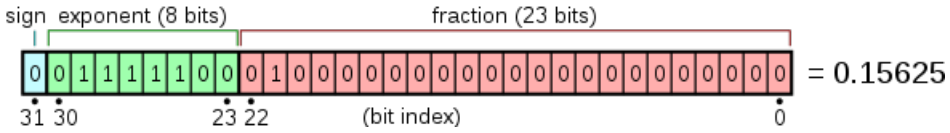
Operaciones de punto flotante (Floating Point, o FP)

Publicado el El Informatico - 26 de diciembre de 2022 -



Representación de los bits de un dígito de punto flotante (precisión simple)

Los procesadores son «malos» con las matemáticas. Aunque parezca que no es así, un procesador sólo es capaz, por defecto, de realizar sumas, operaciones de desplazamientos de bits (bit shifting), operaciones lógicas siguiendo el álgebra de Boole, y comparaciones entre valores y bits. Y no sólo eso, sino que además estamos limitados, por hardware, a valores que son potencias de 2 dado que usamos un sistema de numeración binario. Para cualquier otro tipo de operación, incluso para usar valores decimales, tenemos que ser un poco creativos.

Por ejemplo, he dicho que el procesador sólo sabe sumar. No sabe restar. Pero todos sabemos que podemos introducir un número negativo en una calculadora, y la calculadora es capaz de sustraer ese número negativo de un número positivo. ¿Cómo es posible entonces? Es algo que ya expliqué en su día, en otro artículo. Podemos invertir los bits de un dígito y sumarle 1 para convertirlo en algo que denominamos **complemento a dos** (C2). Si sumamos ese dígito en C2 a otro dígito, sería lo mismo que sumar un valor inferior a 0 a ese dígito (o lo que es lo mismo, restarlo).

Esto es algo que el procesador si entiende mejor, ya que por el álgebra booleana podemos invertir bits «negandolos» (¬a, siendo a un bit con valor 0 o 1), y sumando, que si es algo posible.

Otro tipo de operaciones que podemos hacer es multiplicar o dividir en potencias de 2 usando bit shifting. Podemos realizar ciertos tipos de operaciones mediante aproximaciones, o usando tablas con valores ya conocidos...

Sin embargo, y por defecto, para todas las operaciones que realicemos usamos **valores íntegros**. Si quisiéramos utilizar valores decimales, entonces tenemos que

recurrir a otro tipo de valores. ¿Cómo representamos un valor decimal usando bits y potencias de dos? **Con valores de punto flotante**.

¿Qué es un valor de punto flotante?

Un valor de punto flotante es un valor numérico que se representa como un valor racional. Es decir, es un valor que se puede representar como una fracción, de la siguiente manera:

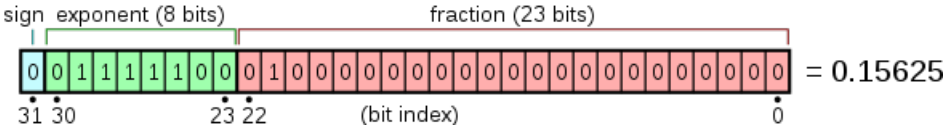
$$\frac{s}{b^{p-1}} \times b^e,$$

Siendo 's' la parte significante del valor numérico (el valor numérico como un integro sin separación de decimales. Por ejemplo, el significante de 8,47 sería 847), 'b' una base (Según el estandard IEE754, la base puede ser 2 o 10), 'p' es la precisión (el número de dígitos en la parte significante. En el caso de 847 son 3 dígitos), y 'e' el exponente (no confundir con la constante de Euler).

Por ejemplo, el número 1,52 tiene como valor significativo 152, y la precisión es 3. Si usamos como base 10, el valor de la fracción es 152/10^(3-2) = 1.52. El número se representaría entonces como 1.52 * 10^0, ya que 10^0 = 1, y 1.52 * 1 = 1.52.

De este modo, podemos representar un número decimal usando números íntegros con una base con un exponente, y una parte que se denomina «mantisa», que es el valor de la fracción. El valor numérico se puede obtener multiplicando la mantisa por la base elevada al exponente, obteniendo así un valor en notación científica (mantisa * 2^exp).

