Arquitetura de Computadores
Aulas Práticas 2017/2018

2. Representação de Números em Vírgula Flutuante

Formato IEEE754 para representação binária de números em vírgula flutuante:

	#bits			
	sinal (s)	expoente (e)	mantissa (m)	total
precisão simples (*single*)	1	8	23	32
precisão dupla (*double*)	1	11	52	64

Em precisão simples, o significado dos bits é o seguinte:

sinal (s)	expoente (e)	mantissa (m)	número
s	$0\ldots0$	$0\ldots0$	zero
s	$0\ldots0$	$a_1\ldots a_{23}$	número não normalizado, $a_i \in \{0,1\} \wedge \exists i : a_i \neq 0$
s	$0\ldots01$ - $1\ldots10$	$a_1\ldots a_{23}$	número normalizado, $a_i \in \{0,1\}$
s	$1\ldots1$	$0\ldots0$	infinito
s	$1\ldots1$	$a_1\ldots a_{23}$	NaN (Not a Number), $a_i \in \{0,1\} \wedge \exists i : a_i \neq 0$

No caso de números normalizados (precisão simples), o número real x representado pelo formato IEEE754 é:

$$x = (-1)^s \times (1 + m(10)) \times 2^{e(10)-127}$$

No caso de números não normalizados (precisão simples), o número real x representado pelo formato IEEE754 é:

$$x = (-1)^s \times m(10) \times 2^{-127+1}$$

Em precisão dupla, o expoente (e) utiliza 11 bits (em vez de 8), a mantissa (m) utiliza 52 bits (em vez de 23) e o desvio no expoente é de 1023 (em vez de 127).

Exemplo 1: Representação do número real 6.6875 no formato IEEE754 precisão simples:

$$6.6875(10) = 6 + 0.6875(10)$$
$$6(10) = 110(2)$$

$$2 \times 0.6875 = \mathbf{1}.375$$
$$2 \times 0.375 = \mathbf{0}.75$$
$$2 \times 0.75 = \mathbf{1}.50$$
$$2 \times 0.5 = \mathbf{1}.0 \qquad \text{(parar quando a parte fracionária é 0)}$$

$$0.6875(10) = 0.1011(2)$$
$$6.6875(10) = 110.1011(2) = 1.101011 \times 2^2 \qquad \text{(representação em vírgula flutuante)}$$

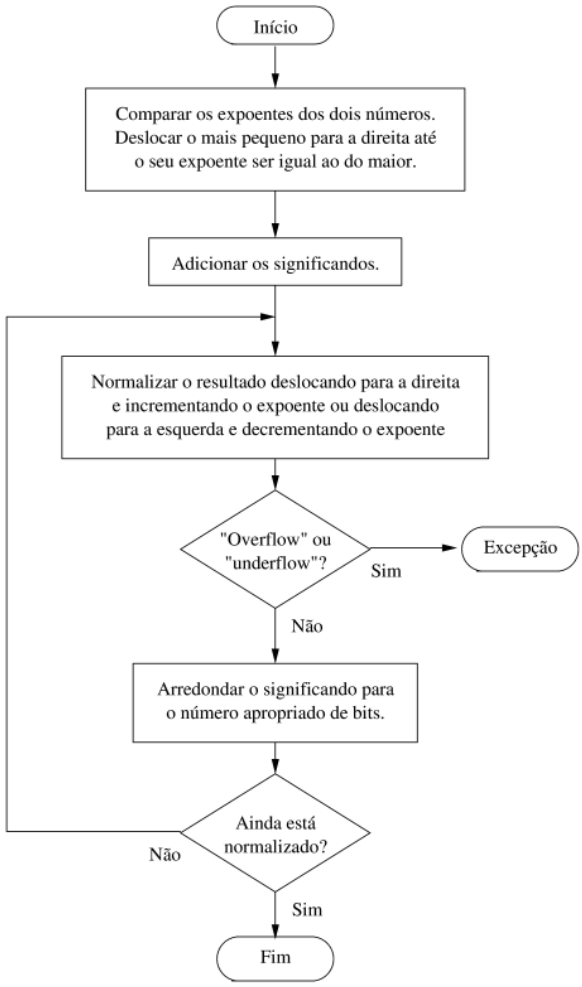
$$s = \mathbf{0}$$
$$e - 127 = 2$$
$$e = 129 = \mathbf{10000001}(2) \qquad \text{(8 bits em precisão simples)}$$
$$m = \mathbf{101011}00000000000000000 \qquad \text{(23 bits em precisão simples)}$$

