Centro de Informática UFPE

Universidade Federal de Pernambuco

Universidade Federal de Pernambuco
Centro de Informática

# Cálculo Numérico (IF215)

**Profª. Maíra Santana**

AULA 3: Aritmética de ponto flutuante

# Objetivos

- Compreender como os números são representados no computador;
- Tipos e análise de erros numéricos.

# Representação de números inteiros

- Nos computadores, os números são armazenados utilizando uma **base binária**;

- Usualmente utilizamos a **base decimal**:

$$245 = 2 \cdot 10^2 + 4 \cdot 10^1 + 5 \cdot 10^0$$

- Um número com **n** dígitos na **base β** pode ser representado na forma:

$$(d_{n-1}d_{n-2}\cdots d_1 d_0)_\beta$$
$$= d_{n-1} \cdot \beta^{n-1} + d_{n-2} \cdot \beta^{n-2} + \cdots + d_1 \cdot \beta^1 + d_0 \cdot \beta^0$$

onde $0 \leq d_j < \beta$

- Ou seja, os dígitos são representados como potências da base.

# Representação de números inteiros

- Exemplos:
  - $(1101)_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$

# Representação de números inteiros

- Exemplos:
  - $(1101)_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = \mathbf{13}$

# Representação de números inteiros

- Exemplos:
  - $(1101)_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = \mathbf{13}$
  - $(312)_8 = 3 \cdot 8^2 + 1 \cdot 8^1 + 2 \cdot 8^0$

# Representação de números inteiros

- Exemplos:
  - $(1101)_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = \mathbf{13}$
  - $(312)_8 = 3 \cdot 8^2 + 1 \cdot 8^1 + 2 \cdot 8^0 = \mathbf{202}$

# Representação de números inteiros

- Exemplos:
  - $(1101)_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = \mathbf{13}$
  - $(312)_8 = 3 \cdot 8^2 + 1 \cdot 8^1 + 2 \cdot 8^0 = \mathbf{202}$
  - $(12E)_{16} =$

na **base hexadecimal**:

| | |
|---|---|
| A | 10 |
| B | 11 |
| C | 12 |
| D | 13 |
| E | 14 |
| F | 15 |

# Representação de números inteiros

- Exemplos:
  - $(1101)_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = \mathbf{13}$
  - $(312)_8 = 3 \cdot 8^2 + 1 \cdot 8^1 + 2 \cdot 8^0 = \mathbf{202}$
  - $(12E)_{16} = 1 \cdot 16^2 + 2 \cdot 16^1 + 14 \cdot 16^0 = \mathbf{302}$

na **base hexadecimal**:

| | |
|---|---|
| A | 10 |
| B | 11 |
| C | 12 |
| D | 13 |
| E | 14 |
| F | 15 |

# Representação de números inteiros

## Conversão de **decimal** para **binário**

- Considere o número $347$ e seja $\left(d_j d_{j-1} \cdots d_1 d_0\right)_2$ a representação dele na base 2.

$$(347)_{10} = d_j \cdot 2^j + d_{j-1} \cdot 2^{j-1} + \cdots + d_1 \cdot 2^1 + d_0 \cdot 2^0$$

# Representação de números inteiros

## Conversão de **decimal** para **binário**

- Considere o número 347 e seja $\left(d_j d_{j-1} \cdots d_1 d_0\right)_2$ a representação dele na base 2.

$$(347)_{10} = d_j \cdot 2^j + d_{j-1} \cdot 2^{j-1} + \cdots + d_1 \cdot 2^1 + d_0 \cdot 2^0$$

Colocando o 2 em evidência, temos:

$$(347)_{10} = 2\,(d_j \cdot 2^{j-1} + d_{j-1} \cdot 2^{j-2} + \cdots + d_1) + d_0$$

# Representação de números inteiros

## Conversão de **decimal** para **binário**

- Considere o número 347 e seja $(d_j d_{j-1} \cdots d_1 d_0)_2$ a representação dele na base 2.

$$(347)_{10} = d_j \cdot 2^j + d_{j-1} \cdot 2^{j-1} + \cdots + d_1 \cdot 2^1 + d_0 \cdot 2^0$$

Colocando o 2 em evidência, temos:

$$(347)_{10} = 2\,(d_j \cdot 2^{j-1} + d_{j-1} \cdot 2^{j-2} + \cdots + d_1) + d_0$$

Como 347 é um número **ímpar** e na base binária apenas os dígitos o (zero) e 1 (um) são possíveis, então, necessariamente $d_0 = 1$.