La distribución de los bits de un valor de punto flotante está en la primera imagen, pero la vuelvo a poner aquí para no hacerte ir al inicio del artículo:



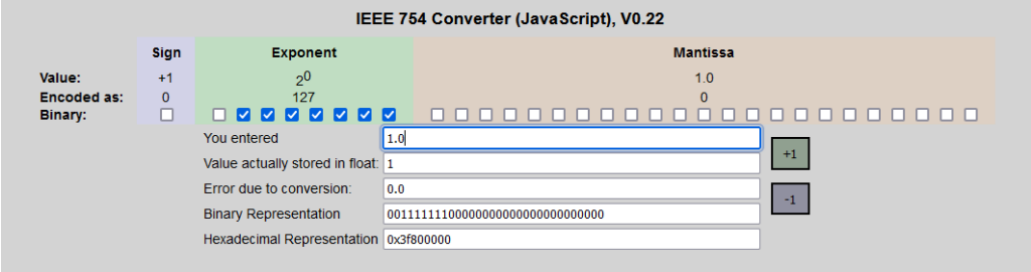
El MSB (el bit más a la izquierda) es siempre el signo. En función de la precisión (simple, doble o cuádruple) se reservan una serie de bits para el exponente 'e', y otra para la mantisa (o fracción). Los valores de punto flotante pueden ser de media precisión (16 bits), precisión simple (32-bits), precisión doble (64-bits), o precisión cuádruple (128-bits, también llamados «quads») y sus valores máximos y mínimos también varían en función del número de bits (son mucho más altos que los de los valores íntegros).

Experimenta con los FP

En la página https://www.h-schmidt.net/FloatConverter/IEEE754.html puedes

encontrar una calculadora que te permitirá experimentar y ver cómo funcionan los valores FP. Te mostrará el valor de los bits de cada número, con el error de redondeo.

Por ejemplo, si convertimos el número 1 a un valor FP de precisión doble (32 bits):



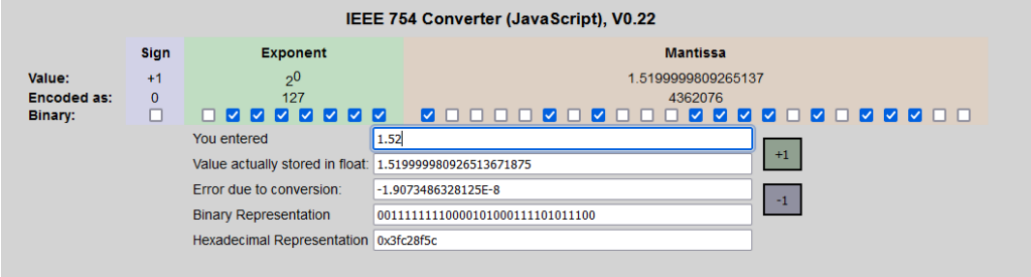
Nos muestra que el valor del exponente es 0 (Los bits del exponente no pueden estar a 0, así que el 0 se corresponde con el 127) y que el exponente (mantisa) es 1 (Cuando todos los bits están a 0, el valor de la fracción es 1. Luego verás por qué). En éste caso, el valor sigue el estandard IEEE 754 usando una base con valor de 2, de modo que el valor final es:

*signo base^exp * fracción*

El signo en 0 es positivo, con lo cual el resultado final es:

2^0 * 1.0 = 1 * 1.0 = 1.0

Si quisiéramos representar el valor 1,52:



En el caso de los decimales, como ya sabemos, no podemos almacenar decimales en un byte. Así que el valor de la fracción se almacena de la siguiente manera:

Los bits 23 a 1 representan cada uno una fracción 1/2^(24-n), donde n es el número del bit. Por ejemplo, el bit 23 representa 1/2. Bit 22 representa 1/4. Bit 21 representa 1/8, etc. A éste valor, se le añade, de forma imaginaria, un bit al final. El bit no existe dentro, se lo añadimos imaginariamente, y su valor es 1. De modo que tenemos 24 bits que representan un valor entre 1.0 y 2.0, ambos incluidos (El rango se representa como [1.0,2.0]), y cuyo valor es la suma de las fracciones correspondientes a los bits activados (Y por eso si la mantisa es 0, el valor es 1.0). En este caso, el resultado es 1.51999..., que es un valor muy aproximado a 1.52 (siempre hay un margen de error). El exponente es 0, así que el valor final es:

1,5199999809265137 * 2 ^ 0 = 1.5199999809265137 (Aproximado a 1.52)

¿Base 2 o base 10?

