Representação de Números em Ponto Fixo

`char`$_{|8|}$, `short`$_{|16|}$, `int`$_{|32|}$, `long`$_{|32|}$, `long long`$_{|64|}$

Números de 31 bits + sinal $\quad -2^{31} < n < +(2^{31} - 1)$

Números positivos de 32 bits $\quad 0 < n < +(2^{32} - 1)$

Representam 2^{32} quantidades distintas

Representação de inteiros com sinal em complemento de dois
é assimétrica: $[-2^{31}, 0] \ \cup \ [0, 2^{31})$

Representação de Números em Ponto Flutuante

`float`$_{|32|}$, `double`$_{|64|}$

- $1.0 \cdot 10^{-12}$ – picosegundos
- $3.15576 \cdot 10^{9}$ – segundos num ano
- aproximações para π e e

Representação posicional:

$$34.567_{10} = 3 \cdot 10 + 4 \cdot 1 + 5 \cdot 0.1 + 6 \cdot 0.01 + 7 \cdot 0.001$$

$$\begin{aligned} 101.1001_2 &= 1{\cdot}2^2 + 0{\cdot}2^1 + 1{\cdot}2^0 + 1{\cdot}2^{-1} + 0{\cdot}2^{-2} + 0{\cdot}2^{-3} + 1{\cdot}2^{-4} \\ &= 1{\cdot}4 + 0{\cdot}2 + 1{\cdot}1 + 1{\cdot}0.5 + 0{\cdot}0.25 + 1{\cdot}0.125 + 1{\cdot}0.0625 \end{aligned}$$

Representação em Ponto Flutuante – float

31	30 … 22	22 … 0
sinal	expoente	fração
	8 bits	23 bits

e bits de expoente, f bits de fração

$V = F \cdot \beta^{E}$ $\qquad$ para fração F, expoente E, e base β

menor número: $\approx 2.0 \cdot 10^{-38}$ $\qquad$ maior número: $\approx 2.0 \cdot 10^{+38}$

|expoente| $\leadsto$ faixa de representação
|fração| $\leadsto$ precisão na representação $\qquad 0 <$ fração < 1
faixa enorme representada por 2^{23} padrões $\neq$s $\qquad 2^{23} \ll 2^{32}$
$\longrightarrow$ precisão é menor que em ponto fixo $\qquad 2^{23}$ coisas $\neq$s

. Princípio 3: good design demands good compromise

Representação em Ponto Flutuante

$V = F \cdot 2^E$ para fração F, expoente E, e base 2

$V = (-1)^{sinal} \cdot (f_1 \cdot 2^{-1} + f_2 \cdot 2^{-2} + f_3 \cdot 2^{-3} + \dots) \cdot 2^E$

Número é normalizado se não há 0s a direita do ponto binário
normalização: desloca fração para esquerda (aumentando precisão)
enquanto decrementa expoente: $0.00101 \cdot 2^3 \overset{\text{norm}}{\leftrightarrow} 0.10100 \cdot 2^1$

Exemplo:
$-0.75_{10} = -3/4 = -3/2^2 =$
$11.0_2/2^2 = -0.11_2 = -0.11 * 2^0$

Representação em Ponto Flutuante

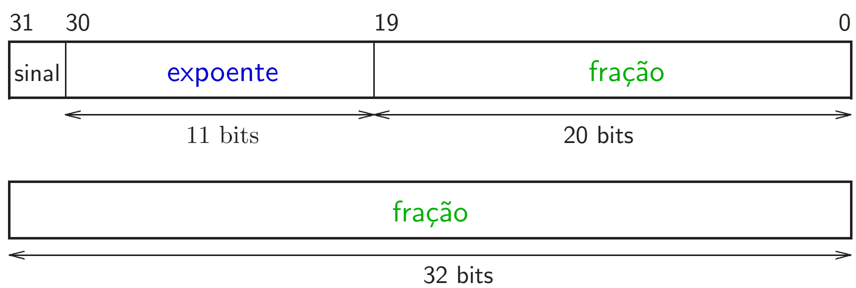
$V = F \cdot 2^E$ para fração F, expoente E, e base 2

