Cálculo Numérico Computacional

Aula 02

Representação de Números Sistema Decimal e Binário

Prof. Lucas Zanovello lucas.tahara@unesp.br

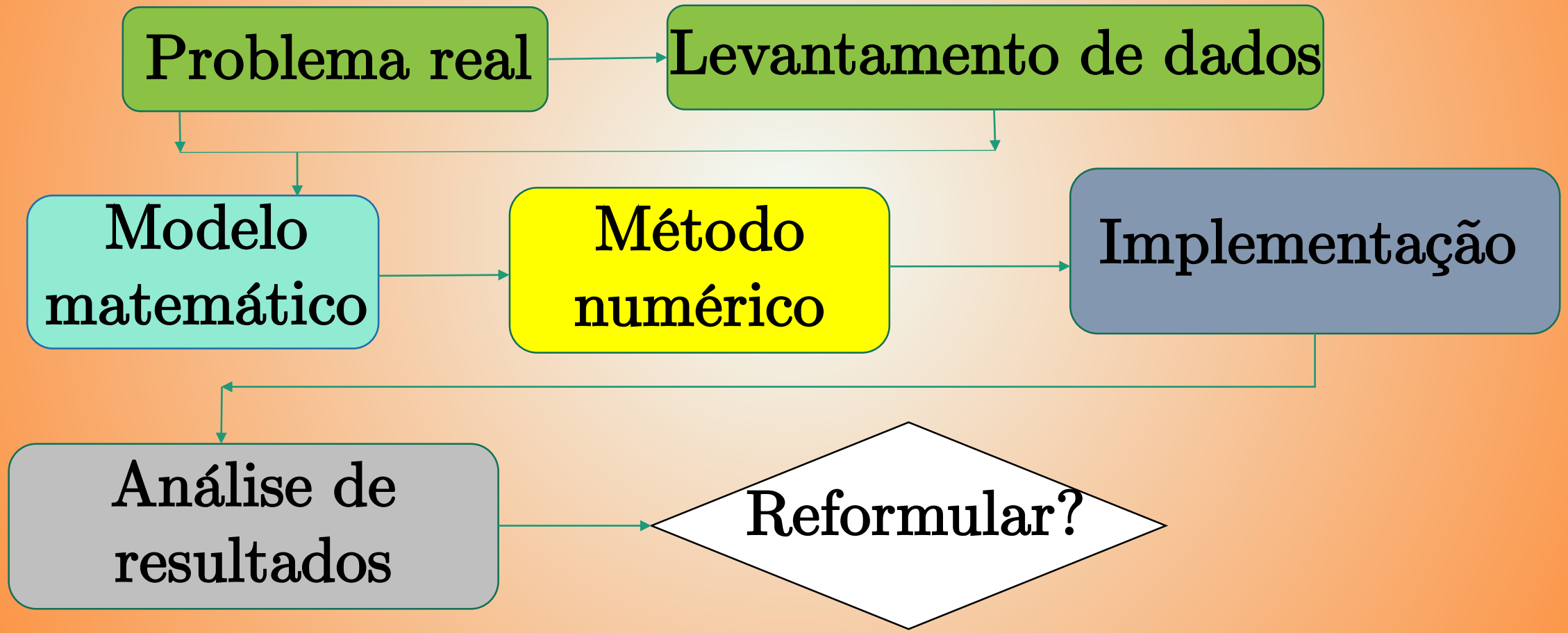
Cálculo Numérico Computacional

Agenda:

1. Introdução;
2. Representação de números;
3. Conversão de números – decimal/binário;
4. Aritmética de ponto flutuante;
5. Encerramento.

Cálculo Numérico Computacional

1. Introdução



Cálculo Numérico Computacional

Os resultados dependem:

- Da precisão de dados de entrada;
- Da representação dos dados no computador;
- Das operações numéricas efetuadas.

Referências:

- MATHEWS, J. H. *Numerical Methods for Mathematics, Science and Engineering*, second edition. Prentice Hall International, INC., 1992.
- OVERTON, M. L. & PAIGE, C. *Notes on Numerical Computing*. Computer Science Department of New York University and McGill University, USA, 1995.
- WILKINSON, J. H. *Rounding Errors in Algebraic Processes*. Prentice Hall, Inc., 1963.

