos) cuando S es 0

s = âˆ’1 (números negativos) cuando S es 1

e = Exp âˆ’ 127 (en otras palabras, al exponente se le suma 127 i se almacena, a esto también se le llama "biased with 127" en inglés)

m = 1.Fracción en binario (esto es, el significando es el número binario 1 seguido por el punto decimal seguido por los bits de Fracción). Por lo tanto, 1 â‰¤ m < 2.

Notas:

1. Los números desnormalizados son iguales excepto que e = âˆ’126 y m = **0.Fracción**. (e NO es -127 : el significando ha de ser desplazado a la derecha por un bit más, de forma que incluya el bit principal, que no siempre es 1 en este caso. Esto se balancea incrementando el exponente a -126 para el cálculo.)
2. âˆ’126 es el menor exponente para un número desnormalizado
3. Hay dos ceros. +0 (S es 0) y âˆ’0 (S es 1)
4. Hay dos infinitos +âˆž (S es 0) y âˆ’âˆž (S es 1)
5. Los NaN s pueden tener un signo y un significando, pero estos no tienen otro significado que el que puedan aportar en pruebas de diagnóstico; el primer bit del significando es a menudo utilizado para distinguir *NaN s señalizados* de *NaN s silenciosos*
6. los NaNs y los infinitos tienen todos los bits a 1 en el campo Exp.

**Un Ejemplo**

Codifiquemos el número decimal -118.625 usando el sistema de la IEEE 754.

Necesitamos obtener el signo, el exponente y la fracción.

Dado que es un número negativo, el signo es "1". Busquemos los demás valores:

Primero, escribimos el número (sin signo) usando notación binaria. Mira el [sistema de numeración binario](http://es.wikipedia.org/wiki/Sistema_de_numeraci%C3%B3n_binario) para ver cómo hacer esto. El resultado es 1110110.101.

Ahora, movamos el punto decimal a la izquierda, dejando sólo un 1 a su izquierda.

1110110.101=1.110110101Â·26 Esto es un número en coma flotante normalizado.

La mantisa es la parte a la derecha del punto decimal, rellenada con ceros a la derecha hasta que obtengamos todos los 23 bits. Es decir 11011010100000000000000.

El exponente es 6, pero necesitamos convertirlo a binario y desplazarlo (de forma que el exponente más negativo es 0, y todos los exponentes son solamente números binarios no negativos). Para el formato IEEE 754 de 32 bits, el desplazamiento es 127, así es que 6 + 127 = 133. En binario, esto se escribe como 10000101.

Poniendo todo junto:

1 8 23 <-- tamaño en bits

+-+--------+-----------------------+

|S| Exp | Mantisa |

|1|10000101|11011010100000000000000|

+-+--------+-----------------------+

31 30 23 22 0 <-- índice del bit (0 a la derecha)

desplazado +127

**Precisión doble 64-bits**

La precisión doble es esencialmente lo mismo exceptuando que los campos son de mayor tamaño (más bits por campo):

1 11 52 <-- tamaño en bits

+-+-----------+----------------------------------------------+

|S| Exp | Mantisa |

+-+-----------+----------------------------------------------+

63 62 52 51 0 <-- índice del bit (0 a la derecha)

desplazado +1023

Los NaN s y los infinitos son representados con todos los bits de los Exp siendo 1 (2047 en decimal).

Para los números normalizados, el exponente es desplazado +1023 (así nuestro *ejemplo* anterior es Exp - 1023) Para números denormalizados el exponente es -1022 (el mínimo exponente para un número normalizadoâ€” no es 1024 porque los números normalizados tienen un bit a 1 delante del punto binario y los números desnormalizados no). Como antes, ambos infinitos y los ceros contienen signo.

**Comparación de números en punto flotante**

La comparación de números en punto flotante se realiza generalmente usando instrucciones de punto flotante. Sin embargo esta representación (IEEE 754) hace la comparación de determinados subconjuntos posible byte-por-byte, si comparten el mismo orden de bytes y el mismo signo, y además los NaN s son excluidos.

Por ejemplo, para dos números positivos a y b, a < b es cierto siempre que los enteros binarios sin signo con los mismos patrones de bits y el mismo orden de bytes que a y b son también ordenados de forma a < b. En otras palabras, dos números positivos (que se sabe que no son NaN s) puede ser comparados con una comparación entre enteros binarios sin signo entre los mismos grupos de bits, teniendo como base que los números tienen el mismo orden de bytes (esta ordenación, por tanto, no puede ser utilizada a través de una *unión* en el [lenguaje de programación C](http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_C). Este es un ejemplo de [ordenación lexicográfica](http://es.wikipedia.org/w/index.php?title=Ordenaci%C3%B3n_lexicogr%C3%A1fica&action=edit&redlink=1).

**Redondeo de números en punto flotante**

El estándar de la IEEE tiene cuatro formas diferentes de redondeo:

* **Unbiased** que redondea al número más cercano, si el número cae en medio, este es redondeado al valor más cercano con un valor par (cero) en su bit menos significativo. Este modo es el requerido como por defecto.
* **Hacia el cero**
* **Hacia el infinito positivo**
* **Hacia el infinito negativo**

**Revisión del estándar**

Nótese que el estándar IEEE 754 esta actualmente en revisión (2004). Ver [IEEE 754r](http://es.wikipedia.org/w/index.php?title=IEEE_754r&action=edit&redlink=1)

**Enlaces externos**

* [Referencias IEEE 754](http://babbage.cs.qc.edu/courses/cs341/IEEE-754references.html)
* [Let's Get To The (Floating) Point por Chris Hecker](http://chrishecker.com/images/f/fb/Gdmfp.pdf)
* [What Every Computer Scientist Should Know About Floating-Point Arithmetic by David Goldberg](http://docs.sun.com/source/806-3568/ncg_goldberg.html) - una buena introducción y explicación.
* [Curso de Representación de los Datos de Carlos Pes](http://www.carlospes.com/curso_representacion_datos/)
* [Comparing floating point numbers Bruce Dawson](http://www.cygnus-software.com/papers/comparingfloats/comparingfloats.htm)

Obtenido de "<http://es.wikipedia.org/wiki/IEEE_punto_flotante>"

[Categoría](http://es.wikipedia.org/wiki/Especial:Categories): [Aritmética computacional](http://es.wikipedia.org/wiki/Categor%C3%ADa:Aritm%C3%A9tica_computacional)