# Representação de números inteiros

## Conversão de **decimal** para **binário**

$$(347)_{10} = 2\,(d_j \cdot 2^{j-1} + d_{j-1} \cdot 2^{j-2} + \cdots + d_1) + d_0$$

Como 347 é um número **ímpar** e na base binária apenas os dígitos o (zero) e 1 (um) são possíveis, então, necessariamente $d_0 = 1$.

- $d_0$ também pode ser interpretado como o **resto da divisão inteira** de 347 por 2;
- A **divisão inteira** de 347 por 2 é 173, portanto:

$$\frac{347}{2} = 173 = d_j \cdot 2^{j-1} + d_{j-1} \cdot 2^{j-2} + \cdots + d_1$$

- Colocando 2 em evidência novamente podemos, agora, encontrar o valor de $d_1$.

# Representação de números inteiros

## Conversão de **decimal** para **binário**

- Esse procedimento pode ser repetido para encontrar todos os dígitos $d_j$. Então:

| n | n // 2 | n % 2 |
|---|---|---|
| 347 | 173 | 1 |
| 173 | 86 | 1 |
| 86 | 43 | 0 |
| 43 | 21 | 1 |
| 21 | 10 | 1 |
| 10 | 5 | 0 |
| 5 | 2 | 1 |
| 2 | 1 | 0 |
| 1 | 0 | 1 |

Logo, a representação binária de 347 é: $(101011011)_2$

# Representação de números com parte fracionária

- Exemplos:
  - $(0.234)_{10} = 2 \cdot 10^{-1} + 3 \cdot 10^{-2} + 4 \cdot 10^{-3}$
  - $(0.1011)_2 = 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} = \mathbf{0.6875}$
  - $(0.17)_8 = 1 \cdot 8^{-1} + 7 \cdot 8^{-2} = \mathbf{0.234375}$

# Representação de números com parte fracionária

## Conversão de **decimal** para **binário**

- O número 0.625 possui uma representação binária $(0.d_1d_2d_3\cdots)_2$.

$$0.625 = d_1 \cdot 2^{-1} + d_2 \cdot 2^{-2} + d_3 \cdot 2^{-3} + \cdots$$

Agora, se **multiplicarmos** os dois lados da equação por 2, temos:

$$1.25 = d_1 + d_2 \cdot 2^{-1} + d_3 \cdot 2^{-2} + \cdots$$

Então, $d_1$ é a **parte inteira** de 1.25. Portanto, $d_1 = 1$.

# Representação de números inteiros

## Conversão de **decimal** para **binário**

- Esse procedimento pode ser repetido para encontrar todos os dígitos $d_j$, mas agora realizando uma sequência de multiplicações:

| n | 2 · n | Parte inteira |
|---|---|---|
| 0.625 | 1.25 | 1 |
| 0.25 | 0.5 | 0 |
| 0.5 | 1.0 | 1 |

Logo, a representação binária de 0.625 é:
$(0.101)_2$

# Aritmética de ponto flutuante

- Os números com parte fracionária são representados no computador com o ponto flutuante

$$\pm(d_0.d_1d_2d_3\cdots d_t)\cdot\beta^e$$

  - $\beta$ é a base;
  - $t$ é o número de bits da mantissa;
  - $e$ é o expoente (*offset* binário).

- Sistema de ponto flutuante:

$$F(\beta, t, e_{min}, e_{max})$$

- Menor número de máquina positivo: $x_{min} = 1{,}00\ldots0 \ * \beta^{e_{min}}$
- Maior número de máquina positivo: $x_{max} = (\beta-1),(\beta-1)\ldots(\beta-1) * \beta^{e_{max}}$

# Aritmética de ponto flutuante

- Padrão IEEE:
  - Base binária;
  - Sinal ocupa 1 bit;
  - Mantissa de até 52 bits;
  - Expoente de até 11 bits.

- Python:

```
import sys
print(sys.float_info)
```

# Tipos de erros

- Arredondamento:
  - Regra geral: número de máquina mais próximo;
  - Precisão dos números de ponto flutuante.
- Truncamento (discretização):
  - Ex.: $\lim_{n \to 0}(\cdots) \longrightarrow \lim_{n \to \approx 0}(\cdots)$
- Modelagem (inerente):
  - Ao desconsiderar algumas variáveis do problema.
- Entrada (inerente):
  - Possíveis erros de medição.

