DCC

DEPARTAMENTO DE CIENCIA
DE LA COMPUTACIÓN

IIC2343 - Arquitectura de Computadores

Clase 11 - Aritmética de números de punto flotante

Susana Figueroa

Suma

- Para poder sumar números de punto flotantes, necesitamos primero fijarnos en su composición.
- Como el número se compone de un bit de signo, los 8 siguientes el exponente y luego la mantisa, la operacion de suma bit a bit no nos va a servir para estos numeros.

$$1.0010 \times 2^{5} + 1.01 \times 2^{-1}$$

Suma

- Para poder sumar números de punto flotantes, necesitamos primero fijarnos en su composición.
- Como el número se compone de un bit de signo, los 8 siguientes el exponente y luego la mantisa, la operacion de suma bit a bit no nos va a servir para estos numeros.
- Lo primero que vamos a tener que hacer es igualar los exponentes al mayor, esto es para que los órdenes de la mantisa coincidan
- Como vamos a igualar los exponentes? Haciendo shift rights sobre la mantisa del número con menor exponente.

$$1.0010 \times 2^{5} + 1.01 \times 2^{-1}$$

$$1.0010 \times 2^{5} + 0.00000101 \times 2^{5}$$

En este caso fue necesario hacer shift rights 6 veces.

Suma

- Ahora que ambos números se encuentran con el mismo exponente se pueden sumar las mantisas.

$$1.0010 \times 2^{5} + 1.01 \times 2^{-1}$$

$$1.0010 \times 2^{5} + 0.00000101 \times 2^{5}$$

$$= 1.00100101 \times 2^{5}$$

Suma

- Luego normalizamos en caso de ser necesario (en nuestro ejemplo no lo es).
- Redondeamos. Hay que acordarnos que nuestra representación tiene una cantidad finita de bits para la mantisa. Es posible que al sumar dos números, el resultado de esta suma exceda la cantidad representable, en este caso es necesario redondear. Si consideramos que son floats según el estándar IEEE-754, en el ejemplo no es necesario redondear dado que tenemos 24 bits para la mantisa (considerando el bit implicito).

$$1.0010 \times 2^{5} + 1.01 \times 2^{-1}$$

$$1.0010 \times 2^{5} + 0.00000101 \times 2^{5}$$

$$= 1.00100101 \times 2^{5}$$

Algoritmo de suma

- Comparar los exponentes. Hacer shift right a la mantisa del número más pequeño hasta que su exponente sea igual al del número más grande.
- Sumar las mantisas.
- Normalizar la suma. Esto puede ser haciendo shift right e incrementando el exponente, o shift left y decrementar el exponente. Depende del caso.
- Redondear el significante al número apropiado de bits. Revisar nuevamente si está normalizado, en caso que no lo esté, volver al paso anterior.

Algoritmo de multiplicación

- Sumar los exponentes desfasados y restar el desfase a la suma para obtener el nuevo exponente desfasado.
- Multiplicar las mantisas.
- Normalizar el producto. Esto puede ser haciendo shift right e incrementando el exponente, o shift left y decrementar el exponente. Depende del caso.
- Redondear el significante al número apropiado de bits. Revisar nuevamente si está normalizado, en caso que no lo esté, volver al paso anterior.
- Verificar el signo. Si los signos de ambos números es el mismo, es positivo. Si los signos son distintos, el producto es negativo. Para asignar el nuevo signo debemos hacer un XOR de los signos originales.

Algoritmo de multiplicación: Ejemplo

$$(1.0000 \times 2^{-1}) \times (-1.11 \times 2^{-2})$$

- Sumar los exponentes desfasados y restar el desfase a la suma para obtener el nuevo exponente desfasado.

$$(-1 + 127) + (-2 + 127) - 127$$
$$= -3 + 127 = 124$$

Algoritmo de multiplicación: Ejemplo

$$(1.0000 \times 2^{-1}) \times (-1.11 \times 2^{-2})$$

- Sumar los exponentes desfasados y restar el desfase a la suma para obtener el nuevo exponente desfasado.

$$(-1 + 127) + (-2 + 127) - 127$$
$$= -3 + 127 = 124$$

- Multiplicar las mantisas.

$$1.0000 \times -1.11 = 1.11$$

Algoritmo de multiplicación: Ejemplo

$$(1.0000 \times 2^{-1}) \times (-1.11 \times 2^{-2})$$

- Sumar los exponentes desfasados y restar el desfase a la suma para obtener el nuevo exponente desfasado.

$$(-1 + 127) + (-2 + 127) - 127$$
$$= -3 + 127 = 124$$

- Multiplicar las mantisas.

$$1.0000 \times -1.11 = 1.11$$

- Redondear y normalizar no tienen efecto en este caso.

Algoritmo de multiplicación: Ejemplo

$$(1.0000 \times 2^{-1}) \times (-1.11 \times 2^{-2})$$

- Sumar los exponentes desfasados y restar el desfase a la suma para obtener el nuevo exponente desfasado.

$$(-1 + 127) + (-2 + 127) - 127$$
$$= -3 + 127 = 124$$

- Multiplicar las mantisas.

$$1.0000 \times -1.11 = 1.11$$

- Redondear y normalizar no tienen efecto en este caso.
- Solo queda verificar el signo. En este caso como tienen signos distintos es negativo.

Algoritmo de multiplicación: Ejemplo

$$(1.0000 \times 2^{-1}) \times (-1.11 \times 2^{-2})$$

- El resultado de la multiplicación queda...

$$-1.11 \times 2^{-3}$$

- Como se vería según el estándar?

1 01111100 11000000000000000000000

DCC

DEPARTAMENTO DE CIENCIA
DE LA COMPUTACIÓN

IIC2343 - Arquitectura de Computadores

Clase 11 - Aritmética de números de punto flotante

Susana Figueroa