COMENTARIO TÉCNICO

Buceando en los MCUs Freescale.....

Por Ing. Daniel Di Lella
Dedicated Field Application Engineer
EDUDEVICES
www.edudevices.com.ar
dilella@arnet.com.ar

www.edudevices.com.ar

"Matemática de Punto Flotante"

Por el CETAD – UNLP
Coordinador Ing. Jose Rapallini
Participantes:
Sebastián Ledesma
e-mail: sledesma@barcala.ing.unlp.edu.ar
Federico Costantino,
e-mail: fcostant@barcala.ing.unlp.edu.ar
Jorge R. Osio,
e-mail: josio@gioia.ing.unlp.edu.ar

1era. Parte.

Nota: La presente serie de artículos está basada en el trabajo realizado por el Centro de Técnicas Analógicas – Digitales (CETAD) de la universidad Nacional de La Plata, y constituye una Nota de Aplicación para los MCU´s de 8 Bits de las familias HC908 y HC9S08 de Freescale Semiconductor.

Breve Descripción:

En esta nota se describen rutinas de multiplicación, división, suma y resta de números en formato de punto flotante, siguiendo las especificaciones de punto flotante descriptas por la Norma 754 de la IEEE [1]. En esta nota las operaciones se realizan con valores en punto flotantes en simple precisión, es decir, que cada número se representa con 32 bits.

Estas rutinas son fácilmente exportables a los MCU´s de las familias HC908 o HC9S08 que son de 8 Bits y que naturalmente no están diseñados para soportar matemática de punto flotante como lo harían procesadores más complejos.
La comunicación para el ingreso de datos se realiza mediante el módulo SCI (UART – Interface serial asincrónica).

Descripción técnica detallada:

Introducción.

Descripción de la Norma IEEE 754.

El estándar de la IEEE para aritmética en punto flotante **(IEEE 754)** es el estándar más ampliamente usado en las operaciones computacionales de punto flotante, y es utilizada por muchas de las implementaciones de CPU y FPU[i]. El estándar define formatos para la representación de números en punto flotante (incluyendo el cero) y valores desnormalizados, así como valores especiales como infinito y NANs[ii]) conjuntamente con un conjunto de operaciones en punto flotante que opera sobre estos valores. También especifica cuatro modos de redondeo y cinco excepciones (incluyendo cuando dichas excepciones ocurren, y que sucede en dichos momentos).

La Norma **IEEE 754** especifica cuatro formatos para la representación de valores en punto flotante: precisión simple (32-bits), precisión doble (64-bits), precisión simple extendida (≥ 43-bits, no usada normalmente) y precisión doble extendida (≥ 79-bits, usualmente implementada con 80-bits). Solo se requieren los valores de 32-bits, los otros son opcionales. Muchos lenguajes especifican que formatos y aritmética de la IEEE implementan, a pesar que a veces son opcionales. Por ejemplo, el lenguaje de programación C, permite utilizar la aritmética de describe la IEEE (el tipo float de C se usa para números en formato de precisión simple de la IEEE y el tipo double usa la precisión doble del la IEEE).

El título completo del estándar es IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Std 754-1985), y también es conocido por IEC 60559:1989, Binary floating-point arithmetic for microprocessor systems (originalmente el número de referencia era IEC 559:1989).

Anatomía de un número en punto flotante

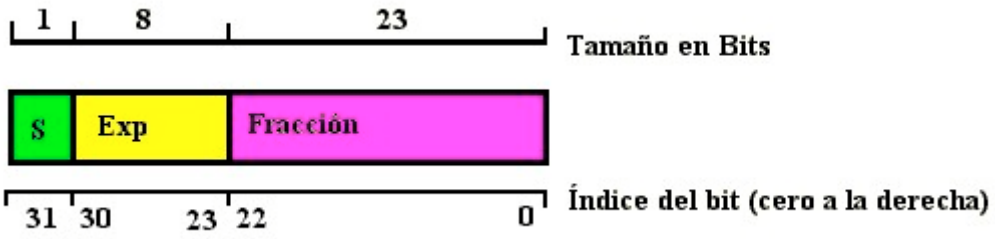
A continuación se realiza una descripción del formato estándar de la IEEE para números de punto flotante.

