Actividad de Representación de Datos

Sistemas de representación de números y operaciones

Para poder representar números en una computadora, sólo se dispone de celdas de memoria que pueden representar patrones de bits. Sólo se puede representar números usando ceros y unos, y no hay posibilidad de representar signos ni separación de parte entera de parte fraccionaria con un punto o una coma. Usando el sistema de numeración binaria, que usa como dígitos al 0 y al 1, se deben adoptar convenciones de representación especiales para poder representar números negativos y números fraccionarios.

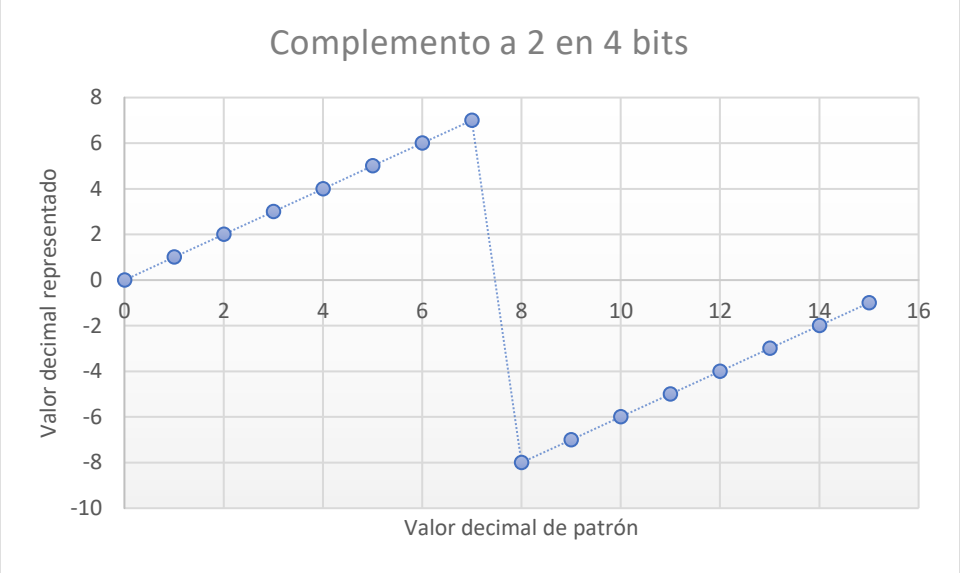
Complemento a dos

Es un sistema de representación de números enteros como patrones de bits de tamaño fijo y conocido. Para poder distinguir números negativos de no negativos se establece que el primer bit del patrón debe indicar el signo del número, asignándose el 0 a los no negativos y el 1 a los negativos.

La representación en complemento a dos de un número decimal entero ***n*** en ***b*** bits, se define como

- la representación binaria de n agregando ceros a izquierda hasta completar b bits, si n no es negativo
- la representación binaria de $2^b + n$, si n es negativo (o sea $2^b - |n|$)

Esta convención determina que se pueden representar números no negativos de hasta b–1 dígitos: el total de números positivos que se pueden representar es $2^{b-1} - 1$ (más el 0 serían 2^{b-1} no negativos), y el de negativos 2^{b-1}.



En esta representación también es fácil encontrar el opuesto de un número, ya que para cambiar el signo de cualquier patrón n_{C2} en b bits, basta con calcular $2^b - n_{C2}$ (su complemento a 2 para b bits) y representarlo en b bits completando con ceros a la izquierda si fuere necesario. Esto significa que calcular el complemento a 2 de un patrón en complemento a 2 equivale a cambiar el signo del número representado en el patrón original.

Es muy fácil obtener el complemento a dos de un número a partir de su complemento a uno (inversión de bits), porque el complemento a dos de un número binario es una unidad mayor que su complemento a uno, es decir $n_{C2} = n_{C1} + 1$.

Ejemplos de representación e interpretación

Si se considera el complemento a 2 en 2 bytes (16 bits), el rango de números decimales que se pueden representar es de $-2^{15} = -32768$ a $2^{15} - 1 = 32767$.

Representación

Para representar el decimal **19347**, se debe cambiar de base decimal a binaria. Para hacerlo en forma eficiente (minimizando cálculos) usando la numeración hexadecimal como apoyo, ya que cada dígito hexadecimal equivale a 4 binarios, se puede proceder:

19347 % 16 = **3**, 19347 // 16 = 1209

1209 % 16 = **9**, 1209 // 16 = 75

75 % 16 = **11 (B)**, 75 // 16 = **4**

El decimal 19347 equivale al hexadecimal 4B93, y al binario 100 1011 1001 0011 ($4_{16}\equiv0100_2$, $B_{16}\equiv1011_2$, $9_{16}\equiv1001_2$ y $3_{16}\equiv0011_2$ y el primer cero a la izquierda del primer dígito significativo se descarta), que en complemento a 2 en 16 bits queda **0100 1011 1001 0011** (se completa con ceros a izquierda hasta completar 16 bits para cumplir la restricción de tamaño fijo y la representación del signo).