Cálculo Numérico Computacional

2. Representação de números

Exemplo: cálculo de uma área circular de raio 100 m.

a) 31400 m²
b) 31416 m²
c) 31415.92654 m²

Qual está correta?

Cálculo Numérico Computacional

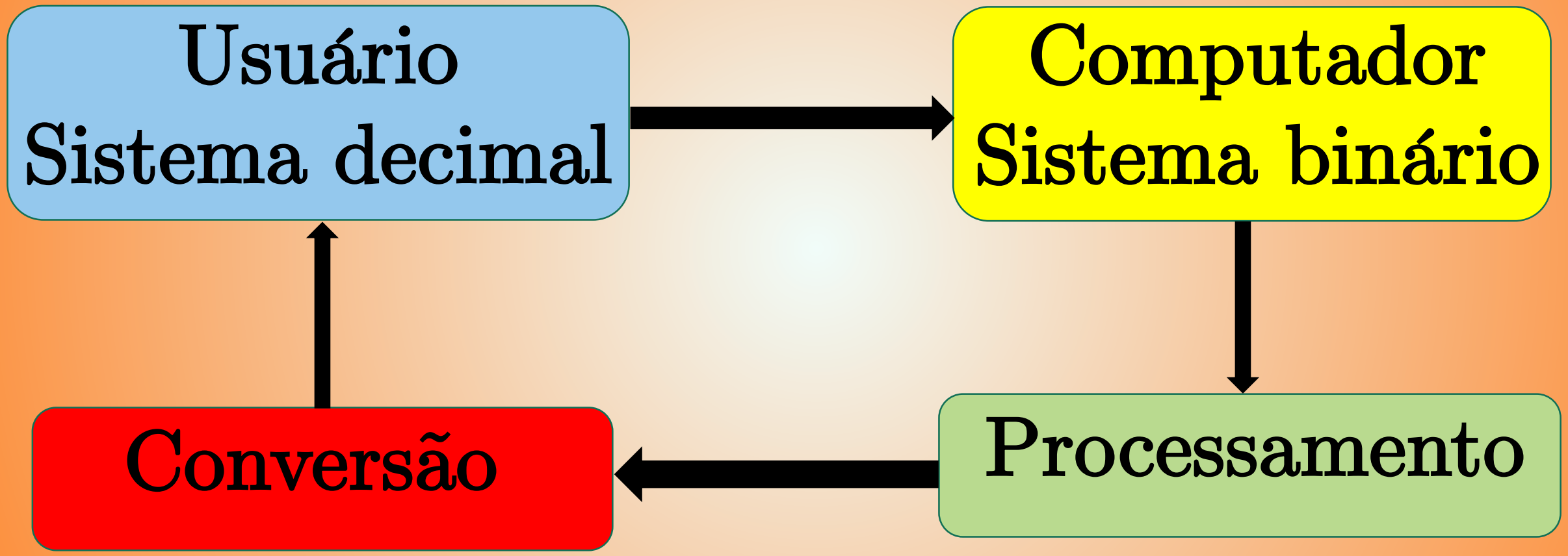
2. Representação de números

Exemplo: cálculo de uma área circular de raio 100 m.

a) $31400\ m^2 \longrightarrow \pi = 3.14$
b) $31416\ m^2 \longrightarrow \pi = 3.1416$
c) $31415.92654\ m^2 \longrightarrow \pi = 3.141592654$

O resultado depende da aproximação utilizada.
A representação depende também da base escolhida ou disponível e do número máximo de dígitos usados na sua representação.

