OPERACIONES ELEMENTALES A REALIZAR EN MULTIPLICACIÓN CON FORMATO DE PUNTO FLOTANTE

AUTOR: GARCÍA LOMELÍ ABRAHAM AMOS
INSTITUTO POLITÉCNICO NACIONAL
ESCOM

1 Introducción

En el transcurso de entregas anteriores ha sido posible observar la habilidad con la cual los desarrolladores de arquitecturas han logrado idear estratégias (posteriormente convertidas en estándares) que permiten representar diferentes tipos de valor a partir de una lógica binaria. Recordando que en función de los componentes físicos que conforman un ordenador, será posible representar a partir de ciertos voltajes los valores de *uno lógico (1)* y *cero lógico (0)*, se puede estipular que una computadora es únicamente capaz de representar datos a partir de circuitos abiertos y cerrados que dejan o no pasar la corriente y el voltaje, generando así diferentes valores; no obstante, según la lógica de la instrucción que emite dichos datos, una secuencia de *unos y ceros* puede representar diferentes cosas, por ejemplo:

- **Letra 'A' en binario:** 01000001
- **Número 65 en binario:** 01000001
- **Exponente de 2^{-62} en el formato IEEE 754**: 01000001

La pregunta obligada sería entonces **¿cómo es que una computadora interpreta de manera diferente una misma secuencia de bits?**; para contestar esta interrogante vale la pena recordar que en función de los diferentes formatos de instrucción, se puede dar un contexto a cada uno de los datos que se envían, siendo así, si se recibe la secuencia 01000001 desde teclado, es muy probable que se trate de la letra A, no obstante si el programador tiene conocimiento de que se está haciendo uso de un estándar específico (como el formtado de punto flotante), muy probablemente interpretará esta cadena como un exponente. A partir de lo anterior es que se ha dado la necesidad de crear diferentes estándares que sean capaces de dar representación fiable a diferentes conceptos como pueden ser:

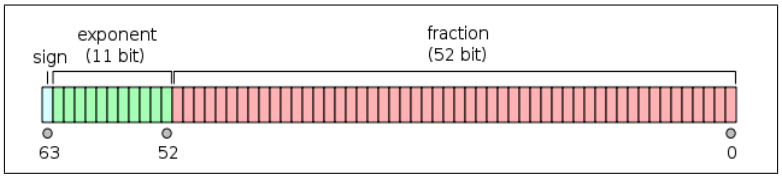
- Números negativos
- Números irracionales
- Números fraccionarios

En tareas anteriores se ha hablado de la dinámica que ofrece el formato **IEEE 754** para representar dichos números a partir de reglas sencillas (tales como la normalización), no obstante bajo lo anterior, se tiene entonces un marco de referencia común que es capaz de **estandarizar la posición del punto decimal** y de esta manera brindar una idea certera de que un número siempre será el mismo, ya que la posición del punto decimal siempre se encontrará tras el primer dígito.

Lo anterior sin embargo, no se limita únicamente a la correcta representación de cifras, sino que brinda la computadora la posibilidad de realizar las operaciones aritméticas básicas a partir de ciertos algoritmos que serán revisados a continuación:

2 Producto de binarios en formato de punto flotante IEEE 754

Antes de lograr realizar el producto de dos números en formato IEEE 754, es primero necesario recordar que cáda número está dado por una cadena de 32 o de 64 bits dividida en tres partes significativas:



- **Sign bit:** Conocido cono bit de signo, en caso de tener un valor de 1 indicará que el número es negativo, de lo contrario será positivo
- **Mantisa:** (también llamada coeficiente o significando) es la parte que contiene los dígitos del número. Mantisas negativas representan números negativos.
- **Exponente:** Indica donde se coloca el punto decimal en relación con el inicio de la mantisa. Para representar números menores a 1 se usan exponentes negativos.