Si quisiéramos obtener la representación de su opuesto, es decir de –19347, la fórmula sería 2^{16}–19347=65536–19347=46189, pero ya que tenemos su patrón en complemento a 2, podemos hallar el complemento a 2 del patrón más fácilmente que con el proceso de cambio de base. Se halla el complemento a 1 (invirtiendo todos los bits) y se suma 1 en binario:

1011 0100 0110 1100 + 1 = **1011 0100 0110 1101**

Para verificar su equivalencia en decimal, se opera fácilmente cambiando primero a hexadecimal

B46D ($1011_2\equiv B_{16}\equiv12_{10}$, $0100_2\equiv4_{16}$, $0110_2\equiv6_{16}$ y $1100_2\equiv D_{16}\equiv13_{10}$)

y luego cambiando a decimal $13x16^0 + 6x16^1 + 4x16^2 + 11x16^3 = 13 + 96 + 1024 + 45056 = 46189$.

Interpretación

Si a la inversa, tuviéramos el patrón previo 1011 0100 0110 1101, y supiésemos que está en complemento a 2, y obviamente en 16 bits, para interpretar a qué número decimal equivale deberíamos proceder como se explica a continuación.

Como el primer bit es un 1, está representando un número negativo. Se podría cambiar a base 10, restárselo a 65536 y poner signo negativo al resultado, pero es menos esforzado (más eficiente) hallar el complemento a 2 para obtener su valor absoluto, cambiar el valor absoluto a decimal (con menos cálculo por su menor valor), y ponerle signo negativo:

0100 1011 1001 0010 + 1 = 0100 1011 1001 0011

Entonces $1011\ 0100\ 0110\ 1101_{C2} \equiv –100\ 1011\ 1001\ 0011_2 \equiv –4B93_{16} \equiv –(3 + 9x16^1 + 11x16^2 + 4x16^3)_{10}$ = **$–19347_{10}$**.

Punto flotante normalizado

La representación de punto flotante (en inglés ***float****ing point*) es una forma de notación científica usada en las computadoras con la cual se pueden representar números reales de una manera muy eficiente y compacta, y con la que se pueden realizar operaciones aritméticas.

En notación científica decimal, un mismo número se puede representar de diferentes formas. Por ejemplo, el número -12,345 puede representarse como -12345×10^{-3}, -1.2345×10^{1}, -0.12345×10^{2}, etc. Cualquier representación es válida, y el conjunto de las representaciones posibles se llama cohorte. Cuando se trabaja con notación científica se suele establecer qué miembro de la cohorte es elegido como estándar o normal.

Para representar números binarios en notación científica como patrones de bits, se requiere dividir los patrones en tres partes: un bit para el signo, un grupo de bits para el exponente, y otro grupo para la mantisa (también denominada coeficiente o significando). La representación se normaliza (hace única) eligiendo cómo se codifica el exponente, si se representa el bit principal o más significativo o no, y la posición relativa del punto (a la izquierda o a la derecha de los dígitos de la mantisa).

La razón de la denominación de "punto flotante", es porque el punto se desplaza o "flota" tantos dígitos como indica el exponente. Al cambiar el exponente, el punto "flota" a otra posición.

El exponente indica cuánto se debe desplazar hacia la derecha o hacia la izquierda el punto binario de la parte significativa o mantisa. Pero no se almacena como un número binario con signo sino como un entero positivo equivalente que va entre 0 el máximo valor representable en la cantidad de bits que se usen para codificarlo. Para ello, al exponente se le debe sumar un desplazamiento (en inglés *bias*), que en el caso de un exponente de b bits (2^b valores), puede ser 2^{b-1} (la mitad de los valores que se pueden representar -exceso de 2^{b-1}), y al final, el rango del exponente de -2^{b-1} a $2^{b-1}-1$ queda representado internamente como un número entre 0 y 2^b-1, donde los números entre 2^{b-1} y 2^b-1 representan los exponentes entre 0 y $2^{b-1}-1$, y los números entre 0 y $2^{b-1}-1$ representan los exponentes entre -2^{b-1} y -1 respectivamente.

