J.T.P. Ing. Steiner Guillermo.
Email:gsteiner@scdt.frc.utn.edu.ar

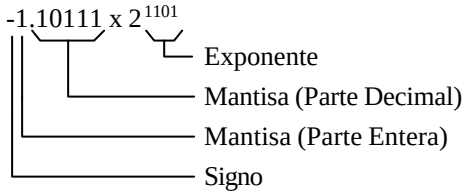
Trabajo Práctico Nro 6 Punto Flotante IEEE

Introducción

El punto flotante es un método muy útil para realizar operaciones numéricas cuando los números involucrados tienen una magnitud muy diferente, el microprocesador 8088 no posee en forma nativa la posibilidad de su uso, se deben recurrir a emuladores o coprocesadores para poder usarlos, en la serie x86 a partir del 80486DX y sus sucesores vino incorporado el coprocesador en el mismo microprocesador, por lo tanto es imposible contar hoy con una PC de escritorio sin manejo nativo de coma flotante, pero no pasa lo mismo con los microcontroladores que en su mayoría no lo soporta.
Es interesante entonces, conocer como es su estructura y uso en el lenguaje asembler, para luego poder ser utilizado en cualquier microprocesador o microcontrolador que no posee ninguna característica de manejo para estos números.

Componentes

Un número en punto flotante esta formado por:



Cada una de estas partes del punto flotante tendrán una ubicación determinada dentro de la codificación del número, y su longitud dependerá del tipo de número.

Exponente

El exponente de un número de punto flotante es modificado antes de su almacenamiento.
El número a almacenar en la posición del exponente será igual a:

$$e_{ieee} = e + 2^{m-1} - 1$$

Donde **e** será el exponente verdadero y **m** es el número de bits del exponente en el formato elegido
Esta modificación se realiza para evitar el uso de un bit de signo, lo que permite realizar comparaciones mas fácil entre exponentes, dado que estamos sumando 2^{m-1}-1 que es igual a sumar la mitad del máximo valor que se puede almacenar en el exponente, transformando el exponente en un número natural.
Ejemplo.
Si nuestro tamaño de exponente es de 8 bits, le estaremos sumando al exponente el numero 127.

Mantisa

La mantisa debe cumplir la condición de: 1 = mantisa < 2
Por lo tanto la parte entera será siempre 1 y solo será guardada en la número la parte decimal justificada a la derecha. (Un caso particular es el de 10-Byte real, en el cual el campo de parte entera existe, y obviamente valdrá siempre 1)

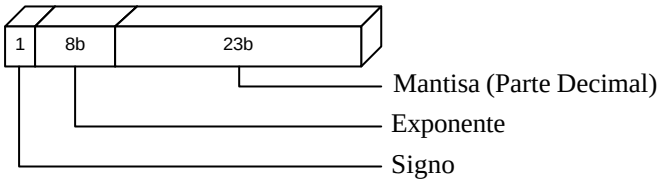
Signo

En el caso del punto flotante, en la mantisa se guarda siempre el valor absoluto de la misma, esto significa que en el caso de los numero negativos no se guarda el complemento a 2 del mismo sino simplemente el valor de la mantisa sin signo.
Solo el bit de signo identificará que tipo de número es
1 → Negativo
0 → Positivo

J.T.P. Ing. Steiner Guillermo.
Email:gsteiner@scdt.frc.utn.edu.ar

Tipos

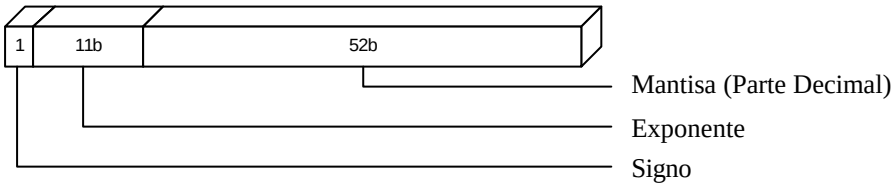
Short Real Number



Signo	1 bit
Exponente	8 bit
Mantisa (Parte Decimal)	23 bit
Total	32 bit (4 Bytes)

Como la parte entera es siempre 1 directamente lo asume y no lo incluye en la codificación

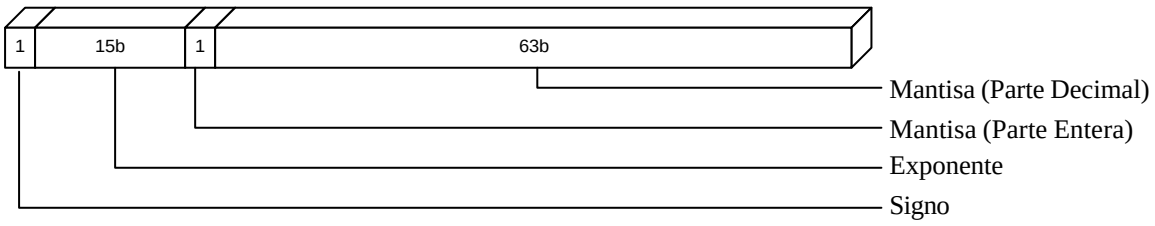
Long Real Number



Signo	1 bit
Exponente	11 bit
Mantisa (Parte Decimal)	52 bit
Total	64 bit (8 Bytes)

Como la parte entera es siempre 1 directamente lo asume y no lo incluye en la codificación

10-Byte Real Number



Signo	1 bit
Exponente	15 bit
Mantisa (Parte Entera)	1 bit
Mantisa (Parte Decimal)	63 bit
Total	80 bit (10 Bytes)

En este caso particular posee parte entera, pero esta es siempre 1 y por lo tanto no es un bit de información.