Convenciones de Bit

Los Bits dentro de una palabra de tamaño W están indexados por enteros en el rango 0 a W−1 inclusive. El bit cuyo índice es 0 se sitúa a la derecha. El menor bit indexado es normalmente el menos significativo.

Precisión simple 32-bits

Un número en punto flotante de precisión simple se almacena en una palabra de 32 bits



Donde **S** es el bit de signo y **Exp** es el campo exponente. (Para el signo: 0 = Positivo; 1= Negativo). El exponente se desplaza un bit a la derecha, ya que tiene un bit de signo.

El valor almacenado es el offset (desplazado 127) del valor actual. El desplazamiento se realiza porque los exponentes pueden ser valores con signo, para permitir la representación de números pequeños (menores a 1) y grandes (mayores a 1).

Para resolver esto, se desplaza el exponente antes de almacenarlo, ajustando su valor en un rango sin signo para poder compararlo con otro. Así, para un número en precisión simple, un exponente en el rango −126 a +127 se desplaza mediante la suma del número 127 para obtener un valor en el rango de 1 a 254 (0 y 255 tienen significados especiales descritos más adelante). Una vez que se realizan las operaciones deseadas en punto flotante, se vuelve a desplazar el número para obtener el exponente real con signo.

Para representar un número en punto flotante, los *bits* se acomodan del siguiente modo:

Signo del número real S:	1 bit
Exponente (entero |Exp|):	8 bits
Mantisa (número real |fracción|):	23 bits

El conjunto de valores posibles pueden ser divididos en los siguientes:

- ceros
- números normalizados
- números desnormalizados
- infinitos
- NaN (por ejemplo, la raíz cuadrada de un número negativo)

Las clases se distinguen principalmente por el valor del campo **Exp**, siendo modificada esta por el campo **fracción**. Se consideran **Exp** y **Fracción** como campos de números binarios sin signo (**Exp** se encuentra en el rango 0–255):

Clase	Exp	Fracción
Ceros	0	0
Números desnormalizados	0	distinto de 0
Números normalizados	1-254	cualquiera
Infinitos	255	0
NaN (no es un número)	255	Distinto de 0

El valor decimal V de un número en punto flotante viene dado por:

$$V = (-1)^S * 2^E * M$$

Donde:

S = 0 (números positivos) o S = 1 (números negativos).
E = $Exp - 127$ (para normalizados) o E = -126 (para desnormalizados).
M = 1,*Fracción* (en binario para normalizados) ; en este caso $1 \leq M < 2$,
o M = *0,Fracción* (en binario para desnormalizados)con lo cual $M < 1$.

Características importantes de la Norma de punto flotante:

1. −126 es el menor exponente para un número normalizado.
2. Hay dos ceros. +0 (S es 0) y −0 (S es 1).
3. Hay dos infinitos +∞ (S es 0) y −∞ (S es 1).
4. Los NaNs pueden tener un signo y un significando, pero estos no tienen otro significado que el que puedan aportar en pruebas de diagnostico; el primer bit del significando es a menudo utilizado para distinguir los *NaNs señalizados* (SnaN: Signalling Not a Number) de los *NaNs silenciosos* (QNaN: Quiet Not a Number). Los primeros es utilizado cuando el dato ingresado al sistema no tiene el formato de un número válido especificado por la norma de punto flotante. Los QNaN especifican que la operación realizada con las cifras ingresadas produce un valor numérico no válido.
5. Los NaNs y los infinitos tienen todos los bits puestos a 1 en el campo Exp.

Ejemplo:

Para entender mejor lo antes dicho se codifica el número decimal -118.625 usando el sistema de la IEEE 754. Para esto se debe obtener el signo, el exponente y la fracción.

