AC1 - Guião Prático 9 (Vírgula Flutuante)

$$N = (+/-)\ (1.f) \times 2^{Exp}$$

Exp → expoente

(1.f) → parte fracionária

mantissa (ou) significando

- O problema da divisão do espaço de armazenamento coloca-se também neste caso, mas agora na determinação do número de bits ocupados pela parte fracionária e pelo expoente.
- Esta divisão é um compromisso entre gama de representação e precisão:

  ⇒ Aumento do nº de bits da parte fracionária → maior precisão na representação

  ⇒ Aumento do nº de bits do expoente → maior gama de representação

Norma IEEE 754 (precisão simples)

31	30 ... 23	22 ... 0
S	Expoente (8 bits - E)	Parte fracionária (23 bits - f)

- S: Sinal da quantidade
- Expoente (8 bits - E): Expoente codificado em excesso de 127 ($2^{n-1} - 1$, com $n = 8$)
- Parte fracionária (23 bits - f): parte fracionária da mantissa

→ A representação é sempre normalizada: o bit à esquerda do ponto binário é sempre 1. Como é sempre 1, esse bit não é explicitamente representado (hidden bit)

- Exemplos:

① Qual o valor, em decimal, representado em 0X 41580000?

0 | 10000010 | 10110000000000000000000

Sinal positivo

Expoente = $2^7 + 2^1 = 128 + 2 = 130$

Expoente = 130 - offset = 130 - 127 = 3 = E

Mantissa = (1 + f) = 1 + .1011 = 1.1011

A quantidade representada será $+1.1011 \times 2^3$

$$R = +1.1011 \times 2^3 = \left(1 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 1\times2^{-4}\right) \times 2^3$$

$$= +1.6875 \times 8 = +13.5$$

② codificar no formato vírgula flutuante IEEE 754 precisão simples, o valor $-12593.75_{10} \times 10^{-3}$ =

$= -12.59375$

Parte inteira = $12_{10} = 1100_2$

Parte fracionária = $0.59375_{10} = 0.10011_2$

$12.59375_{10} = 1100.10011_2 \times 2^0$

Normalização: $1100.10011_2 \times 2^0 =$

$= 1.100100111_2 \times 2^3$

Expoente codificado: $+3 + 127 =$

$= 130_{10} = 10000010_2$

0.59375	x 2
①. 18750	
0.18750	x 2
⓪. 37500	x 2
⓪. 75000	x 2
①. 50000	
.50000	x 2
①. 00000	

1	10000010	10011000000000000000
C	1	4 9 8 0 0 0

0xC1498000

- Representação de quantidades em **precisão simples** (32 bits)

31	30 ... 23	22 ... 0
S	Expoente (8 bits) - ϵ	Parte fracionária (23 bits) - f

$N = (-1)^S \, 1.f \times 2^{(\epsilon - 127)}$

Precisão simples → tipo *float*

" "

- Representação de quantidades em **precisão dupla** (64 bits)

63	62 ... 52	51 ... 0
S	Expoente (11 bits) - ϵ	Parte fracionária (52 bits) - f

$N = (-1)^S \, 1.f \times 2^{(\epsilon - 1023)}$

Precisão dupla - tipo *double*

↳ A quantidade zero é representável no formato descrito.

- Técnicas de arredondamento do resultado

① Truncatura (exemplo com d = 2 bits)
↳ parte frac.

Val	Trunc(val)	Erro
x.00	x	0
x.01	x	$-1/4$
x.10	x	$-1/2$
x.11	x	$-3/4$

$$\text{Erro médio} = \frac{0 - 1/4 - 1/2 - 3/4}{4} = \frac{-1/4 - 2/4 - 3/4}{4} =$$

$$= \frac{\frac{-6}{4}}{4} = -\frac{6}{16} = -\frac{3}{8}$$

② Arredondamento simples (d = 2 bits)

val	Arred (val)	Erro
x.00	x	0
x.01	x	$x - x.25 = -1/4$
x.10	x + 1	$(x+1) - x.5 = +1/2$
x.11	x + 1	$(x+1) - x.75 = +1/4$

$$\text{Erro médio} = \frac{0 - 1/4 + 1/2 + 1/4}{4} = 1/8 \;//$$

→ Somar 1 ao 1º bit à direita do ponto binário e truncar o resultado

↳ $\boxed{\text{arred}(val) = \text{trun}(val + 0.5)}$

x.00	x.01	x.10	x.11
+0.1	+0.1	+0.1	+0.1
x.10	x.11	x+1.00	x+1.01

→ O erro médio é mais próximo de zero do que no caso da truncatura, mas ligeiramente polarizado do lado positivo

③ Arredondamento para o par mais próximo

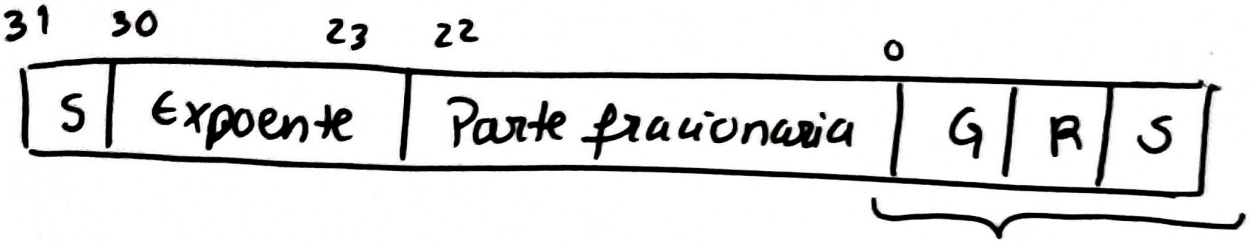
Val	Arred (val)	Erro	val	Arred (val)	Erro
x0.00	x 0	0	x1.00	x 1	0
x0.01	x 0	$-1/4$	x1.01	x 1	$-1/4$
x0.10	x 0	$-1/2$	x1.10	x+1	$+1/2$ ←
x0.11	x 1	$+1/4$	x1.11	x+1	$+1/4$

x0.10	x1.10
+ 0.0	+ 0.1
x0.10	x+1.00

$$\text{Erro médio} = \frac{\left(0 - 1/4 - 1/2 + 1/4\right) + \left(0 - 1/4 + 1/2 + 1/4\right)}{4}$$

$$= -1/8 + 1/8 = 0 \;//$$

Norma IEEE 754 - arredondamentos



G - Guard Bit

R - Round Bit

(S) - sticky Bit

resultado da soma lógica de todos os bits à direita do bit R (p.e. se houver à direita de R pelo menos um bit a '1', então S = '1').