Trabajo Práctico IRC

"Números flotantes en formato IEEE 754"

Asignatura: Instalación y Reemplazo de Componentes Internos

Curso: 6to Informática

Alumno: Laureano Diez

Profesor: Alejandro Rodríguez Costello

Julio 2021

Números flotantes en Simple Precisión

El objetivo del trabajo práctico es, a grandes rasgos, entender e interpretar el formato IEEE 754 a través del lenguaje assembler bajo el procesador QtSpim (MIPS). El mismo es un estándar técnico desarrollado por el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE es la sigla en inglés) introducido en 1985, con ajustes en 2008 y 2019. Predispone normas sobre el uso de comas flotantes (floating points) para facilitar la portabilidad a la hora de desarrollar distintos tipos de trabajos.

Valor Binario:

A través de las convenciones impuestas por el formato, podemos convertir números de coma flotante a sistema binario. El mismo consta de 32 bits, representado en 3 partes:

- **Signo**:

  El signo del número flotante se distingue con el **bit inicial**, siendo 0 si el mismo es **positivo** o un 1 si es **negativo**.

- **Exponente**:

  El exponente de un número binario ocupa **8 bits** del número binario, siendo el resultado del siguiente procedimiento:

  1. Convertir el valor entero del decimal flotante a binario
  2. Correr la coma, de derecha a izquierda, hasta encontrar un dígito 1 significativo
  3. Sumar la cantidad de veces que se corrió la coma a 127
  4. Calcular el binario del resultado del paso anterior

- **Mantisa**:

  La mantisa ocupa los **23 bits restantes** del número binario, y representa, a grandes rasgos, a la coma flotante del número decimal. Si bien la mantisa ocuparía realmente 24 bits, se obvia el 1 implícito al principio de la cadena.

  La mantisa es, esencialmente, un valor distinto de 0 siempre y cuando el valor del decimal con coma flotante no sea 0. Por ende, se puede decir que **la mantisa es realmente un valor arbitrario a modo de carga para el exponente**, resultado de ecuaciones logarítmicas complejas. Es por esto mismo que puede diferir en distintos casos.

  La forma para calcular la mantisa de un número de coma flotante más aproximada, es realizar su conversión a binario, multiplicando el valor de coma flotante del entero por 2 tantas veces hasta alcanzar un 1 sin flotante. El resultado de dicha ecuación deja consecuentemente un valor entero igual a 0 o 1. En caso de que deje un valor de 1 flotante, se multiplica por 2 su parte decimal. Entonces, el valor binario de la mantisa es equivalente a los resultados enteros que se obtienen en su orden de cálculo respectivo.

Por ejemplo, para convertir 2,25 a binario bajo el estándar **IEEE 754** siguiendo los pasos anteriores, obtenemos:

- **Signo**: 0, por ser positivo. Ocupa el primer lugar del número binario

  0 xxxxxxxx xxxxxxxxxxxxxxxxxxxxxxx

- **Exponente**: convertimos 2 a binario:

  $2_{10}=10_2$

  Como la coma del decimal está ubicada a la derecha del entero, la cantidad de movimientos hacia la izquierda para encontrar el 1 es de 1. Entonces, el exponente tiene un valor de 128. Al pasarlo a binario, nos queda:

  $128_{10}=10000000_2$

Entonces, el valor binario de 2,25 nos queda por ahora:

0 10000000 xxxxxxxxxxxxxxxxxxxxxxx

- **Mantisa**: convertimos 0.25 a binario:

  0.25x2 => 0.5 x 2 => 1

  => $0.25_{10}=01_2$

  Entonces, el valor binario del número flotante sería 10,01. Al haber movido la coma una posición hasta el primer 1, nos queda 1,001. Entonces, la mantisa es 001. Pero, al tener que ocupar 23 bits en el formato IEEE 754, debemos rellenar con 0 la cantidad de bits que hagan falta para completar el total. Entonces quedaría:

  $0.25_{10}=00100000000000000000000_2$

Como ya tenemos la mantisa, **podemos afirmar que el valor binario de 2,25 es de:**

0 10000000 00100000000000000000000

Valor Hexadecimal:

Para convertir un decimal a hexadecimal, resulta simple realizarlo con su valor en sistema binario. Para esto, agrupamos los bits de **derecha a izquierda**, en grupos de **cuatro dígitos**, hasta completar 8. Los mismos, en la misma dirección, van a ser multiplicados por dos a la potencia de su posición comenzando en cero, hasta tres. Para seguir con el ejemplo anterior, nos queda de la siguiente forma:

01000000000100000000000000000000 -> 0100 0000 0001 0000 0000 0000 0000 0000

<------

El primer grupo, consta de 4 ceros, por ende la suma de la ecuación ya detallada va a dar 0, por lo que hasta ahora tenemos un hexadecimal:

0xXXX00000

En el grupo número 6, tenemos un bit significativo:

0001 => (1x2^0=1) + (0x2^1=1) + (0x2^2=1) + (0x2^3=1) = **1**

Entonces, nuestro hexadecimal pasa a ser:

0xXX100000

Consecuentemente, el grupo 7 presenta ceros en sus cuatro posiciones, mientras que el grupo 8, y por ende el último, presenta un 1 en tercera posición:

0100 => (0x2^0=1) + (0x2^1=1) + (1x2^2=4) + (0x2^3=1) = **4**

Entonces, **el valor hexadecimal de 2,25 calculado bajo el formato IEEE 754 es**:

0x40100000

Desarrollo del programa:

Como consigna, se propuso desarrollar un programa en assembly en MIPS capaz de leer la entrada de un número flotante y convertirlo a binario y hexadecimal, bajo el formato IEEE 754. El desarrolló constó de aprendizajes a través de foros en línea y otras fuentes en distintos idiomas.

El programa convierte el valor ingresado a binario, mostrando su signo, exponente y mantisa. Luego, lo convierte en hexadecimal. El usuario puede optar por ingresar más valores para procesar, o ingresar un 0 para finalizar el programa. Utilizando los métodos explicados arriba, cada subrutina cumple la función necesaria para que el programa compilado se vea de la siguiente manera:

```
_____________________________________________________________


Ingrese un numero flotante: 2.25

Representación binaria: 01000000000100000000000000000000
Signo: Positivo (+)
Exponente: 128
Mantisa: 1.12500000
Representación hexadecimal: 0x40100000
_____________________________________________________________


Ingrese un numero flotante: 0

_____________________________________________________________


Cero Ingresado
```