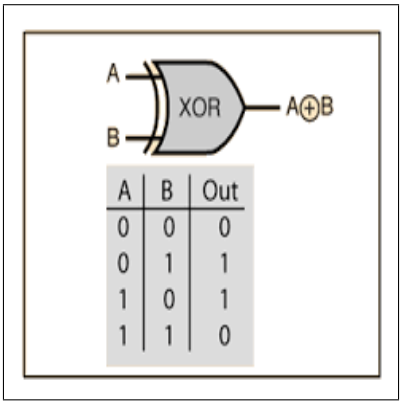
Dado lo anterior, supongamos que se tienen dos números IEEE (X, Y) donde cada uno posee mantisa, bit signado y exponente; siendo así el producto $X * Y$ estará dado por:

$$X * Y = X_m * Y_m + 2^{X_e + Y_3} \tag{1}$$

Donde:

- X_m: Es el valor de la mantisa del número X
- Y_m: Es el valor de la mantisa del número Y
- X_e: Es el valor que se encuentra en el campo del exponente de X
- Y_e: Es el valor que se encuentra en el campo del exponente de Y

Esta primera fórmula es capáz de darnos el valor exácto del producto de dos números IEEE 754, no obstante deja a un lado el signo. El estándar 754 a previsto esta situación y propone el uso de la compuerta XOR para resolver el cálculo del signo resultante. Si recordamos que *para signos iguales, el producto es positivo y para diferentes será negativo*, entonces dicho comportamiento también se adapta a:



Dado lo anterior, el signo del producto $X * Y$ estará dado por:

$$(X * Y)_{signo} = X_{signo} \text{ XOR } Y_{signo}$$

Ententiendo la lógica anterior, se procederá a realizar una serie de ejemplos didácticos para comprender la dínamica del producto bajo el formato IEEE 754

2.1 Ejemplo 1

Supóngase el número -2.5 el cuál se representa en el formato IEEE 754 como:

$$11000000001000000000000000000000 \tag{2}$$

Separando la palabra de 32 bits se tiene:

$$1\ 10000000\ 01000000000000000000000$$

Dado lo anterior se asignan los siguientes valores a X:

$$X_{signo} = 1 \tag{3}$$

$$X_e = 10000000 \tag{4}$$

$$X_m = 01000000000000000000000 \tag{5}$$

Nuestro número Y tendrá el valor de 4.23 el cual se representa en IEEE 754 como:

$$01000000100001110101110000101001 \tag{6}$$

Separando en los diferentes campos de la palabra de 32 bits se tiene:

$$0\ 10000001\ 00001110101110000101001$$

Asignando los valores a las variables que hemos propuesto en (1) se tiene:

$$Y_{signo} = 0 \tag{7}$$

$$Y_e = 10000001 \tag{8}$$

$$Y_m = 00001110101110000101001 \tag{9}$$

A continuación se procede a realizar las operaciones de la fórmula (1) diciendo que el resultado del producto será conocido como C y dado que es un número IEEE 754 poseerá los mismos campos que X y que Y:

1. Es importante recordar que **el valor decimal a usar en X_e y Y_e no es el que se obtiene directamente de la palabra digital**, es necesario realizar la resta de dicho valor menos 127, por lo tanto se dirá que:

$$X_e = X_e - 127 = 1 \tag{10}$$

$$Y_e = Y_e - 127 = 2 \tag{11}$$

2. Se dice que el nuevo valor del exponente será:

$$C_e = X_e + Y_e + 127 = 1 + 3 + 127 = 130 \tag{12}$$

   Su representación binaria es:

$$C_e = 10000010 \tag{13}$$

3. El valor de la mantiza C_m será el producto binario de $X_m * Y_m$ por lo tanto:

$$C_m = X_m{*}Y_m = 00001110101110000101001{*}01000000000000000000000 = 01010010011001100110011 \tag{14}$$

4. El bit de signo estará dado por

$$C_{signo} = (X * Y)_{signo} = X_{signo} \text{ XOR } Y_{signo} = 1 \text{ XOR } 0 = 1$$