Valor máximo da fração é $F_{\max} = 1 -$ ulp
ulp $= 2^{-f}$ *Unit in the Last Position*

Se R é resultado de operação aritmética e $R > F_{\max}$
então
mantissa deve ser reduzida: $F \cdot 2^E = (F/2) \cdot 2^{E+1}$ $/ =$ asr

Representação em Ponto Flutuante – double

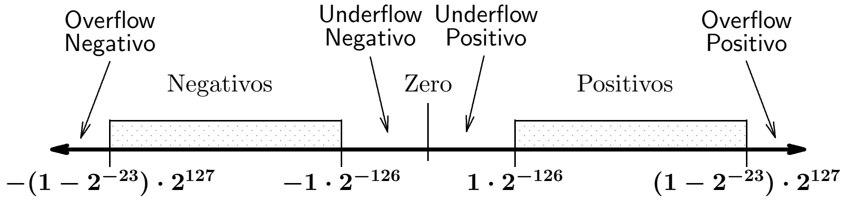
menor número: $\approx 2.0 \cdot 10^{-308}$ maior número: $\approx 2.0 \cdot 10^{+308}$



Formato:
$V = (-1)^{sinal} \cdot (f_1 \cdot 2^{-1} + f_2 \cdot 2^{-2} + f_3 \cdot 2^{-3} + \dots) * 2^E$

Faixa de Valores Representáveis

Faixa dos PF positivos: $F_{\min} \cdot 2^{E_{\min}} \leq V^{+} \leq F_{\max} \cdot 2^{E_{\max}}$
$|V^{+}| = |V^{-}|$



overflow: expoente muito grande para representação $> +127$

underflow: expoente muito pequeno para representação < -126

representação do zero?

Padrão IEEE 754

Padrão "universal" para representação em ponto flutuante

Primeiro dígito significativo da fração é implícito, à esq do ponto:
`s eeee eeee` **[1]**`.ffff ffff ffff ffff ffff fff`

	sinal	exp	mant
float	1	8	23+1
double	1	11	52+1

$\rightsquigarrow$ fração $\in [1, 2)$

números devem ser sempre normalizados!!!

Zero é caso especial: expoente e fração são todos zero
1`.ffff`$\cdots$`fff` = significando

Formato:
$(-1)^{s} \cdot (1 + \text{fração}) \cdot 2^{E}$
sinal · significando · expoente

Padrão IEEE 754 – expoente deslocado (i)

Qual a representação em float para os números 2^{4} e 2^{-4}?

`0 0000 0100 [1].0000` $\cdots$ `000` $\mapsto 2^{+4}$

`0 1111 1100 [1].0000` $\cdots$ `000` $\mapsto 2^{-4}$

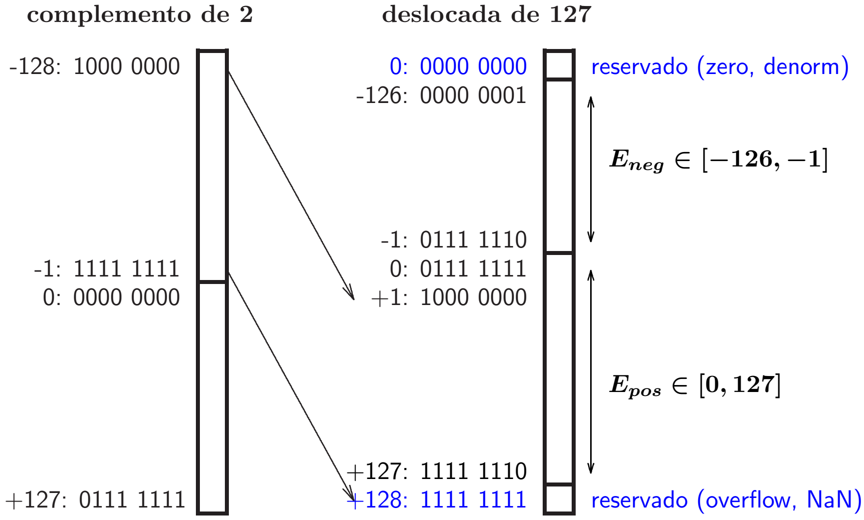
Considerando as duas representações como inteiros,
qual delas representa o maior número?

