Representação de Dados Ponto Flutuante

Noemi Rodriguez
Ana Lúcia de Moura

http://www.inf.puc-rio.br/~inf1018

Representação em Ponto Flutuante

Tipos C **float** e **double**

Codifica números racionais na forma $\mathbf{x * 2^y}$

- adequado para valores grandes ou muito pequenos

Padrão IEEE-754 (~1985)

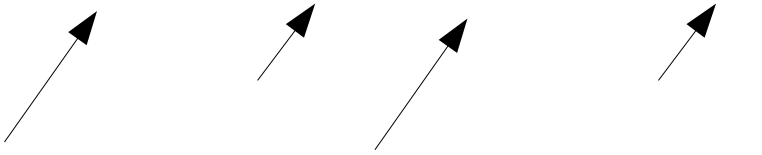
- representação de número de ponto flutuante e suas operações
- portabilidade

Ponto Flutuante

Base 10

- números racionais p/q → mantissa X 10 ^{exp}
- ponto (vírgula) "flutua", compensado pelo expoente

$$129 = 12.9 \times 10^{1} = 1.29 \times 10^{2}$$



aproximação → representação de **alguns** números

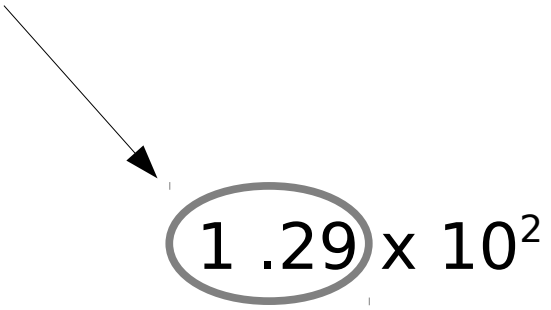
Normalização

Múltiplas representações possíveis

$129 \times 10^0 = 12.9 \times 10^1 = 1.29 \times 10^2 = 0.129 \times 10^3 = 1290 \times 10^{-1}$

Escolha de uma representação padrão (**normalizada**)