Cálculo Numérico Computacional



Cálculo Numérico Computacional

Sistemas numéricos → Sistemas de notação usados para representar quantidades abstratas denominadas números

↓

Sistemas dependem da base que se utiliza

← Base é o número de símbolos diferentes ou algarismos necessários para representar um número

Cálculo Numérico Computacional

A chamada conversão de base é a passagem de uma base para outra

A representação muda, mas o valor quantitativo é preservado

Cálculo Numérico Computacional

Tipos de numeração mais usuais:

- Base 10 (decimal) – 0 a 9
- Base 2 (binária) – 0 a 1
- Base 8 (octal) – 0 a 7
- Base 16 (hexadecimal) – 0 a F

Cálculo Numérico Computacional

Denary (base 10)	Hexadecimal (base 16)	Octal (base 8)	Binary (base 2)
0	0	000	00000000
1	1	001	00000001
2	2	002	00000010
3	3	003	00000011
4	4	004	00000100
5	5	005	00000101
6	6	006	00000110
7	7	007	00000111
8	8	010	00001000
9	9	011	00001001
10	A	012	00001010
11	B	013	00001011
12	C	014	00001100
13	D	015	00001101
14	E	016	00001110
15	F	017	00001111

Cálculo Numérico Computacional

Denary (base 10)	Hexadecimal (base 16)	Octal (base 8)	Binary (base 2)
16	10	020	00010000
17	11	021	00010001
18	12	022	00010010
19	13	023	00010011
20	14	024	00010100
21	15	025	00010101
22	16	026	00010110
23	17	027	00010111
24	18	030	00011000
25	19	031	00011001
26	1A	032	00011010
27	1B	033	00011011
28	1C	034	00011100
29	1D	035	00011101
30	1E	036	00011110
31	1F	037	00011111

Cálculo Numérico Computacional

Base Binária:

- Ótimo para representação eletrônica
- Difícil para nossa visualização

Quanto menor a base, mais algarismos são necessários para representar os números.

Cálculo Numérico Computacional

$$(482)_{10} = 4 \times 10^2 + 8 \times 10^1 + 2 \times 10^0$$

$$(10011)_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + \\ + 1 \times 2^1 + 1 \times 2^0$$

Cálculo Numérico Computacional

3. Conversão de números – decimal/binário

Convertendo para o sistema decimal:

$$(10011)_2 = 1\times \mathbf{2^4} + \mathbf{0} \times \mathbf{2^3} + \mathbf{0} \times \mathbf{2^2} + \mathbf{1} \times \mathbf{2^1} + \mathbf{1} \times \mathbf{2^0} = 16 + 0 + 0 + 2 + 1 = 19$$