- Dado que es un número negativo, el signo es "1".
- Se escribe el número (sin signo) usando notación binaria. El resultado es 1110110.101.
- Se mueve el punto decimal a la izquierda, dejando solo un 1 a su izquierda. $1110110.101 = 1.110110101 \cdot 2^6$, este es un número en punto flotante normalizado.
- La fracción es la parte que está a la derecha del punto decimal, se llena con ceros a la derecha hasta obtener todos los 23 bits. Es decir 11011010100000000000000.
- El exponente es 6, pero se lo debe convertir a binario y desplazarlo (de forma tal que todos los exponentes son solamente números binarios sin signo).
  Para el formato IEEE-754 de 32 bits, el desplazamiento es 127, así es que 6 + 127 = 133. En binario, esto se escribe como 10000101

Poniendo todo junto se tiene el número en punto flotante normalizado:

Exp desplazado + 127

Comparación de números en punto flotante

La comparación de números en punto flotante se realiza generalmente usando instrucciones de punto flotante. Sin embargo esta representación (IEEE 754) hace la comparación de determinados subconjuntos posible byte-por-byte, si comparten el mismo orden de bytes y el mismo signo, y los NaN s son excluidos.

Por ejemplo, para dos números positivos a y b, a < b es cierto siempre que los enteros binarios sin signo con los mismos patrones de bits y el mismo orden de bytes que a y b, son también ordenados de forma a < b. En otras palabras, dos números positivos (que se sabe que no son NaN s) pueden ser comparados con una comparación entre enteros binarios sin signo entre los mismos grupos de bits, teniendo como base que los números tienen el mismo orden de bytes.

Redondeo de números en punto flotante

El estándar de la IEEE tiene cuatro formas diferentes de redondeo:

- **Unbiased** que redondea al número más cercano, si el número cae en medio, este es redondeado al valor más cercano con un valor par (cero) en su menos significativo. Este modo es el requerido como por defecto.
- **Hacia el cero**
- **Hacia el infinito positivo**

- **Hacia el infinito negativo**

Operaciones matemáticas en punto flotantes.

Como se ha visto, los números en punto flotante se representan por tres campos: significando ó fracción del significando, signo del significando y el exponente.

Las operaciones de multiplicación y división son más fáciles de efectuar que las de suma y resta [3], como veremos a continuación.

Algoritmos de operaciones:

Algoritmo de multiplicación.

Se deben realizar varias verificaciones antes de realizar el producto de las mantisas:

Verificar NaN: Si alguno de los argumentos es NaN (Q o S), el resultado es SNaN.

Verificar 0 * ±INFINITO: Si un argumento es 0 y el otro es INFINITO, el resultado es QNaN.

Verificar INFINITO: Si alguno de los argumentos es INFINITO, el resultado es INFINITO con el signo que le corresponda:

Verificar argumento 0: Si alguno de los argumentos es 0, el resultado es 0 (x * 0 = 0).

Suma de exponentes:

Los exponentes se deben sumar algebraicamente, pero como éstos están formateados con el bias, se deben antes desnormalizar y luego sumar. El resultado de esa suma se debe luego formatear con el bias.

Cuando una cifra es desnormalizada (exponente = 0) la suma se efectúa de la misma manera

Si el resultado de la suma se va por encima del máximo (+127 = FE) permitido por el rango, la operación será infinita. Si el resultado se va por debajo del mínimo (-126 = 01) entonces la operación será cero.

Multiplicar mantisas:

Los significandos se deben multiplicar, pero éstos están representados sólo con la parte decimal, donde tácitamente el entero es un "1" (para el caso de cifras normalizadas). Por lo tanto antes de utilizar la mantisa se debe poner un uno al principio de esta. Al resultado se lo vuelve a expresar con solo la parte decimal.

Para el caso de cifras desnormalizadas se desplazará la mantisa solo un lugar hacia la izquierda para que ambas cifras a multiplicar empiecen en el mismo bit. Con respecto al primer bit que se seteo anteriormente este será perdido al desplazar la mantisa, por lo que no me preocupa.

