Universidad del País Vasco
Facultad de Informática

Departamento de Arquitectura y Tecnología de Computadores

Arquitectura de Computadores I

Aritméticos 8 (solución): coma flotante

Representa el número *-10,98* en coma flotante, utilizando un bit de signo, 7 bits para la mantisa y 4 bits para el exponente. Representa la mantisa normalizada según la técnica del bit oculto. Representa el exponente según la representación "*exceso 7*". Representa el número según las técnicas *inmediato superior, inmediato inferior, truncamiento* y *redondeo,* calculando el error obtenido en cada caso.

Solución

Para empezar, diferenciaremos las tres partes del número: signo, mantisa y exponente. Los calcularemos en el mismo orden en que los hemos escrito.

Puesto que es un número negativo, el bit de signo será `1`.

En lo que se refiere a la mantisa, a la hora de pasarla a binario se diferencian la parte entera y la parte fraccionaria. Para obtener la parte entera hay que hacer divisiones consecutivas, mientras que para obtener la parte fraccionaria hay que hacer multiplicaciones consecutivas. Las divisiones consecutivas de la parte entera son las siguientes:

10 entre 2: resultado 5 y resto 0.
5 entre 2: resultado 2 y resto 1.
2 entre 2: resultado 1 y resto 0.
1 entre 2: resultado 0 y resto 1.

Por lo tanto el valor 10 en binario se representa mediante la secuencia de bits `1010`, concatenando los restos obtenidos de derecha a izquierda en el orden obtenido.

Las multiplicaciones consecutivas de la parte fraccionaria son las siguientes:

0,98 por 2: el resultado es 1,96.
La parte entera nos indica el valor del siguiente bit y la parte fraccionaria la utilizaremos para calcular los bits siguientes.
0,96 por 2: el resultado es 1,92.
Otra vez, separaremos la parte entera de la fraccionaria.
0,92 por 2: el resultado es 1,84.
0,84 por 2: el resultado es 1,68.
0,68 por 2: el resultado es 1,36.
0,36 por 2: el resultado es 0,72.
0,72 por 2: el resultado es 1,44.

Podríamos seguir haciendo multiplicaciones, puesto que no es posible representar este número en binario con un número finito de bits. Concatenando los bits obtenidos hasta ahora, la mantisa nos queda de la siguiente manera:

`1010,1111101...`

Ese número no está normalizado. Normalizándolo nos queda de la siguiente manera:

$1,0101111101... * 2^3$

Si utilizamos la técnica del bit oculto, tal y como se indica en el enunciado, no guardaremos lo que hay en la parte entera, pero tendremos que tenerlo en cuenta a posteriori a la hora de interpretar este número. Ahora tenemos que calcular cuáles serán los 7 bits que utilizaremos para representar ese número. Puesto que es un número negativo, en los casos *inmediato superior* y *truncamiento* obtendremos la misma secuencia de bits, mientras que para el *inmediato inferior* se obtendrá otra secuencia de bits. En cuanto al *redondeo* se refiere, el valor del bit de mayor peso no considerado determinará que la representación coincida con una u otra de las anteriores.

Por lo tanto, en los casos *inmediato superior* y *truncamiento*, los 7 bits de la mantisa son `0101111`, puesto que el número inmediatamente superior, o dicho de otra manera, el número superior más pequeño es el que se representa quitando el resto de los bits. Si fuera un valor positivo esto no se cumpliría.

En cambio, en el caso del *inmediato inferior*, los 7 bits de la mantisa son `0110000`, puesto que ese es el mayor número que se puede representar en valor negativo por debajo del número que realmente queremos representar.

En el caso del *redondeo* tendremos que hacer un cálculo para ver cuál será la secuencia de bits que utilizaremos: sumándole la mitad de la base (puesto que estamos utilizando la base 2, será un 1) al bit de mayor peso de entre los que no vamos a utilizar se obtiene la secuencia de 7 bits que vamos a utilizar:

```
0101111101...
      +1
--------
01100000
```

Como se ve, en este caso, la sucesión de bits obtenida es igual a la del *inmediato inferior*, la secuencia `0110000`.

En cuanto al exponente se refiere, en cualquiera de los casos el valor a representar es 3, utilizando la representación *exceso 7* en 4 bits. Si hacemos *3 más 7*, la representación será 10 en decimal y en binario se representa mediante la secuencia de bits `1010`, tal y como hemos visto anteriormente (al calcular la parte entera de la mantisa).

Resumiendo, después de todos los cálculos realizados, en el caso de *inmediato superior* y de *truncamiento* la secuencia de bits para representar ese número es la siguiente (el signo, la mantisa y el exponente):

`1 0101111 1010`

Por otro lado, en el caso de *inmediato inferior* y *redondeo*, la secuencia de bits es ésta:

`1 0110000 1010`

Ahora nos falta calcular los errores. Diferenciaremos las dos representaciones que hemos obtenido, para ver cuáles son realmente los números que se han representado.

La secuencia de bits `1 0101111 1010` indica el negativo del valor absoluto representado por la sucesión `1010,1111`. En este caso, el valor absoluto es el siguiente: $2^3+2^1+2^{-1}+2^{-2}+2^{-3}+2^{-4}=$ 10,9375. Por lo tanto, el valor real representado con esta sucesión de bits es -10,9375. Para calcular el error, se utilizan los valores absolutos, restando al mayor el menor valor. Por lo tanto, si le restamos 10,9375 al valor 10,98 el error obtenido para las representaciones *inmediato superior* y *truncamiento* es 0,0425.

Por otra parte, la secuencia de bits `1 0110000 1010` indica el negativo del valor absoluto representado por la sucesión `1011,0000`, exactamente: $2^3+2^1+2^0=11$. Por lo tanto, el valor representado realmente es el -11. En cuanto al error se refiere, restándole 10,98 al valor 11, obtenemos un error de 0,02 para los casos *inmediato inferior* y *redondeo*. Como se puede ver, el error del *redondeo* es menor que el otro, tal y como se nos ha dicho en la teoría.