$$6.6875(10) = \mathbf{0\ 10000001\ 101011}00000000000000000 \qquad \text{(representação no formato IEEE754)}$$

e vice-versa:

$$\begin{aligned} 1.\mathbf{101011} \times 2^2 &= (-1)^0 \times (1 + \mathbf{1} \times 2^{-1} + \mathbf{0} \times 2^{-2} + \mathbf{1} \times 2^{-3} + \mathbf{0} \times 2^{-4} + \mathbf{1} \times 2^{-5} + \mathbf{1} \times 2^{-6}) \times 2^2 \\ &= (1 + 1/2 + 1/8 + 1/32 + 1/64) \times 4 \\ &= 6.6875(10) \end{aligned}$$

A figura abaixo apresenta o algoritmo de adição em vírgula flutuante para números no formato IEEE754:



Exemplo 2: Aplicação do algoritmo de adição à operação $0.90625 + 0.1875$:

$$
\begin{aligned}
0.90625(10) &= 1.1101 \times 2^{-1} \\
0.1875(10) &= 1.1 \times 2^{-3} \\
0.1875(10) &= 0.011 \times 2^{-1} && \text{(acertar para o maior expoente)} \\
1.1101 + 0.0110 &= 10.0011 && \text{(fazer a adição)} \\
10.0011 \times 2^{-1} &= 1.00011 \times 2^{0} && \text{(normalizar, não é preciso arredondar)}
\end{aligned}
$$

Representação do resultado no formato IEEE754 precisão simples:

$$1.00011 \times 2^0 = 0\ 01111111\ 00011000000000000000000$$

Regras de arredondamento:

- Arredondamento em direção ao zero (*round toward 0*): despreza os bits excedentes (i.e., não faz nada).
- Arredondamento em direção ao infinito positivo (*round toward* $+\infty$): arredonda para cima.
- Arredondamento em direção ao infinito negativo (*round toward* $-\infty$): arredonda para baixo (i.e., não faz nada).
- Arredondamento para o mais próximo (*round to the nearest*): arredonda para o mais próximo de modo a ficar com o bit menos significativo da mantissa a zero (ver exemplos abaixo dos bits a arredondar):
  - **0XX** - arredonda para baixo.
  - **111** - arredonda para cima.
  - **110** - arredonda para cima.
  - **101** - arredonda para cima.
  - **100** - arredonda para cima se o bit menos significativo da mantissa for 1, caso contrário arredonda para baixo.

Exemplo 3: Aplicação do arredondamento para o mais próximo (*round to the nearest*) ao número em vírgula flutuante 1.0101100:

- arredondamento para uma mantissa de 4 bits: 1.0110 (arredonda para cima)
- arredondamento para uma mantissa de 3 bits: 1.011 (arredonda para cima)
- arredondamento para uma mantissa de 2 bits: 1.01 (arredonda para baixo)
- arredondamento para uma mantissa de 1 bit: 1.1 (arredonda para cima)

1. Indique a representação binária, em formato IEEE754 precisão simples, dos seguintes números reais.

- 0.25
- 0.046875
- -16.375
- -0.5625

2. Passe para a base decimal as sequências binárias, em formato IEEE754 precisão simples, dos seguintes números reais.

- 0 01111101 10010000000000000000000
- 0 10000010 11001000000000000000000
- 1 01111100 01011000000000000000000
- 1 10000011 10011100000000000000000

3. Qual o menor número normalizado positivo que pode representar com precisão simples? E não normalizado? E o maior número normalizado com precisão simples?

4. Calcule o resultado das seguintes operações usando o algoritmo de adição em vírgula flutuante IEEE754 com arredondamento para o mais próximo (*round to the nearest*).

- 32.5 + 0.046875
- -0.875 - 0.5625
- -325.875 + 0.546875
- 524288.25 - 0.0078125

5. Repita o exercício anterior assumindo que a mantissa tem apenas 3 bits (e não os 23 dos números representados em precisão simples).

6. Calcule o resultado da operação 1.00101 x 2^{-126} - 1.00010 x 2^{-126}. Há *overflow*? Há *underflow*? Que tipo de valor IEEE754 é obtido?

7. Verifique que o resultado das seguintes operações, usando o algoritmo de adição em vírgula flutuante IEEE754 e assumindo que o expoente tem apenas 3 bits (e não os 8 dos números representados em precisão simples), resultam em *overflow* ou *underflow*.

- 14 + 14
- 0.4375 - 0.28125