Base 10: $1 <= \text{mantissa} < 10$ ← base



Números Fracionários

Base 10

$$12.34_{10} = 1 * 10^{1} + 2 * 10^{0} + 3 * 10^{-1} + 4 * 10^{-2}$$

potências POSITIVAS de 10 — potências NEGATIVAS de 10

Base 2

$$101.11_{2} = 1 * 2^{2} + 0 * 2^{1} + 1 * 2^{0} + 1 * 2^{-1} + 1 * 2^{-1}$$

potências POSITIVAS de 2 — potências NEGATIVAS de 2

$$= 4 + 1 + ½ + ¼ = 5.75$$

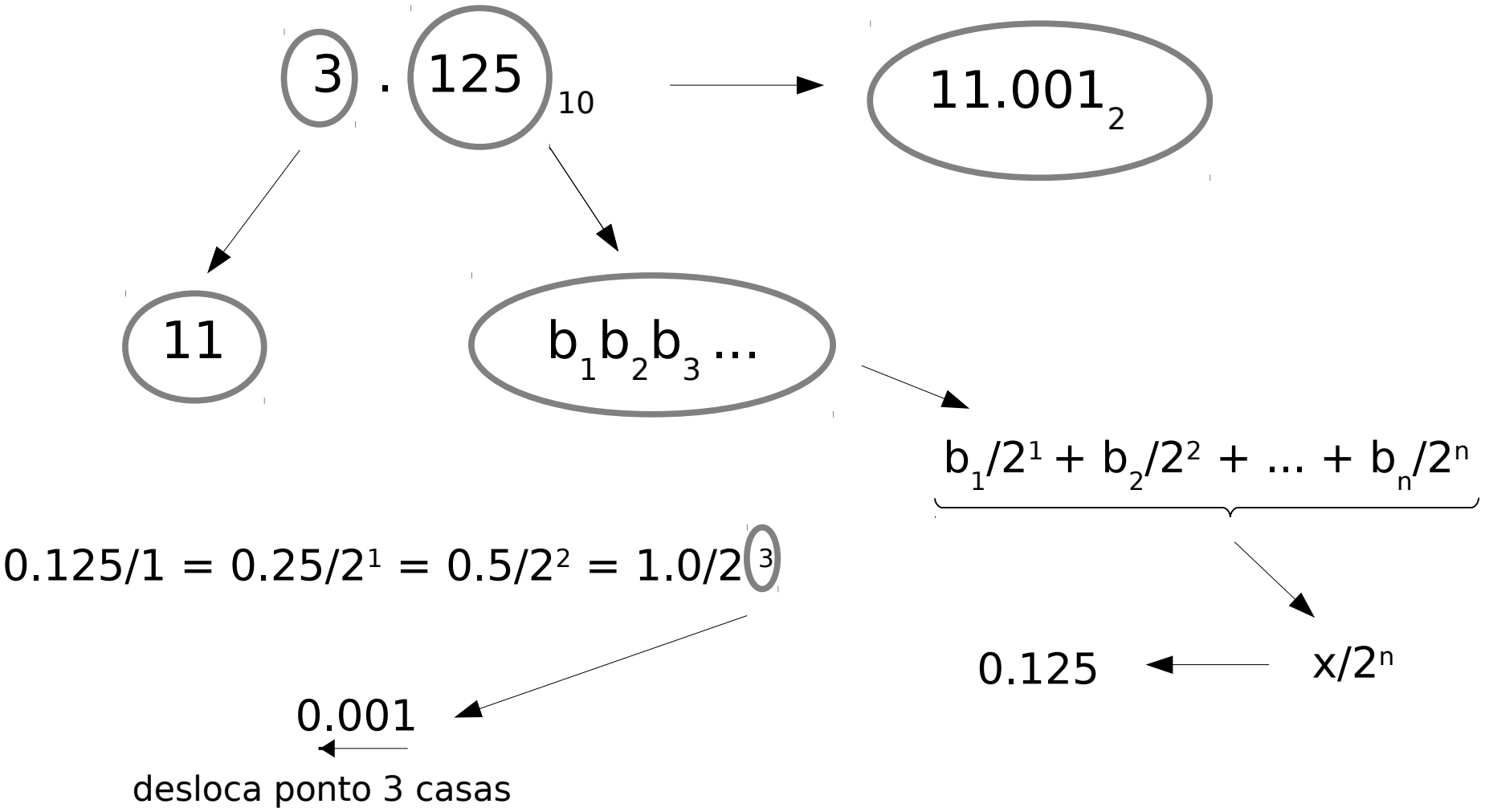
Conversão binário → decimal

$1.001 = 1 * 2^{0} + 1 * 2^{-3} = 1 + 1/8 = 1.125$

$101.111 = 1 * 2^{2} + 1 * 2^{0} + 1 * 2^{-1} + 1 * 2^{-2} + 1 * 2^{-3} =$
$5 + 1/2 + 1/4 + 1/8 = 5.875$

$11.0011 = 1 * 2^{1} + 1 * 2^{0} + 1 * 2^{-3} + 1 * 2^{-4} =$
$3 + 1/8 + 1/16 = 3.1875$

Conversão decimal → binário



Algoritmo de Conversão

$0.625 = 0.b_1b_2b_3... b_n$

$\downarrow * 2$

$1.25 = b1 .b_2b_3... b_n \longrightarrow$ 0.1

$0.25 = 0.b_2b_3... b_n$

$\downarrow * 2$

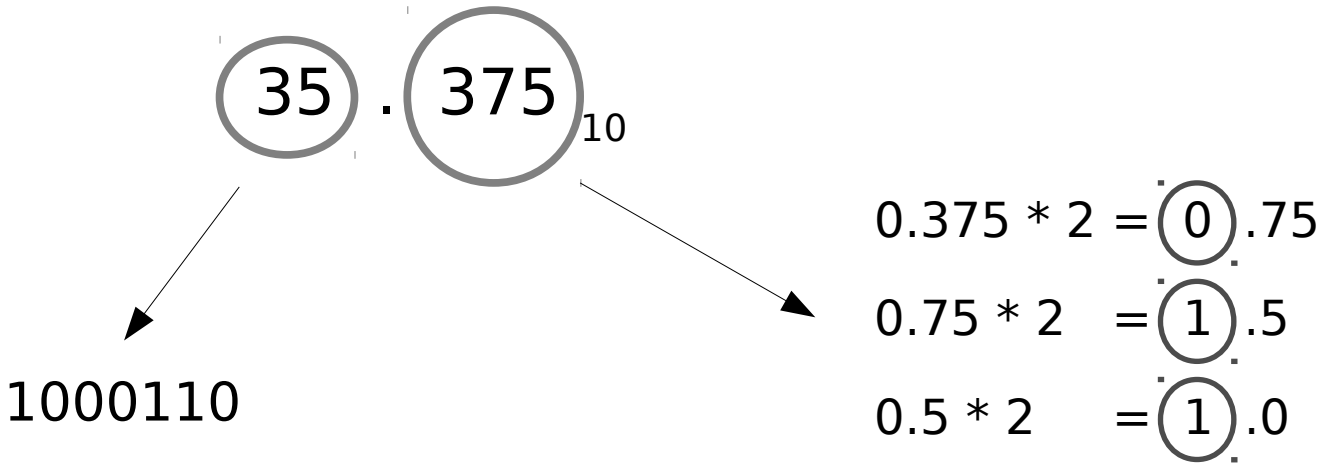
$0.5 = b2 .b_3... b_n \longrightarrow$ 0.10