O expoente não é representado em complemento de dois
para que se possa comparar `floats` como se fossem inteiros `slt`

Formato:
$(-1)^{s} \cdot (1 + \text{fração}) \cdot 2^{(E-\text{deslocamento})}$
$(-1)^{s} \cdot (1 + f_1 \cdot 2^{-1} + f_2 \cdot 2^{-2} + f_3 \cdot 2^{-3} + \ldots) \cdot 2^{(E-\text{desloc})}$

onde deslocamento é 127 ou 1023

Padrão IEEE 754 – expoente deslocado (ii)



Padrão IEEE 754 – expoente deslocado (iii)

$(-1)^s \cdot (1 + \text{fração}) \cdot 2^{(E - \text{deslocamento})}$

Com expoente deslocado, número menor tem expoente menor

pode comparar floats e doubles com instruções para inteiros:

$\rightarrow$ `beq` e `slt`

Faixas de expoente e da fração permitem representar a recíproca de F^+_{min} sem overflow: $1/F^+_{min} < F^+_{max}$

Parâmetros do Formato IEEE 754	float	double
bits de precisão	24	53
Expoente máximo $E_{\max}$	127	1023
Expoente mínimo $E_{\min}$	-126	-1022
Deslocamento no exp.	127	1023

Padrão IEEE 754

Valores Especiais

Expoente	Fração	representa
$e = E_{\min} - 1$	$f = 0$	± 0
$e = E_{\min} - 1$	$f \neq 0$	$0.f \times 2^{E_{min}}$ ‡
$E_{\min} \leq e \leq E_{\max}$	—	$1.f \times 2^e$
$e = E_{\max} + 1$	$f = 0$	$\pm\infty$
$e = E_{\max} + 1$	$f \neq 0$	NaN

‡ formato denormalizado: $2^{-149} < F < 2^{-126}$

Operação com NaN resulta em NaN

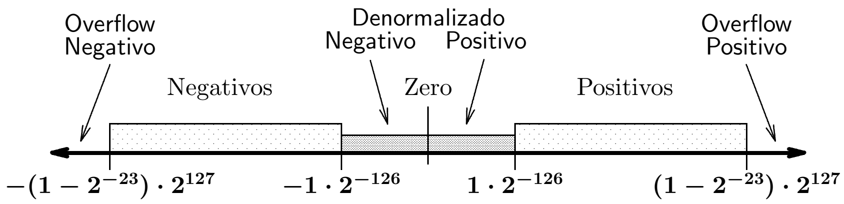
$5+$NaN $\rightarrow$ NaN

$0 \cdot \infty \rightarrow$ NaN

mas $1/0 \rightarrow \pm\infty$

Padrão IEEE 754 – núm denormalizados

Números com expoente menor que $E_{\min}$ são legais e possibilitam underflow gradual: x, y pequenos, se $x \neq y$ então $x - y \neq 0$



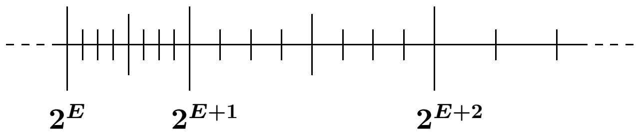
Padrão IEEE 754 – precisão

Seja x um número Real e $\mathcal{F}(x)$ sua representação em PF

O erro absoluto de representação é $\mathcal{F}(x) - x$

Sejam F_1 e F_2 tais que $F_1 \leq x \leq F_2$
então $\mathcal{F}(x)$ pode ser F_1 ou F_2.