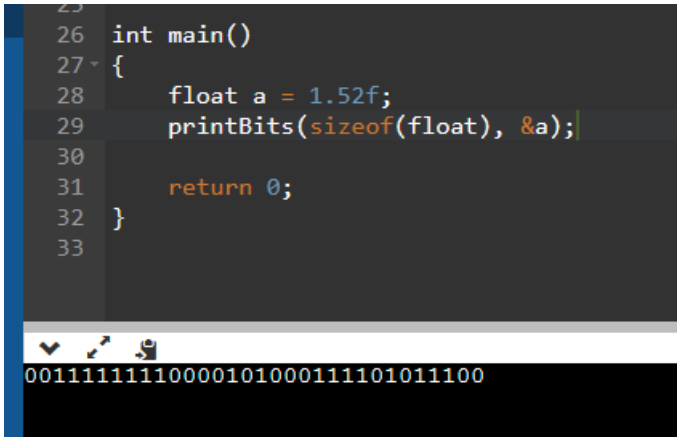
Es el eterno dilema. ¿Cuándo debemos usar base 2, y cuando 10? Al igual que ocurre con los logaritmos, usamos base 2 cuando es algo relativo a ingeniería, y base 10 cuando es algo relativo a las matemáticas.

Por ejemplo, para almacenar un valor FP usando el estandard IEEE-754, se usa base 2. Mientras que para representar un valor numérico usando notación científica, se usa base 10.

Todas las calculadoras usan base 10 tanto para la notación científica, como para los logaritmos no naturales.

Notación científica en la calculadora de Windows

Y si examinamos los bits de un FP en un programa en C, por ejemplo con valor 1,52, podemos comprobar que se almacena con base 2 porque su representación es la misma que hemos visto en la página anterior:



En el caso del almacenamiento de valores FP en bits, tiene aún más sentido usar una base 2, ya que al fin y al cabo todo se representa en potencias de base 2.

FPU, Floating Point Unit

Hace algo más de 3 décadas, los procesadores aún no podían realizar operaciones de punto flotante. En su lugar, se usaba emulación por software de puntos flotantes para poder realizar operaciones decimales con cierta precisión. Como podéis imaginar, este proceso era extremadamente lento dada la complejidad de las

operaciones requeridas en función de éstos dígitos, ya que requiere realizar numerosas multiplicaciones y divisiones entre diferentes valores, algo que como ya he dicho no es la especialidad del procesador. Y a eso, añadidle la cantidad de memoria requerida para ello.

Por este motivo, se lanzaron al mercado co-procesadores matemáticos capaces de realizar estas tareas en paralelo con el procesador, acelerando enormemente el cálculo de éstas tareas.

Coprocesador matemático Intel i387SX

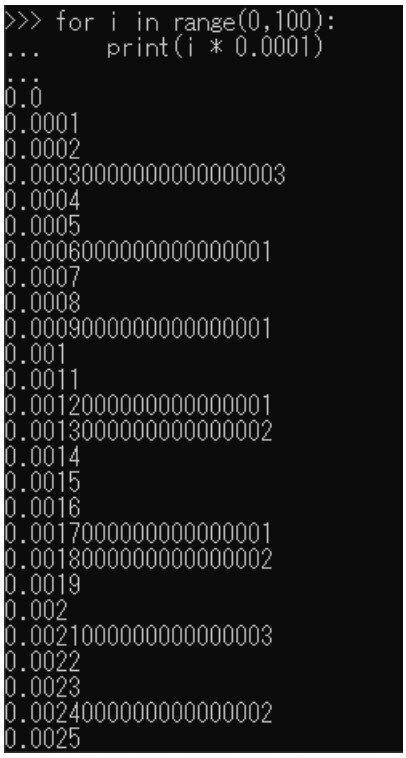
A éstos co-procesadores matemáticos se les llamaba también FPU, o Floating Point Unit.