El signo resultado, es "+" si ambos argumentos tienen el mismo signo y "-" en caso contrario

Normalizar resultado:

Primero se ve si el exponente es mayor o igual a –126. Si es así se busca que en el MSB del segundo byte quede en uno. Para esto hacemos lo siguiente
:

- Si el resultado tiene la forma [01,MM...] (cifra en el segundo byte) se correrá la mantisa dos lugares hacia la izquierda para que el 1 se elimine ya que quedará en forma implícita. No se incrementa el exponente.
- Si el resultado tiene la forma [10,MM...] (cifra en el segundo byte) se correrá la mantisa un lugar hacia la izquierda por la misma razón que la anterior. Ahora si se incrementará el exponente

- Si la cifra es desnormalizada (exponente = 0) ajustamos la mantisa (desplazamos hacia la izquierda) hasta que quede de la forma 1,x...x. Cada vez que corremos la mantisa decrementamos el resultado de la suma de exponentes. Luego se desplaza una ves mas pero sin decrementar el exponente.

Datos a tener en cuenta al hacer rutina de multiplicación:

- Si el resultado tiene 48 bits entonces habrá dos bits en la parte entera ([1x,xxxxxx]). En este caso hay que **incrementar el exponente** y **correr toda la mantisa 1 lugar a la izquierda**, quedando ([1,xxxxxxxx]).
- Si el resultado tiene 47 bits la mantisa es del tipo [01,xxxxxx]. Entonces **NO se incrementa** el exponente pero **se desplaza la mantisa 2 lugares a la izquierda.**
- En ambos casos anteriores al resultado hay que descartarle los últimos 24 bits (3 bytes) ya que la palabra de la mantisa es de solo 24 bits. Aquí se generará un error: ERROR DE TRUNCAMIENTO.
- Una vez obtenido el exponente se lo desplazará hacia la derecha un lugar y su LSB ocupara el primer bit del segundo byte.
- Si uno de las cifras es desnormalizada ajustamos la mantisa (desplazamos hacia la izquierda) hasta que quede de la forma 1,x...x. Cada vez que corremos la mantisa decrementamos el resultado de la suma de exponentes.

Algoritmo de división:

Se deben realizar varias verificaciones antes de realizar el cociente de las mantisas:

Verificar NaN: Si alguno de los argumentos es NaN (Q o S), el resultado es SNaN.

Verificar 0 / 0: Si ambos argumentos son 0, el resultado es QNaN.

Verificar INFINITO / INFINITO: Si ambos argumentos son INFINITO, el resultado es QNaN.

Verificar INFINITO / X: Si el argumento dividendo es INFINITO, el resultado es INFINITO.

Verificar X / INFINITO: Si el argumento divisor es INFINITO, el resultado es 0.

Verificar "no 0" / 0: Si el argumento divisor es 0, el resultado es INFINITO con el signo del dividendo

Verificar argumento 0: Si el argumento dividendo es 0, el resultado es 0 ($0 / x = 0$; con $x \neq 0$).

Resta de los exponentes:

Los exponentes se deben restar algebraicamente, pero como éstos están formateados con el bias, se deben antes desnormalizar y luego restar. El resultado de esa resta se debe luego formatear con el bias.

Si el resultado de la resta se va por encima del máximo (+127 = FE) permitido por el rango, la operación será infinita. Si el resultado se va por debajo del mínimo (-126 = 01) entonces la operación será cero.

Dividir mantisas:

La mantisa resultado, es el cociente de las mantisas.. El signo resultado, es "+" si ambos argumentos tienen el mismo signo y "-" en caso contrario.

El caso de **división** es análogo, donde puede ser que al dividir los significados el resultado de menor que 1, y se deba correr la coma hacia la derecha, es decir, restar un "1" al resultado de restar los exponentes.

Normalizar resultado (si es necesario): Dado que las mantisas normalizadas son mayores o iguales a 1 y menores a 2, el cociente es mayor a 0.5 y menor a 2: por lo que sólo puede ser necesario un desplazamiento a la izquierda (el exponente del resultado debe ajustarse en forma acorde).