Se $F_1 = M2^E$ então $F_2 = (M + ulp)2^E$
e o erro máximo é $1/2|F_1 - F_2| = ulp \cdot 2^E$



O erro relativo de representação é $\delta(x) = (\mathcal{F}(x) - x)/x$

Padrão IEEE 754 – exemplos

Exemplo 1:
$-0.75_{10} = -3/4 = -3/2^2 =$
$11.0_2/2^2 = -0.11_2 = -0.11 * 2^0 \overset{\text{norm}}{\leftrightarrow} -1.1 * 2^{-1}$

representado em float:
$(-1)^s \cdot (1 + \text{fração}) \cdot 2^{(\text{expoente}-127)}$
$(-1)^1 \cdot (1 + 0.1000\ldots0000) \cdot 2^{(126-127)}$

1	0111 1110	1000 0000 0000 0000 0000 000

representado em double:
$(-1)^s \cdot (1 + \text{fração}) \cdot 2^{(\text{expoente}-1023)}$
$(-1)^1 \cdot (1 + 0.1000\ldots0000) \cdot 2^{(1022-1023)}$

Padrão IEEE 754 – exemplos

$(-1)^s \cdot (1 + \text{fração}) \cdot 2^{(\text{expoente}-127)}$

Exemplo 2:

$0.5_{10} = 0.1_2 \overset{\text{norm}}{\leftrightarrow} 1.0 \cdot 2^{-1}$

$(-1)^0 \cdot (1 + 0.0000\ldots0000) \cdot 2^{(126-127)}$

0	0111 1110	0000 0000 0000 0000 0000 000

Exemplo 3:

$1.0_{10} = 1.0_2 \overset{\text{norm}}{\leftrightarrow} 1.0 \cdot 2^{0}$

$(-1)^0 \cdot (1 + 0.0000\ldots0000) \cdot 2^{(127-127)}$

0	0111 1111	0000 0000 0000 0000 0000 000

Padrão IEEE 754 – exemplos

$(-1)^s \cdot (1 + \text{fração}) \cdot 2^{(\text{expoente}-127)}$

Exemplo 4:

$2.0_{10} = 10.0_2 \overset{\text{norm}}{\leftrightarrow} 1.0 \cdot 2^{1}$

$(-1)^0 \cdot (1 + 0.0000\ldots0000) \cdot 2^{(128-127)}$

0	1000 0000	0000 0000 0000 0000 0000 000

Exemplo 5:

sinal=1, expoente=129, fração=0100. . .0000

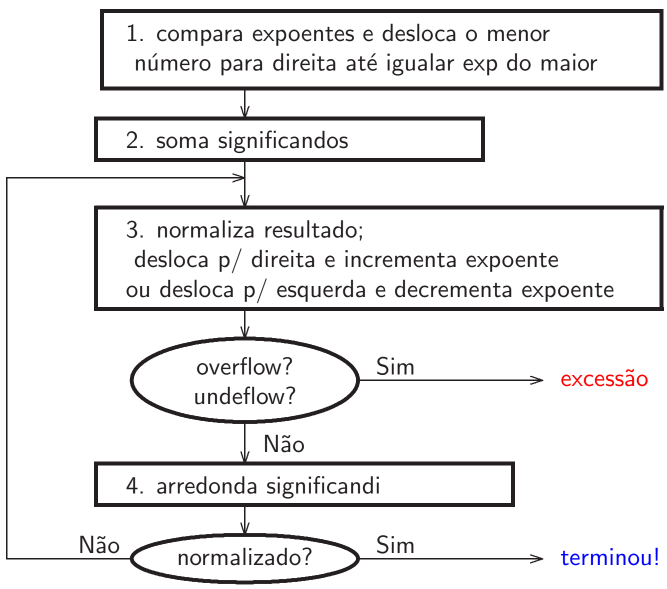
1	1000 0001	0100 0000 0000 0000 0000 000