$0.5 = 0.b_3... b_n$

$\downarrow * 2$

$1.0 = b3 .b_4... b_n \longrightarrow$ 0.101

Conversão decimal → binário



1000110.011_2

Limitações da Representação

A notação decimal representa números que podem ser escritos como $\sum_{k=-n}^{m} d_k \cdot 10_k$

não há representação precisa para 1/3 ou 5/7

A notação binária representa números que podem ser escritos como

$$\sum_{k=-n}^{m} b_k \cdot 2^k$$

Outros números devem ser **aproximados**

parte fracionária possui sequências repetidas indefinidamente

Exemplo de Sequência Infinita

0.3 * 2 = 0.6

0.6 * 2 = 1.2

0.2 * 2 = 0.4

0.4 * 2 = 0.8

0.8 * 2 = 1.6

0.6 * 2 = 1.2

...

0.0 1001 1001 1001 ....

mais bits, maior precisão

Representação IEEE 754

Representa números na forma $\mathbf{x * 2^y}$

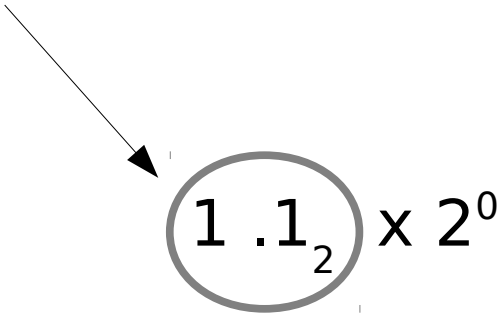
com um par adequado (x,y)

Existem várias representações possíveis

$1.5_{10} = 1.1_2 \times 2^0 = 11.0_2 \times 2^{-1} = 0.11_2 \times 2^1$

Normalização

Base 2: $1 <= \text{mantissa} < 2$ ← base



Codificação: Forma Numérica

$$(-1)^s \; M \;\; 2^E$$

Bit de sinal **s** determina se número é negativo ou positivo

Mantissa **M** é um valor **fracionário** 1 <= M < 2

Expoente **E** indica potência positiva ou negativa de 2

Codificação

$$(-1)^s \; M \;\; 2^E$$

Bit **s** indica sinal (1 : n° negativo, 0 : n° positivo)

Campo **exp** **codifica** **E**

Campo **frac** **codifica** **M**

Padrão IEEE: Precisão

`float` (32 bits): **`exp`** = 8 bits, **`frac`** = 23 bits, **`s`** = 1 bit

faixa de valores: 2^{-126} até 2^{127}

`double` (64 bits): **`exp`** =11 bits, **`frac`** = 52 bits, **`s`** = 1 bit

faixa de valores: 2^{-1022} até 2^{1023}

Representação de Valores

$(-1)^s \; M \; 2^E$

frac = M - 1 (1 <= **M** < 2) ← normalização

1 é implícito → $M = 1 . \underbrace{b_1 b_2 b_3 \ldots}_{\textbf{frac}}$

exp = E + bias ← representação em excesso 2^n

exp é um valor sem sinal e **bias** = $\mathbf{2^{n-1} - 1}$

float (precisão simples) → **bias = 127** ($2^{8-1} - 1 = 2^7 - 1 = 127$)

double (precisão dupla) → **bias = 1023** ($2^{11-1} - 1 = 2^{10} - 1 = 1023$)