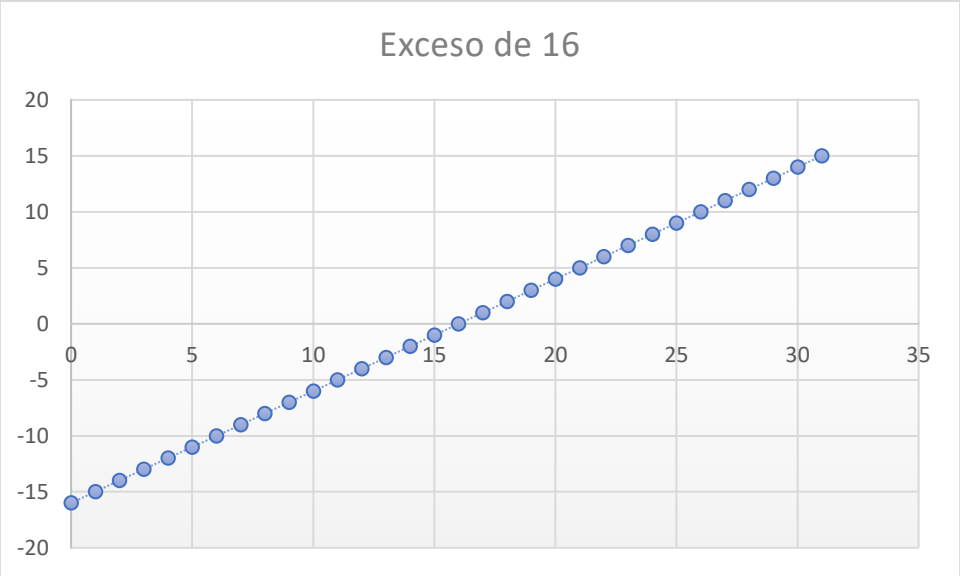
Cabe aclarar que hay otra convención que en vez de considerar un desplazamiento de 2^{b-1} bits para representar exponentes, suma en cambio $2^{b-1}-1$.

Para los números con un exponente en el rango normal (no todos unos o todos ceros), el bit inicial de la mantisa siempre será 1. En consecuencia, el bit 1 principal puede ser implícito en lugar de estar explícitamente presente en la codificación de los números. Esta regla se denomina convención de bit principal, o también convención de bits implícita o convención de bits ocultos. La regla permite que el formato de representación tenga un dígito binario más de precisión.

Ejemplos de representación e interpretación

Los lenguajes de programación utilizan estándares para representar números en punto flotante en 4 bytes (32 bits), 8 (64 bits) y, para mayores precisiones, 16 y hasta 32 bytes (128 y 256 bits respectivamente).

Por motivos prácticos, se considera una convención de punto flotante en **patrones de 16 bits**, con **5 bits para el exponente** en exceso de 2^{5-1}=16, **bit principal oculto** y punto binario a la **izquierda** de los dígitos de la mantisa.



Los patrones son de la forma

s|eeeee|mmmmmmmmmm

Donde:

- el exponente 00000 con la mantisa 0000000000 representa el valor ±0 (que puede tener signo negativo por ser resultado de alguna operación aritmética)
- el exponente 11111 se reserva para representar
  - ±∞ si la mantisa es 0000000000
  - NaN (no número -*Not a Number*) si la mantisa no es 0000000000

    Los siguientes casos pueden generar el estado de NaN en la mayoría de los lenguajes de programación que acepten este estado como retorno de una función matemática (Python lo hace):

    - Todas las operaciones matemáticas que posean NaN como operando matemático
    - Las divisiones indeterminadas o por infinito (0/0, ∞/∞, ∞/-∞, -∞/∞)
    - Las multiplicaciones de 0 por infinito (0×∞, 0×-∞)
    - Las restas de valores infinitos
    - Aplicando funciones que excedan el dominio de esta. Por ejemplo, la raíz cuadrada de un número negativo, logaritmo de cualquier número menor o igual que 0, o la inversa de un coseno que sea menor que -1 o mayor que +1.
- para exponentes entre 00001 y 11110 representa $(-1)^s$ x 1.mmmmmmmmmm x $10^{eeeee-10000}$

O sea que el mayor valor absoluto representable es 1.1111111111 x $10^{11110-10000}$ = 111|1111|1111|0000, que equivale al decimal 7 x 16^3 + 15 x 16^2 + 15 x 16^1 = 32752 (el exponente binario 11110-10000=1110 equivale al decimal 14).