# Representação de erros

- Erro absoluto:
  - É a diferença entre o valor exato ($x$) e o valor aproximado ($\bar{x}$):

$$\Delta\bar{x} = x - \bar{x}$$

- Erro relativo:
  - É o quociente entre o erro absoluto e o valor exato:

$$\delta\bar{x} = \frac{\Delta\bar{x}}{x} = \frac{(x - \bar{x})}{x}$$

- Percentual:
  - É o produto do erro relativo por 100:

$$\mathrm{p}\bar{x} = 100 \cdot \delta\bar{x} \ \%$$

# Exemplos de erros numéricos

https://colab.research.google.com/drive/1NJsh32cDkbYsfoNThhDMFBW535Cfh6wh?usp=sharing

# Representação binária do número 0.1

$$0.1 = d_1 \cdot 2^{-1} + \cdots$$

| n | 2 · n | Parte inteira | |
|---|---|---|---|
| 0.1 | 0.2 | 0 | d1 = 0 |
| 0.2 | 0.4 | 0 | d2 = 0 |
| 0.4 | 0.8 | 0 | d3 = 0 |
| 0.8 | 1.6 | 1 | d4 = 1 |
| 0.6 | 1.2 | 1 | d5 = 1 |
| 0.2 | 0.4 | 0 | d6 = 0 |
| 0.4 | 0.8 | 0 | d7 = 0 |
| 0.8 | 1.6 | 1 | d8 = 1 |
| 0.6 | 1.2 | 1 | d9 = 1 |
| 0.2 | 0.4 | 0 | d10 = 0 |
| … | … | … | … |

# Justificativa

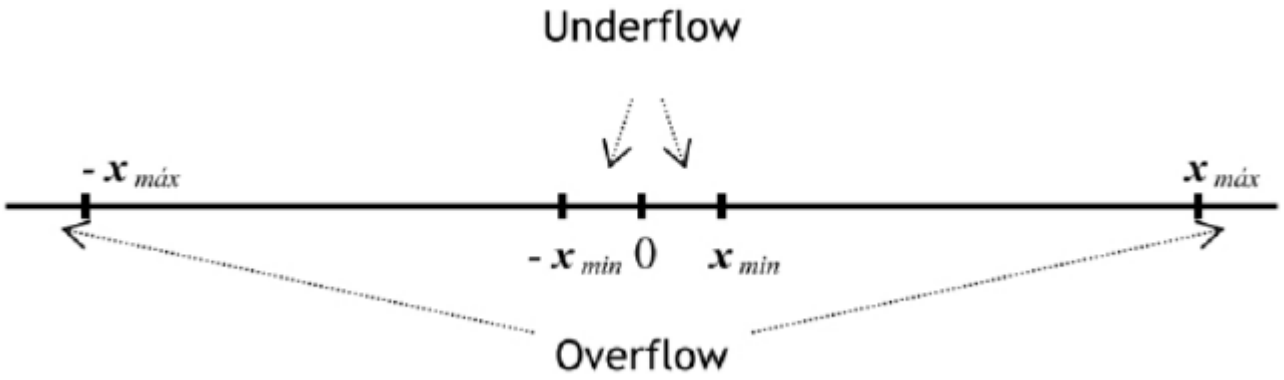
- Em base 2, 1/10 e uma fração binária que se repete infinitamente (dízima periódica);
- Não importa quantos dígitos você está disposto a usar, 0.1 não pode ser representado exatamente como uma fração de base 2;
- Se limitarmos a representação a qualquer número finito de bits obtemos apenas uma **aproximação**.

# Recomendações

- Conhecer o sistema de representação que está sendo utilizado para realizar operações numéricas;
- Cuidado com operações que modificam a ordem de magnitude:
  - Divisão por valores próximos a zero;
  - Subtração de valores próximos.
- Cuidado com comparação direta de números com expressões:
  - Uso de operadores lógicos.

# Atenção aos arredondamentos!

- Overflow: quando o resultado da operação ( $\bar{x}$ ) pertence a $(-\infty, -x_{máx}) \cup (x_{máx}, +\infty)$;
- Underflow: quando o resultado da operação ( $\bar{x}$ ) pertence a $(-x_{min}, 0) \cup (0, x_{min})$.



# Referências


- Métodos Numéricos. José Dias dos Santos e Zanoni Carvalho da Silva. **(capítulo 1);**
- Cálculo Numérico – aspectos teóricos e computacionais. Márcia A. Gomes Ruggiero e Vera Lúcia da Rocha Lopes. **(capítulo 1)**.