Aritmética de Punto Flotante

Pase de un número real a Short Real IEEE

Dado un número lo pasaremos a punto flotante en base 2
$x = 3{,}625$

J.T.P. Ing. Steiner Guillermo.
Email:gsteiner@scdt.frc.utn.edu.ar

$2^n = 3{,}625$

$n = \frac{\ln(3{,}625)}{\ln(2)} = 1{,}85798$

Separando parte entera y decimal del exponente

$2^{1{,}85798} = 2^{0.85798} \times 2^1 = 3{,}625$

$2^{0.85798} = 1{,}8125$

Resultado

El número será de la forma $(-1)^S \cdot f \cdot 2^e$

Al ser el número del ejemplo positivo entonces ya resolvimos la primera incógnita **s** = 0
f deberá ser un número cuya parte entera sea 1 por lo tanto debe cumplir que $1 <= f < 2$, el número obtenido de elevar 2 a la parte decimal de **n** da como resultado un número comprendido entre 1 y 2
La parte entera de **n** será entonces **e**

Pasamos el resultado a base 2

Parte entera = 1

Parte decimal

0,8125 x 2	=	1,625	Entero = 1
0,625 x 2	=	1,25	Entero = 1
0,25 x 2	=	0,5	Entero = 0
0,5 x 2	=	1	Entero = 1

$f_2 = 1{,}1101$

$x = (-1)^0 \cdot 1{,}1101 \cdot 2^1$

$e = 1$

$s = 0$

$f = 1{,}1101$

Representación en formato estándar IEEE (Short Real)

Signo = 0

Mantisa = $11010000000000000000000_2$

Exponente

$exp = 1 + 2^7 - 1 = 1 + 127 = 128$

$exp = 10000000_{\ 2}$

Resultado

$x = 3{,}625$

$x = 1{,}1101 \times 2^1$

x = 0_10000000_11010000000000000000000

$x = 40680000_{16}$

J.T.P. Ing. Steiner Guillermo.
Email:gsteiner@scdt.frc.utn.edu.ar

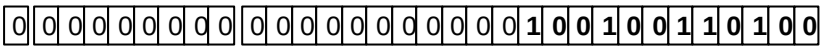
Pase de entero a Short Real IEEE

Describiremos aquí un método sencillo de pasar entero a pto flotante para ser aplicado a un programa.
A modo de ejemplo necesitamos almacenar el número **2356** en formato de punto flotante tipo Short Real de 32 bit.
La función a realizar, deberá transformar el formato de un entero al formato de un punto flotante, realizando los siguientes pasos:

- Cargar el bit de signo con el signo del número
- Guardar en la mantisa el valor ABSOLUTO del número
- Calcular el exponente y guardarlo en el campo correspondiente

Para el cálculo del exponente deberemos saber primero como estamos almacenando el número.

Hasta ahora tenemos



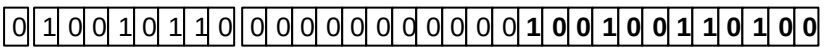
donde $2356_{10} = 100100110100_2$

La mantisa ingresada corresponde a la parte decimal del número por lo tanto el número nos quedaría como

$x = 1{,}0000000000\ \ 0100100110\ \ 1 \cdot 2^{e}$

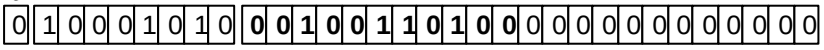
Esto significa que el exponente no puede ser 0, sino que será un número que me corra la coma a la derecha tantos lugares como longitud de mantisa tenga, para este caso serán 23, además sabemos que el valor a almacenar en el campo del exponente es

$$\exp = 23+127 = 150$$



Un vez echo esto debemos realizar otra operación llamada normalización, esta operación realiza sucesivos shift register a la izquierda al mismo tiempo que se decrementa al exponente hasta que el shift register provoque el desborde de un uno.
Esto es necesario porque como ya vimos este formato supone la existencia de un uno como parte entera, condición que no respetaba el número que ingresamos.

Quedando el número final como



Trabajo Práctico a Desarrollar

Realizar una función en Assembler en el cual tomando un número entero de 32 bit lo transforme a un número real short.
Utilizar la función para transformar un vector previamente cargado de al menos 5 valores distinto de enteros, este conjunto de números de prueba deberá incluir números positivos negativos y el cero.
NOTA IMPORTANTE: No es necesario realizar una interfaz con el usuario, simplemente se cargan en un vector los números a transformar y luego se visualiza los números transformados en debug.
Utilizar el programa en C adjunto para verificar los resultados.

```
#include <stdio.h>
int main(int argc, char* argv[])
{
	int num;
	unsigned int *aux;
	float numf;
	printf("Ingresar Nro a codificar:");
	scanf("%d",&num);
	numf = num;
	aux = (unsigned int *)&numf;
	printf("entero          = %d \n",num);
	printf("entero en hex   = %#X\n\n",num);
	printf("flotante        = %f \n",numf);
	printf("flotante en hex = %#X\n\n",*aux);
	return 0;
}
```