5. Por último se procede a realizar la concatenación de C_{signo}, C_e, C_m, dado lo anterior se tiene que:

$$C = 11000001001010010011001100110011_{IEEE754} = -10.575_{10} \tag{15}$$

2.2 Ejemplo 2

Supóngase el número $3.1416 \approx \pi$ el cuál se representa en el formato IEEE 754 como:

$$01000000010010010000111111111001 \tag{16}$$

Separando la palabra de 32 bits se tiene:

$$0\ 10000000\ 10010010000111111111001$$

Dado lo anterior se asignan los siguientes valores a X:

$$X_{signo} = 0 \tag{17}$$

$$X_e = 10000000 \tag{18}$$

$$X_m = 10010010000111111111001 \tag{19}$$

Nuestro número Y tendrá el valor de $2.7182 \approx e$ el cual se representa en IEEE 754 como:

$$01000000001011011111011011111101 \tag{20}$$

Separando en los diferentes campos de la palabra de 32 bits se tiene:

$$0\ 10000001\ 01011011111011011111101$$

Asignando los valores a las variables que hemos propuesto en (1) se tiene:

$$Y_{signo} = 0 \tag{21}$$

$$Y_e = 10000000 \tag{22}$$

$$Y_m = 01011011111011011111101 \tag{23}$$

En este caso en particular se observa que las únicas variaciones son en los valores de la matiza por lo tanto:

1. Se procede a obtener los valores de las mantizas

$$X_e = X_e - 127 = 1 \tag{24}$$

$$Y_e = Y_e - 127 = 1 \tag{25}$$

2. Se dice que el nuevo valor del exponente será:

$$C_e = X_e + Y_e + 127 = 1 + 1 + 127 = 129 \tag{26}$$

   Su representación binaria es:

$$C_e = 10000001 \tag{27}$$

3. El valor de la mantiza C_m será el producto binario de $X_m * Y_m$ por lo tanto:

$$C_m = X_m{*}Y_m = 10010010000111111111001{*}01011011111011011111101 = 00010001010000111001000 \tag{28}$$

4. El bit de signo estará dado por

$$C_{signo} = (X * Y)_{signo} = X_{signo}\ \text{XOR}\ Y_{signo} = 0\ \text{XOR}\ 0 = 0$$

5. Por último se procede a realizar la concatenación de C_{signo}, C_e, C_m, dado lo anterior se tiene que:

$$C = 01000001000010001010000111001000_{IEEE754} = 8.539497_{10} \tag{29}$$

3 Posibles excepciones

Debido al hecho de que muchas de las operaciones propuestas en el algoritmo anterior obtendrán un resultado, el cual deberá ser almacenado dentro de las casillas designadas para su localización (es decir 8 bits para exponente y 23 para la mantiza) se corre el riezgo de que el resultado del producto de las mantizas de los operandos sea mucho más grande que $16,777,215$ (el valor máximo soportado) o que la suma sea mayor a 511. Este tipo de situaciones ya han sido tomadas en cuenta anteriormente por la IEEE y a raiz de lo anterior se listan las siguientes excepciones:

- **Overflow y underflow**: Sucede cuando el tamaño de la mantiza no es lo suficientemente grande para almacenar el resultado del producto; en estos casos se reinicia su valor de 11111111111111111111111 a 00000000000000000000000 lanzando una excepción.
- **Números denormales**: Si el resultado del cálculo del exponente en una multiplicación es menor que el exponente mínimo representable, puede ocurrir que el resultado sea un número denormalizado.

4 Bibliografía

References

[1] M. Morris Mano, *Arquitectura de computadoras*, Pearson Educación, 1994.

[2] M. Morris Mano, *Lógica digital y diseño de computadores*, Pearson Educación, 1982.

[3] David A. Patterson, John L. Hennessy, *Estructura y diseño de computadores*, Reverte, 2000.

[4] Francisco A. Martínez Gil, Gregorio Martín Quetglás, *Introducción a la programación estructurada en C*, Universitat de València, 2003.