Si el resultado queda en forma desnormalizada puede que la mantisa llegue a ser desplazada hasta 23 lugares a la derecha. El número de desplazamiento estará dado por el número de incrementos que se le hará al exponente hasta llegar al cero.

Algoritmos de suma y resta:

Recomendaciones antes de sumar o restar números en punto flotante [4]:

- A+(B+C) suele ser distinto a (A+B)+C
- En el caso de las sumas, es conveniente sumar los valores "ordenados" de menor a mayor (no es necesario un orden estricto).
- Al sumar 2 números muy diferentes puede ser que la operación no sea necesaria ya que el sumando mas chico no llega a afectar al mayor.
- Al restar 2 números muy próximos el resultado que se obtendrá, puede ser todo error de redondeo.

Algoritmo de suma y resta.

Se deben realizar varias verificaciones antes de realizar la suma o resta de las mantisas.

Verificar NaN: Si alguno de los argumentos es NaN (S o Q), el resultado es SNaN.

Verificar INFINITO - INFINITO: Si ambos argumentos son INFINITO, pero de distinto signo, el resultado es QNaN.

Verificar + INFINITO: Si alguno de los argumentos es +INFINITO, el resultado es +INFINITO.

Verificar - INFINITO: Si alguno de los argumentos es -INFINITO, el resultado es -INFINITO.

Verificar argumento 0: Si alguno de los argumentos es 0, el resultado es el "otro" argumento (x + 0 = x).

Verificar diferencia de exponentes:

Si la diferencia entre los exponentes de los argumentos es mayor a los bits de la mantisa, el resultado es el "mayor" argumento (ya que el argumento menor no logra afectar el resultado). Por ejemplo si un número tiene exponente +12 y otro -15 (la diferencia es de 27) y se deben sumar, se estaría fuera de la capacidad de representación en punto fijo simple precisión donde sólo se puede representar la fracción del significando con 23 bits.

Igualar exponentes (si es necesario):

Sólo es posible realizar estas operaciones cuando los números en punto flotante tienen el mismo exponente. Se debe desplazar a la derecha la mantisa del argumento con menor exponente la cantidad de bits necesarios hasta igualar los exponentes, lo cual implica que se perderá resolución.

Complementar (si es necesario):

Si los argumentos tienen distinto signo, se debe complementar la mantisa del de menor exponente.

En el caso de que la operación fuera resta se deberá cambiar el signo del sustraendo con el fin de plantear la operación como una suma.

Sumar mantisas:

La mantisa resultado, es la suma de las mantisas. El exponente del resultado, es el mayor de ambos. El signo resultado es el signo del de mayor exponente.

Normalizar resultado (si es necesario):

El resultado se debe volver a normalizar, teniendo en cuenta que debe ser de la forma 1,x.....x (precisión simple y doble), pudiendo ser necesario ajustar de nuevo los exponentes.
Si la mantisa obtenida es 0, el resultado es 0.
Si la mantisa es mayor o igual a 2, debe desplazarse a derecha y aumentar el exponente del resultado en forma acorde. Si la mantisa es menor a 1, debe desplazarse a izquierda y decrementar el exponente del resultado en forma acorde.

Continuará.....
Nota de Radacción: El lector puede descargar este artículo y artículos anteriores de ***"Buceando..."*** desde la sección ***"Artículos Técnicos"*** en el sitio web de **EduDevices (**www.edudevices.com.ar **)**

[i] FPU: **Unidad de Punto Flotante** (o **Floating Point Unit** en inglés) es un componente de la CPU especializado en el cálculo de operaciones en punto flotante.

[ii] NAN: Acrónimo inglés de "**Not a Number"** (no es un número) que se usa para referirse, en ámbitos informáticos, a representaciones de indeterminaciones, raíces de números negativos, etc. En general, simboliza cualquier operación cuyo resultado no puede expresarse con un valor numérico válido.