Apéndice B

Coma flotante (ejemplo)

Los ordenadores trabajan con números decimales utilzando (en binario, pero para la explicación es mejor utilizar base 10) dos formatos:

- Punto "fijo" (fixed point): los números se almacenan en una lista de k_e elementos para la parte entera y k_d para la parte decimal, donde ambos números k_e y k_d están fijos de antemano: Las

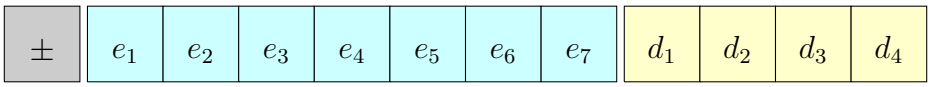

FIGURA 1. Punto fijo con 7 dígitos enteros y 4 decimales (y uno de signo).

operaciones con este tipo de aritmética tienen siempre la misma precisión "absoluta" (cada operación se realiza exactamente con k_d decimales). Los valores extremos que pueden almacenarse son $\pm 10^{k_e}$ y los valores más pequeños significativos son $\pm 10^{-k_d}$. Cuando un valor "externo" se almacena en una variable de este tipo, los dígitos decimales más allá del k_d se truncan. Por tanto, se puede llegar a perder una precisión de $\simeq 10^{-k_d}$ como mucho. El problema de este tipo de aritmética es doble:

1. Los extremos máximos por lo general son insuficientes (en binario, 32 dígitos solo cubren unos 4×10^9 valores, cuatro mil millones, que en seguida se queda corte).
2. La resolución decimal puede también quedarse corta en problemas de muy alta precisión (que hoy en día son habituales en física e ingenierías).

- "Punto flotante" (usualmente llamado *coma* flotante, en castellano), (floating point): los números se almacenan en una pareja formada por k_m elementos de la llamada "mantisa" y k_e elementos del "exponente" (aparte del signo). En principio la estructura es similar al punto fijo: Un número en esta expresión (la de la figura 2) es igual a

$$\pm 0.m_1 m_2 \cdots m_8 \times 10^{\pm e_1 e_2}$$

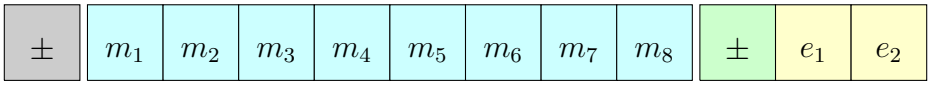

FIGURA 2. Punto flotante con mantisa de 8 dígitos enteros y exponente de 3 (dos más signo), y un signo.

donde el signo en cada parte será el que esté especificado. Se exige que el primer dígito de la mantisa sea no nulo ($m_1 \neq 0$). En binario, el signo + es un 0 y el singo − es un 1. Por definición, el valor máximo que se puede almacenar en esta coma flotante es $\pm 0.99999999 \times 10^{99}$ (un valor enorme), mientras que en decimal, los valores ínfimos son $\pm 0.1 \times 10^{-99}$ (un cero seguido de 100 ceros y un 1 tras la coma). Como se ve, este tipo de almacenamiento permite cálculos en un rango mucho más amplio que el punto fijo. Esto hace que sea posible utilizar este tipo de aritmética en problemas en que los valores de las variables sean muy altos y en problemas en que sean muy bajos, sin perder *precisión relativa*. Cada vez que un número (real) se almacena en esa coma flotante, al ser de la forma

$$0.m_1 \cdots m_8 \times 10^e$$

el error que se comete es del orden de 0.00000001×10^e, es decir, 10^{e-8}: por tanto, si e es grande, el error *absoluto* es grande, y si e es pequeño, el error *absoluto* es pequeño. Lo que caracteriza la coma flotante es que *al almacenar un número en coma flotante, el error relativo que se comete es siempre del mismo orden.*

Pero para ello necesitamos dos definiciones. Supongamos que x_0 es un número que se quiere calcular (o almacenar) y que $\overline{x}_0$ es una aproximación.

DEFINICIÓN 27. El *error absoluto* cometido al utilizar $\overline{x}_0$ en lugar de x_0 es: $|x_0 - \overline{x}_0|$. El *error relativo* es

$$\frac{|x_0 - \overline{x}_0|}{|x_0|}$$

(que solo tiene sentido si $x_0 \neq 0$, claro).

El error absoluto mide la "distancia" que hay entre el valor y su aproximación. El error relativo mide el tamaño del error "comparado con" el valor exacto. Por lo general, el error relativo es más informativo que el absoluto, pero es importante tener en cuenta ambos, *siempre*.

Por tanto, en coma flotante, el *error relativo* cometido al truncar un valor y almacenarlo es *siempre* menor que 10^{-k_m} (en decimal, en

binario será con un 2): viene dado por la longitud de la mantisa. El problema es que el error absoluto puede ser muy grande (del orden de $10^{99-8} = 10^{97}$, cuando se almacenan números de ese tamaño.

Ejemplo 12 (Desbordamiento). Se tiene un controlador digital que almacena números en punto fijo (decimal) con 5 dígitos enteros y 4 decimales. El contador comienza en 0 y se aumenta en 0.001 cada vez (suma milésimas de segundo). ¿Cuánto tardará en ocurrir un desbordamiento?

Como se almacena en punto fijo con $5+4$ dígitos, el mayor valor posible es 99999.9999. El desbordamiento se produce cuando el contador llegue a 10^5, por tanto, cuando pasen

$$10^5/10^{-3} = 10^8$$

milésimas de segundo, es decir, 10^5 segundos, que son

$$\frac{10^5}{24 \cdot 60 \cdot 60} \simeq 1.157$$

días. Es decir, al cabo de 1 día y medio se producirá un error *a ciencia cierta*.

Ejemplo 13 (Pérdida de precisión). El mismo contador que antes se almacena en un registro en coma flotante con 5 dígitos de mantisa y 4 de exponente (contando el signo). ¿Cuándo se perderá precisión?

El número 0.001 en nuestra coma flotante es 1×10^{-3}. Para que las milésimas sigan teniendo relevancia, el número mayor que se puede considerar que tenga cinco cifras es:

$$99.999 = 0.99999 \times 10^2.$$

Por tanto, cuando el contador llegue a 99.999, se producirá una pérdida de precisión (en este caso total: ya no hay milésimas y el contador se queda parado en 100). Es decir, al cabo de 100 segundos el contador falla *totalmente*.

El ejemplo 13 es extremo (el formato de coma flotante utilizado es muy burdo porque la mantisa es muy pequeña) pero es lo que le ocurrió al sistema Patriot en la Guerra del Golfo, esencialmente.

La precisión simple *sigue existiendo*: por un lado, las entidades financieras *tienen obligación* de trabajar con un número específico de decimales. Por otro lado, los *microcontroladores* están diseñados para utilizar punto fijo porque la implementación de las operaciones básicas (y, sobre todo, de la suma y la resta) es mucho más sencilla (y, por tanto, más económica).

El formato estándar de coma flotante es el documento conocido como IEEE-754, que posee varias actualizaciones (el original es de 1987).