Cálculo Numérico Computacional

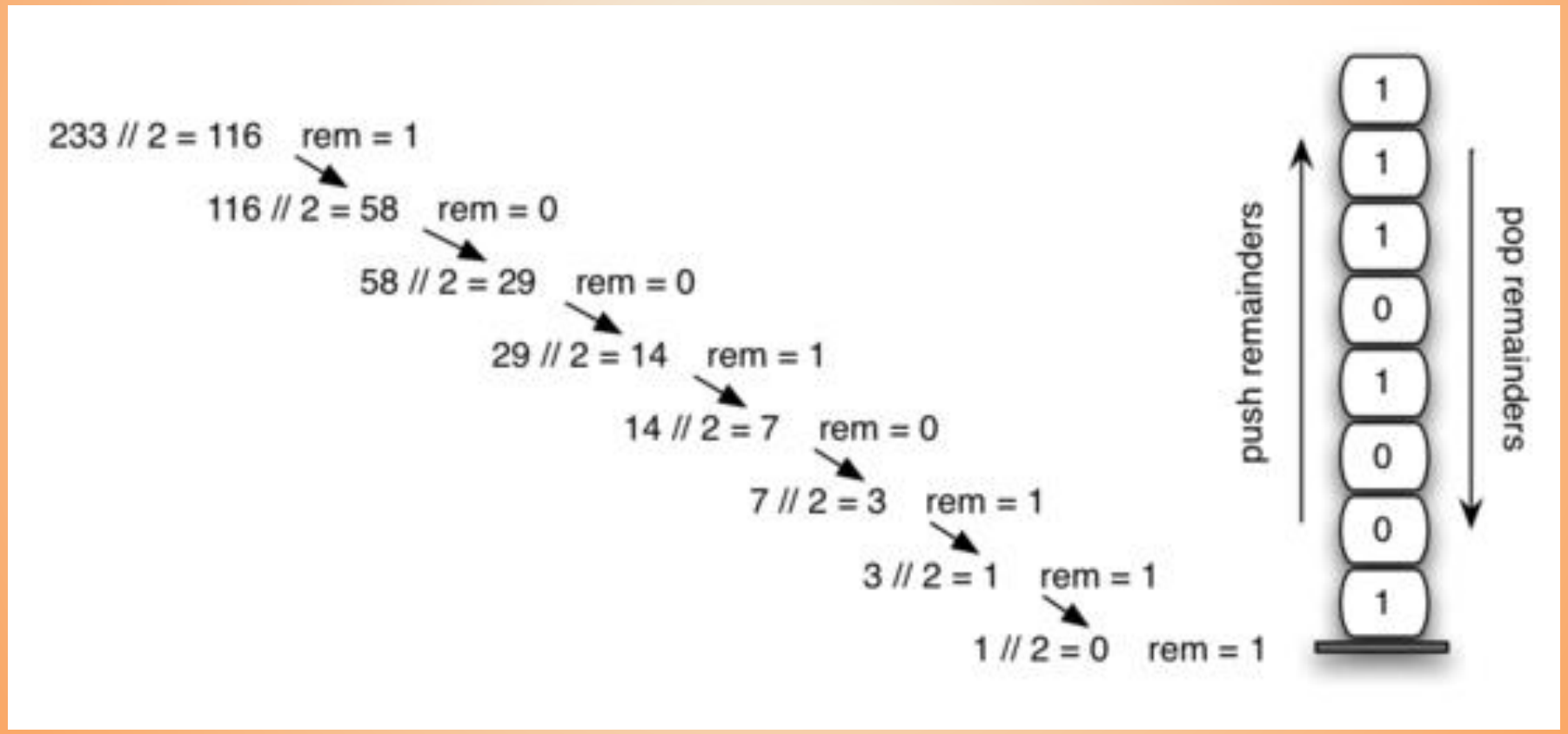
```
// C++ program to convert binary to decimal
#include<iostream>
using namespace std;

// Function to convert binary to decimal
int binaryToDecimal(int n)
{
    int num = n;
    int dec_value = 0;

    // Initializing base value to 1, i.e 2^0
    int base = 1;

    int temp = num;
    while (temp)
    {
        int last_digit = temp % 10;
        temp = temp/10;
        dec_value += last_digit*base;

        base = base*2;
    }

    return dec_value;
}
```

Cálculo Numérico Computacional

```
# Python3 program to convert
# binary to decimal

# Function to convert
# binary to decimal
def binaryToDecimal(n):
    num = n;
    dec_value = 0;

    # Initializing base
    # value to 1, i.e 2^0
    base = 1;

    temp = num;
    while(temp):
        last_digit = temp % 10;
        temp = int(temp / 10);

        dec_value += last_digit * base;
        base = base*2;
    return dec_value;
```

Cálculo Numérico Computacional

Agora, do sistema decimal para binário:



Cálculo Numérico Computacional

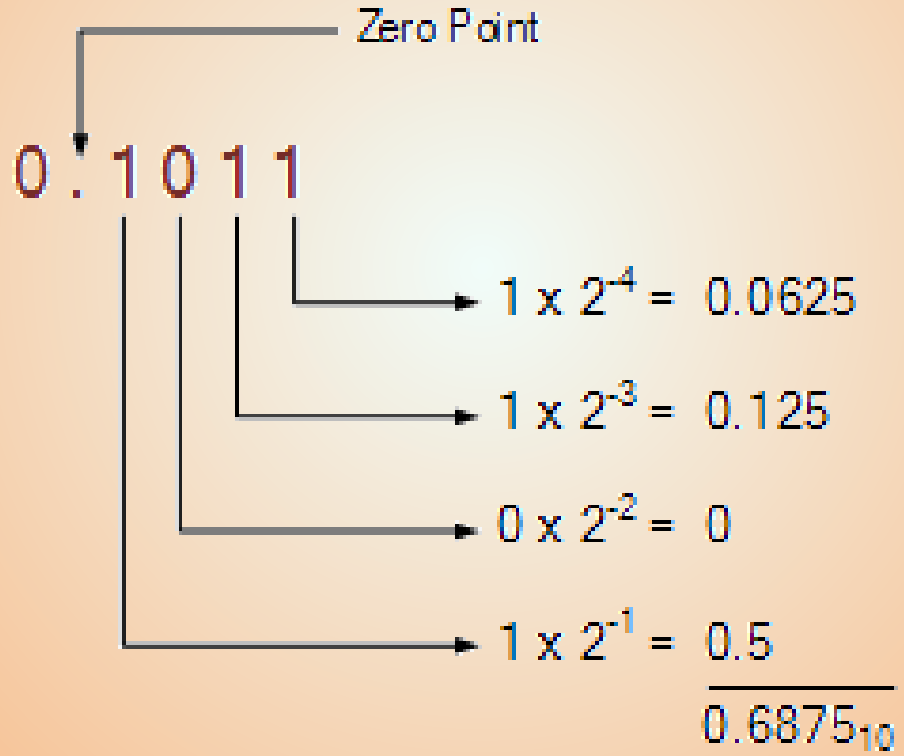
```
// C++ program to convert a decimal
// number to binary number

#include <iostream>
using namespace std;

// function to convert decimal to binary
void decToBinary(int n)
{
    // array to store binary number
    int binaryNum[1000];

    // counter for binary array
    int i = 0;
    while (n > 0) {

 // storing remainder in binary
array
        binaryNum[i] = n % 2;
        n = n / 2;
        i++;
    }

    // printing binary array in reverse
order
    for (int j = i - 1; j >= 0; j--)
        cout << binaryNum[j];
}
```

Cálculo Numérico Computacional

```
# Python3 program to convert a
# decimal number to binary number

# function to convert
# decimal to binary
def decToBinary(n):

    # array to store
    # binary number
    binaryNum = [0] * n;

    # counter for binary array
    i = 0;
    while (n > 0):

        # storing remainder
        # in binary array
        binaryNum[i] = n % 2;
        n = int(n / 2);
        i += 1;

    # printing binary array
    # in reverse order
    for j in range(i - 1, -1, -1):
        print(binaryNum[j], end = "");
```