Obsérvese que aparecen limitaciones importantes para representar números cuya conversión a base 2 requiera más de 11 dígitos (si la parte entera supera 11 dígitos se debe redondear, o si la representación de la parte entera más la fraccionaria requiere más de 11 dígitos se debe podar en 11).

Y el menor valor absoluto no nulo representables es 1.0 x $10^{1-10000}$ = 0.0000|0000|0000|0010, que equivale al decimal 2 x 16^{-4} = 0,000030517578125 (el exponente binario 1-10000=-1111 equivale al decimal -15).

Representación

Para representar el decimal **–564.11** se puede proceder:

Para obtener la parte entera en numeración binaria, usando la hexadecimal como atajo:

564 % 16 = **4**, 564 // 16 = 35

35 % 16 = **3**, 35 // 16 = **2**

La parte entera es $234_{16} \equiv 10|0011|0100_2$

Para obtener la parte fraccionaria en base 2, usando hexadecimal como atajo:

0.11 x 16 = **1**.76

0.76 x 16 = **12**.16 (C)

La parte fraccionaria es $1C\ldots_{16} \equiv 0001|1100\ldots_2$ (no es necesario obtener más dígitos porque no podrán representarse)

Entonces para representar 1**0 0011 0100.0**001 1100… (se resaltan los 10 dígitos que se podrán representar) hay que correr el punto 9 posiciones a la izquierda. El exponente en exceso de 16, para mover el punto 9 posiciones a la derecha es 9+16=25, que en binario con 5 bits queda 11001.

En la convención establecida, el número se redondearía a 1|11001|0001101000

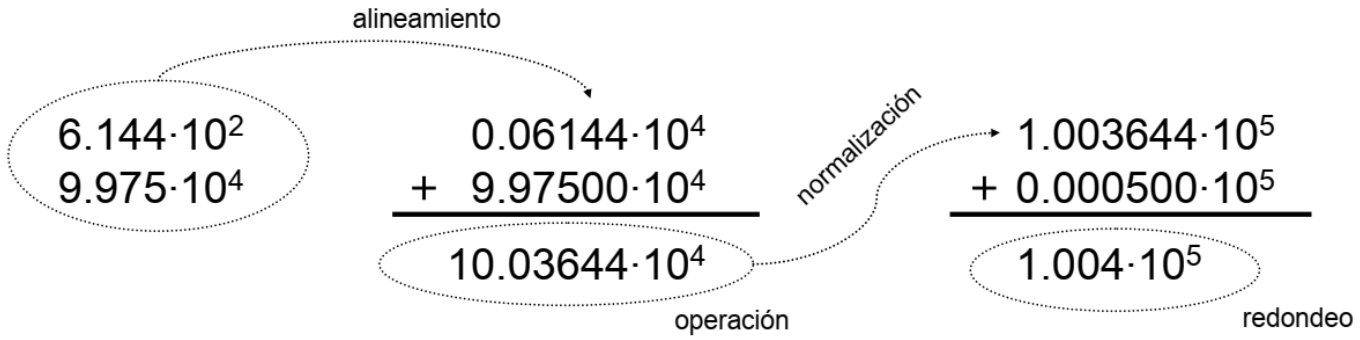
Interpretación

Para convertir el patrón previo a decimal se calcula $(-1)^1 \times 1.0001101000 \times 10^{11001-10000} = -1000110100.0$

Lo que resulta -564.0 (por la poda de dígitos fraccionarios).

Suma/resta en punto flotante

Para sumar o restar en notación científica decimal, si los exponentes de los operandos son distintos se debe alinear los dígitos del que tiene el exponente menor de manera que coincidan con los correspondientes del otro en la potencia de 10 que representan, y eventualmente redondear el resultado, si se trabaja con un número fijo de dígitos:



Método de suma/resta

- Extraer signos, exponentes y magnitudes
- Tratar operandos especiales (por ejemplo, alguno de ellos cero)
- Desplazar la mantisa del número con exponente más pequeño a la derecha |e1-e2| bits
- Fijar el exponente del resultado al máximo de los exponentes
- Si la operación es suma y los signos son iguales, o si la operación es resta y los signos son diferentes, sumar las mantisas. En otro caso restarlas
- Detectar desborde *(overflow)* de la mantisa
- Normalizar la mantisa, desplazándola a la derecha o a la izquierda hasta que el dígito más significativo esté delante del punto.
- Redondear el resultado y renormalizar la mantisa si es necesario.
- Corregir el exponente en función de los desplazamientos realizados sobre la mantisa.