Como tantas otras cosas, a lo largo de los años, los procesadores pasaron a incorporar el FPU dentro del propio procesador, y a día de hoy todos los procesadores (al menos los que siguen la arquitectura x86) siguen integrando, entre muchos otros componentes, un FPU con su correspondiente set de instrucciones.

Precisión y posibles errores

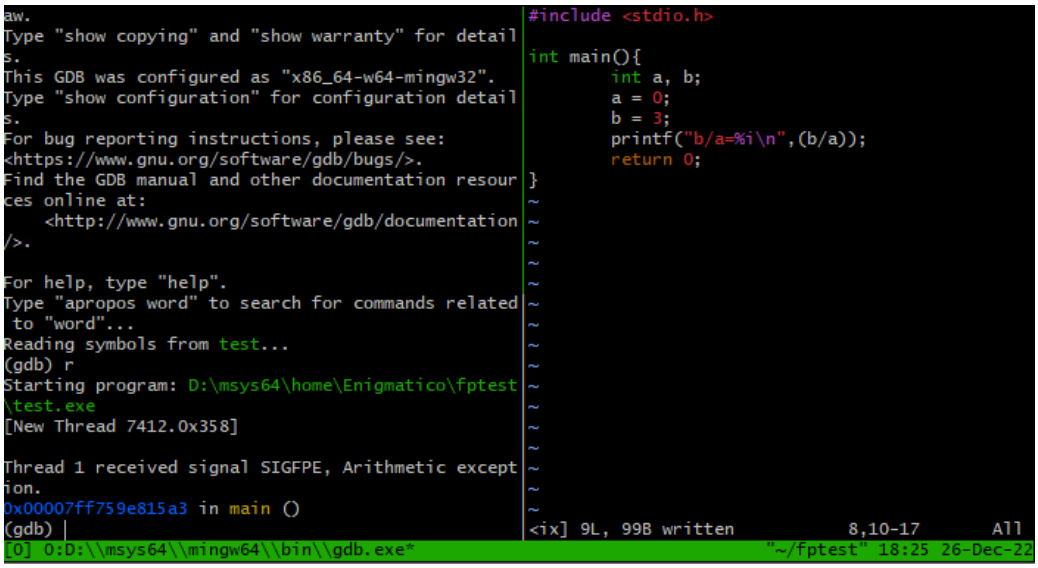
Sin embargo, los valores FP no son perfectos y en función de la precisión, puede haber errores durante el redondeo de los valores. Lo que puede dar lugar a errores en los cálculos y, por tanto, a errores en el funcionamiento de un programa.

Por ejemplo, en el siguiente programa en Python:



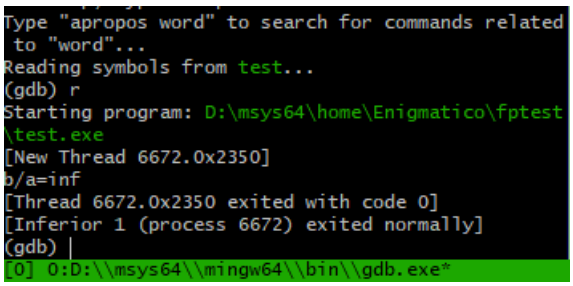
Los valores esperados serían 0.0, 0.0001, 0.0002, 0.0003, 0.0004, etc. Pero podemos comprobar que en algunos de los valores hay decimales extra que no están dentro de lo esperado. Esto se debe en un error a la hora de redondear el valor de la operación. En ocasiones los decimales deberían de redondearse a 0, pero cuando usamos valores FP los valores no se redondean nunca a 0, sino a un valor muy, muy próximo a 0, dando lugar en ocasiones a decimales extra.

Esto es más notable cuando dividimos entre 0, algo que en matemáticas no es posible ya que una división entre 0 no está definida, pero que el estandard de los FP si define. Por ejemplo, el siguiente programa en C muestra (o intenta mostrar) el resultado de una división entre 0, con dos valores integros:



Si ejecutamos el programa en GDB, podemos ver que se lanza una señal SIGFPE al intentar dividir entre 0 y el proceso es terminado. Pero, ¿Qué pasa cuando a y b son dos valores FP?