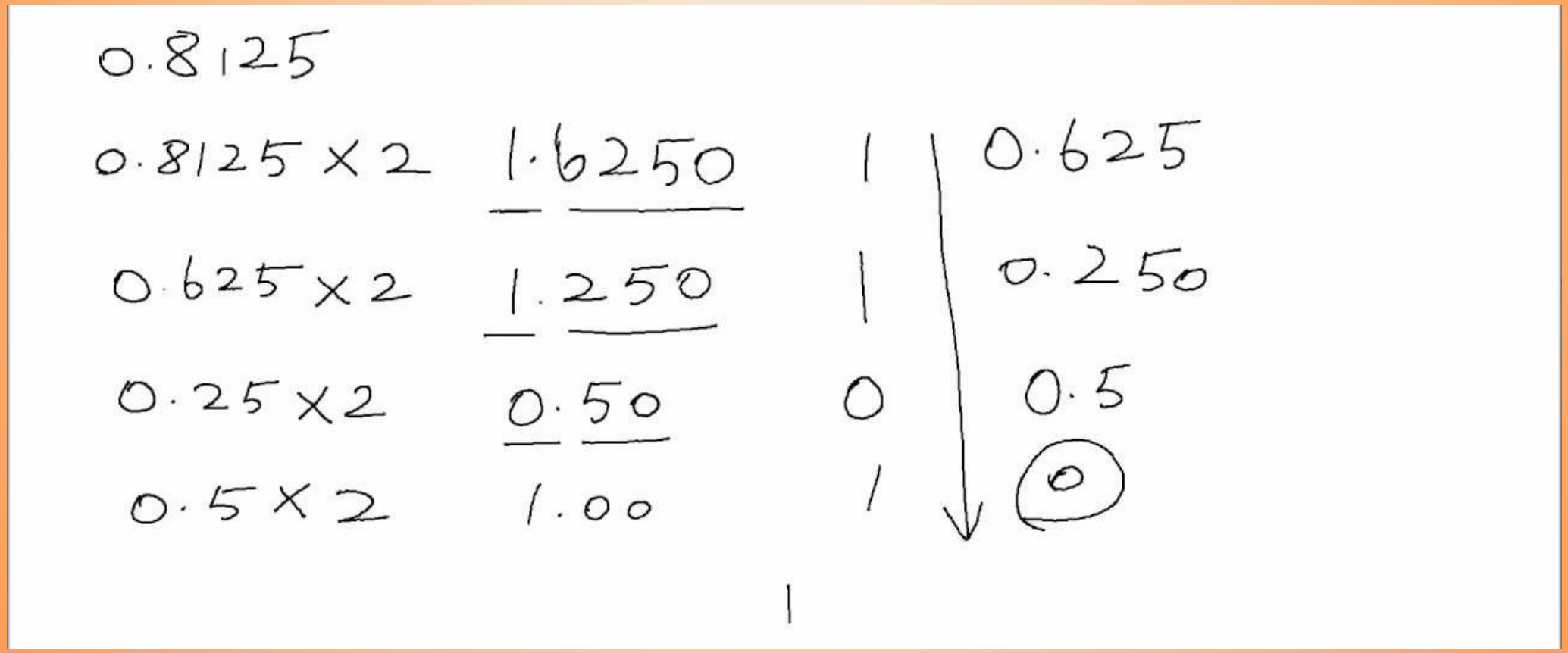
Cálculo Numérico Computacional

Números fracionários – base 2 para base 10:



Cálculo Numérico Computacional

Números fracionários – base 10 para base 2:



Cálculo Numérico Computacional

Na transformação da base 10 para base 2 sempre será possível para números inteiros; no entanto, nem sempre será possível escrever quando for número fracionário em representação finita.

Exemplo:

$$(0{,}6)_{10} = ?$$

Cálculo Numérico Computacional

Conclusão:

Série infinita, ou seja, o computador não conseguirá armazenar com exatidão o número em binário.

Erro de conversão de base

EXERCÍCIOS

Cálculo Numérico Computacional

4. Aritmética de ponto flutuante

Um número real, no sistema da máquina, é representado na forma:

$$\pm, (.\, d_1 d_2 \ldots d_t) \times \beta^e \quad \text{ou} \quad (-1)^s \times (.\, d_1 d_2 \ldots d_t) \times \beta^e$$

Onde: β é a base em que a máquina opera;
t é o número de dígitos na mantissa; $0 \leq d_j \leq (\beta - 1)$, $j = 1, \ldots, t$, $d_1 \neq 0$
e é o expoente no intervalo $[l, u]$
s é o sinal (0 se for positivo e 1 se for negativo)

Cálculo Numérico Computacional

A representação utilizada nas máquinas digitais é chamada NOTAÇÃO DE PONTO FLUTUANTE, na qual o espaço é dividido em:

- Sinal do número;
- Parte fracionária – MANTISSA;
- Área para expoente.

Cálculo Numérico Computacional

O espaço de armazenamento é limitado

O que se armazena são **INTERVALOS DISCRETOS**

A precisão depende do espaço disponível para o armazenamento.

Cálculo Numérico Computacional

Precisão simples

- 32 bits ou 4 bytes;
- 1 bit para o sinal do número (+ ou -);
- 8 bits para armazenar um número inteiro que é o expoente da base;
- 23 bits para a mantissa.

±	$e_1 e_2 \dots e_8$	$d_1 d_2 \dots d_{23}$

Precisão dupla

- 64 bits ou 8 bytes;
- 1 bit para o sinal do número (+ ou -);
- 11 bits para armazenar um número inteiro que é o expoente da base;
- 52 bits para a mantissa.

±	$e_1 e_2 \dots e_{11}$	$d_1 d_2 \dots d_{52}$

Cálculo Numérico Computacional

Na aritmética de ponto flutuante (*Floating-Point System*), um sistema é representado por 4 números e abrange um subconjunto de números reais.

$$F(\beta, t, m, M)$$

$\boldsymbol{\beta}$: base utilizada;
$\boldsymbol{t}$: tamanho ou número de dígitos (precisão);
$\boldsymbol{m}$ e $\boldsymbol{M}$: menor e maior expoentes, respectivamente.