$(-1)^s \cdot (1 + \text{fração}) \cdot 2^{(\text{expoente}-127)}$

$(-1)^1 \cdot (1 + 0.0100\ldots0000) \cdot 2^{(129-127)}$

$-1 \cdot (1 + 0.25) \cdot 2^2 = -5.0_{10}$

Adição em Ponto Flututante



Adição em Ponto Flututante II

Exemplo: $\mathbf{9.999 * 10^{1} + 1.610 * 10^{-1}}$
base-10, significando com 4 dígitos (1.3) mais 2 dígitos no expoente

1. compara; desloca significando e ajusta expoente do menor
$\mathbf{0.01610 * 10^{1}}$ trunca para quatro dígitos: $\mathbf{0.016 * 10^{1}}$

2. soma

$$\begin{array}{r} \mathbf{9.999} \\ \mathbf{+0.016} \\ \hline \mathbf{10.015} \end{array}$$

3. normaliza
$\mathbf{10.015 \cdot 10^{0} \overset{\text{norm}}{\leftrightarrow} 1.0015 \cdot 10^{1}}$

.

4. arredonda e trunca para 4 dígitos
se dígito à direita $\mathbf{0 \leq d \leq 4}$, arredonda para menos;
senão $(\mathbf{5 \leq d \leq 9})$, arredonda para mais erredonda?

$\mathbf{9.999 * 10^{1} + 1.610 * 10^{-1} = 1.002 * 10^{1}}$

Adição em Ponto Flututante III

Exemplo: $\mathbf{0.5 - 0.4375}$
base-10, significando com 4 dígitos (1.3) mais 2 dígitos no expoente

$\mathbf{0.5_{10} = 0.1_{2} \overset{\text{norm}}{\leftrightarrow} 1.000 \cdot 2^{-1}}$

$\mathbf{-0.4375_{10} = -7/2^{4} = -111_{2}/2^{4} = -0.0111_{2} \overset{\text{norm}}{\leftrightarrow} -1.110 \cdot 2^{-2}}$

1. compara; desloca significando e ajusta expoente do menor; trunca
$\mathbf{-1.110 \cdot 2^{-2} \rightarrow -0.111 \cdot 2^{-1}}$

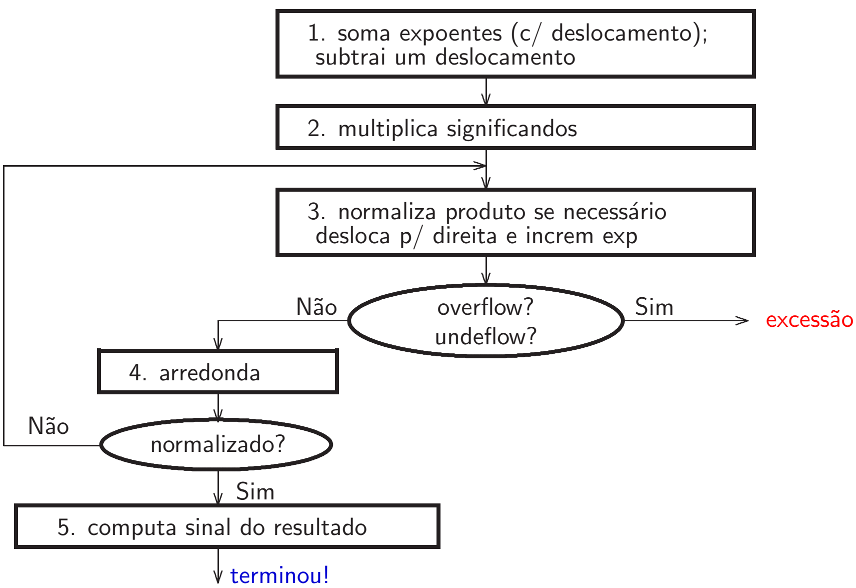
2. soma

$$\begin{array}{r} \mathbf{1.000 \cdot 2^{-1}} \\ \mathbf{-0.111 \cdot 2^{-1}} \\ \hline \mathbf{0.001 \cdot 2^{-1}} \end{array}$$

3. normaliza
$\mathbf{0.001 \cdot 2^{-1} \overset{\text{norm}}{\leftrightarrow} 1.000 \cdot 2^{-4}}$