Exemplo 1: Precisão Simples

Como representar um número **X** em float?

float f = 15213.0;

$15213_{10} = 11101101101101_2 = 1.1101101101101_2 * 2^{13}$

13 casas para a esquerda

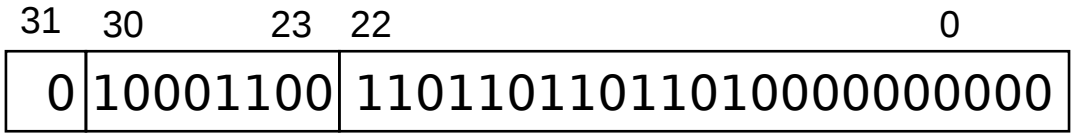
M

E

frac (23 bits) = $11011011011010000000000_2$

exp (8 bits) = E + bias = 13 + 127 = 140 = 1001100_2

s = 0



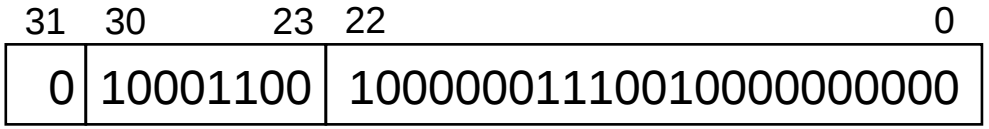
Binário	**0100**	**0110**	**0110**	**1101**	**1011**	**0100**	**0000**	**0000**
Hex:	**4**	**6**	**6**	**D**	**B**	**4**	**0**	**0**

DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Exemplo 2: Precisão Simples

Como encontrar o valor de um número **X** representado em IEEE 754?

Hex:	**4**	**6**	**4**	**0**	**E**	**4**	**0**	**0**
Binário	**0100**	**0110**	**0100**	**0000**	**1110**	**0100**	**0000**	**0000**



frac = 1000000111001_2 → **M** = 1.1000000111001_2

exp = 10001100_2 → 140 → **E** = 140 - 127 = 13

s = 0

$$V = (-1)^0 * (1 + 4153/8192) * 2^{13} = 12345_{10}$$

Exemplo 3: Precisão Dupla

double d = 178.125;

$178.125_{10} = 10110010.001_2 = 1.0110010001_2 * 2^7$

7 casas para a esquerda

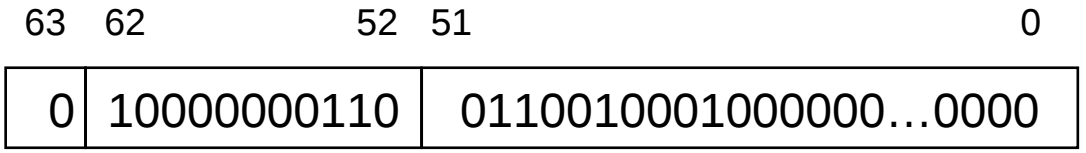
M

E

frac (52 bits) = $\underline{0110010001}000...000_2$

exp (11 bits) = E + bias = 7 + 1023 = 1030 = 10000000110_2

s = 0



Binário	**0100**	**0000**	**0110**	**0110**	**0100**	**0100**	**0000**	**0000**	**…**	**0000**		
Hex:	**4**	**0**	**6**	**6**	**4**	**4**	**0**	**0**	**00**	**00**	**00**	**00**

DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Valores Especiais

zero	S= 0	exp = 0	M = 0
+ ∞	S=0	exp=111...111	M = 0
- ∞	S=1	exp=111...111	M = 0
NaN	S	exp=111...111	M ≠ 0

infinito: permite representação de *overflow*

- multiplicação, divisão por 0

NaN (*not a number*)

- operações que não resultam em um número real (√-1 , ∞ - ∞)

em math.h: NAN, INFINITY

Valores Denormalizados

Representação de 0 e números muito próximos de 0

M = frac → não há "1" implícito (0 <= M < 1)

E = 1 - bias → -126 ou -1022

$$(-1)^S * 0.b_1b_2b_3\ldots. * 2^{-126}$$
$$(-1)^S * 0.b_1b_2b_3\ldots. * 2^{-1022}$$