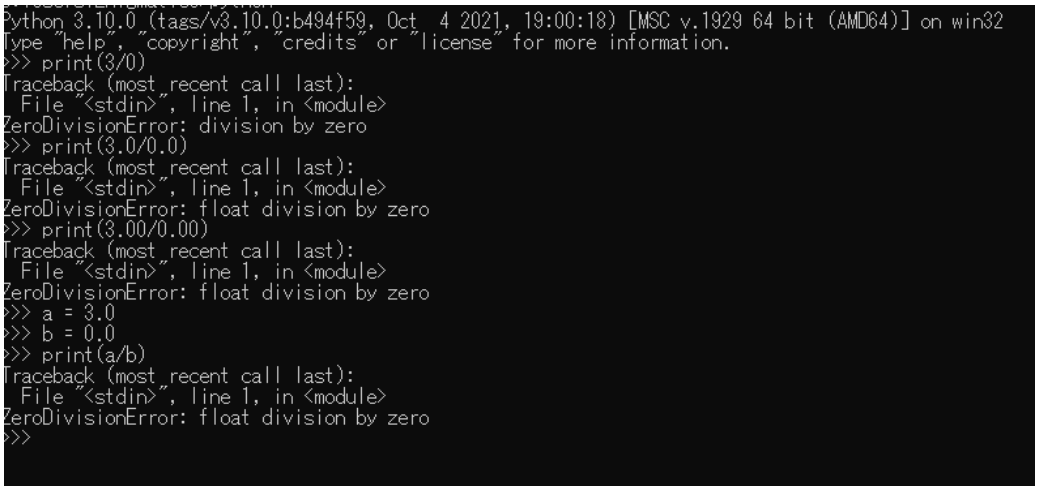
La operación de división entre 0 como un FP es válida y el resultado es +infinito. El programa finaliza normalmente, tal y como muestra GDB:



La explicación es muy sencilla. No es posible dividir entre 0 usando FP, en su lugar se redondea 0 a un valor muy, muy aproximado a 0, pero sin llegar a 0. Considera un valor numérico real ‘a’,y otro valor real ‘b’ que no es 0 pero cuyo valor se aproxima a 0, de modo que x = a/b.

Según b se aproxima a 0, el valor de x aumenta cada vez más. Matemáticamente podemos decir que el límite según b se aproxima a 0, sin llegar a 0 (b > 0) es +infinito.

En Python, sin embargo, no es posible dividir entre 0.



Pero sin embargo, podemos calcular cosas como la tangente de 90º.

Porque al igual que antes, Python está usando FP y en FP, el 0 literal no existe, sino que se trata de una aproximación. Este tipo de errores son muy notables sobre todo en software de calculo, simulación, y también en juegos donde éste tipo de errores producen resultados inesperados en los mismos, y que hay que tener muy en cuenta a la hora de desarrollar aplicaciones.

Lo ideal es que si necesitamos realizar cálculos matemáticos usando FP, debemos tener muy en cuenta el dominio de las funciones utilizadas y programar excepciones para los casos donde la función no tenga un valor definido.

ANTERIOR

¿Qué narices es Mastodon? ¿Y cómo funciona?

SIGUIENTE

¿Sufres «ghosting»? Tal vez te interese leer ésto

Entradas Recientes

- Estoy hasta las narices de la web moderna
- Ingenieria inversa básica con Ghidra
- Acerca de la nueva ley transgénero (Y sobre la disfória de género)
- Depresiones causadas por las redes sociales
- ¿Necesito saber matemáticas para aprender informática?
- ¿Es el fín de los discos duros tradicionales?

Categorías

Actualidad | Android | Básicos | Ciberseguridad | Clima | Criptografía | Electronica | Emulación / Virtualización | FOSS | Hacking | Hardware | Informática | Internet | Juegos | Opinion | Otros | Personal | Privacidad | Programación | Tecnología | Time Machine | Tutoriales

RSS

Subscribirse al feed RSS