4. arredonda e trunca para 4 dígitos
$\mathbf{0.5 - 0.4375 = 1.000 \cdot 2^{-4}}$

Multiplicação em Ponto Flututante



Multiplicação em Ponto Flututante II

Exemplo: $\mathbf{1.110 \cdot 10^{10} \times 9.200 \cdot 10^{-5}}$ b-10, $|M| = \mathbf{1.3}$, $|E| = \mathbf{2}$

1. soma expoentes – com deslocamento!!

$$\begin{array}{r} \mathbf{10 + 127 = 137} \\ \mathbf{+(-5 + 127) = 122} \\ \hline \mathbf{259} \end{array}$$

$$\mathbf{259 - 127 = 132 = 127 + 5}$$

2. multiplica significandos

$$\begin{array}{r} \mathbf{1.110} \\ \mathbf{\times 9.200} \\ \hline \mathbf{0000} \\ \mathbf{0000} \\ \mathbf{2220} \\ \mathbf{9990} \\ \hline \mathbf{10.212000} \end{array}$$

$\mathbf{10.212000 \rightarrow 10.212 \cdot 10^5}$

3. normaliza:
$\mathbf{10.212 \cdot 10^5 \overset{norm}{\leftrightarrow} 1.0212 \cdot 10^6}$

4. arredonda e trunca para 4 dígitos:
$\mathbf{1.0212 \cdot 10^6 = 1.021 \cdot 10^6}$

5. calcula sinal: $+ \times + = +$
$\mathbf{+1.021 \cdot 10^6}$

Multiplicação em Ponto Flututante III

Exemplo: $\mathbf{0.5 \times -0.4375}$ b-10, $|M| = \mathbf{1.3}$, $|E| = \mathbf{2}$
$\mathbf{1.000 \cdot 2^{-1} \times -1.110 \cdot 2^{-2}}$

1. soma expoentes – com deslocamento!!
$\mathbf{(-1 + 127) + (-2 + 127) - 127 = 124}$ $\mathbf{124 = 127 - 3}$

2. multiplica significandos

$$\begin{array}{r} \mathbf{1.000} \\ \mathbf{\times 1.110} \\ \hline \mathbf{0000} \\ \mathbf{1000} \\ \mathbf{1000} \\ \mathbf{1000} \\ \hline \mathbf{1110000} \end{array}$$

$\mathbf{111000 \rightarrow 1.110 \cdot 2^{-3}}$

3. normaliza:
$\mathbf{1.110 \cdot 2^{-3}}$

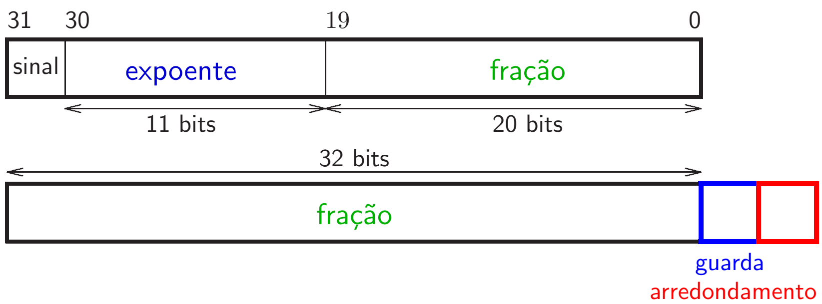
4. arredonda e trunca para 4 dígitos:
$\mathbf{1.110 \cdot 2^{-3}}$

5. calcula sinal: $- \times + = -$
$\mathbf{-1.110 \cdot 2^{-3} = -0.00111}$
$\mathbf{= -7/32 = -0.21875}$

Exatidão

Num double só $\mathbf{2^{53}}$ números em $[\mathbf{0, 1}]$ são representados exatamente

IEEE 754 prescreve uso de 2 bits adicionais na implementação:
bit de guarda e bit de arredondamento que garantem
precisão melhor que metade do bit menos significativo da fração



Sem bits de guarda e arredondamento, qual é a perda de precisão a cada operação?