Cálculo Numérico Computacional

Exemplo: $F(2,8,-4,3)$

$x =$	0	010	11100110
$y =$	0	010	11100111

$$x = (-1)^0 \times 2^2 \times (0.11100110) = (11.100110)_2 = (3.59375)_{10}$$

$$y = (-1)^0 \times 2^2 \times (0.11100111) = (11.100111)_2 = (3.609375)_{10}$$

Cálculo Numérico Computacional

x e y são dois números consecutivos neste sistema de representação, sendo que o número decimal 3,6 não possui representação exata.

Exemplo: $F(2,3,-1,2)$

- Base 2 com mantissa 0 ou 1 sendo que na forma normalizada, o primeiro dígito é obrigatoriamente 1
- Mantissa 0.100; 0.101; 0.110; 0.111
- Expoente variando de $\mathbf{2^{-1}}$ a $\mathbf{2^{2}}$

Cálculo Numérico Computacional

Quantidade de números representáveis:

$$N = 2 \times (\beta - 1) \times \beta^{t-1} \times (M - m + 1) + 1$$

Quanto maior o número de dígitos na mantissa, maior a precisão dos números representáveis e menor o intervalo entre dois números.

Cálculo Numérico Computacional

Exemplo: $\boldsymbol{\beta = 10,\ t = 3,\ e \in [-5; 5]}$

Os números serão representados na forma:

$$\mathbf{0}.\boldsymbol{d_1 d_2 d_3} \times \mathbf{10}^{\boldsymbol{e}}$$

$$\mathbf{0} \leq \boldsymbol{d_j} \leq (\mathbf{10} - \mathbf{1}), \boldsymbol{d_1} \neq \mathbf{0}, \boldsymbol{e} \in [-\mathbf{5}; \mathbf{5}]$$

Cálculo Numérico Computacional

Menor número possível: $m = 0.100 \times 10^{-5}$

Maior número possível: $M = 0.999 \times 10^{5}$

Considerando: $G = \{x \in \Re \mid m \leq |x| \leq M\}$

Caso 1) $x \in G \longrightarrow$ Truncamento ou arredondamento

Caso 2) $|x| < m \longrightarrow$ *Underflow*

Caso 3) $|x| > M \longrightarrow$ *Overflow*

Cálculo Numérico Computacional

Exemplo 3

Dar a representação dos números a seguir num sistema de aritmética de ponto flutuante de três dígitos para $\beta = 10$, $m = -4$ e $M = 4$.

x	Representação obtida por arredondamento	Representação obtida por truncamento
1.25	0.125×10	0.125×10
10.053	0.101×10^2	0.100×10^2
–238.15	-0.238×10^3	-0.238×10^3
2.71828...	0.272×10	0.271×10
0.000007	(expoente menor que –4)	=
718235.82	(expoente maior que 4)	=

Cálculo Numérico Computacional

Represente no sistema $F(10, 3, 5, 5)$ os números

$$x_1 = 1234.56, \quad x_2 = -0.00054962, \quad x_3 = 0.9995,$$
$$x_4 = 123456.7, \quad x_5 = 0.0000001.$$

Resposta:

$$\texttt{fl}(x_1) = 0.123 \times 10^4, \quad \texttt{fl}(x_2) = -0.550 \times 10^{-3},$$
$$\texttt{fl}(x_3) = 0.100 \times 10^1.$$

Para x_4 e x_5 tem-se *overflow* e *underflow*, respectivamente.

Cálculo Numérico Computacional

Adição / subtração

- Escolher o número com menor expoente entre x e y e deslocar sua mantissa para a direita um número de dígitos igual à diferença absoluta entre os respectivos expoentes;
- Colocar o expoente do resultado igual ao maior expoente entre x e y;
- Executar a adição/subtração das mantissas e determinar o sinal do resultado;
- Normalizar o valor do resultado, se necessário;
- Arredondar o valor do resultado, se necessário;
- Verificar se houve *overflow/underflow*.

Cálculo Numérico Computacional

Multiplicação

- Colocar o expoente do resultado igual à soma dos expoentes de x e y;
- Executar a multiplicação das mantissas e determinar o sinal do resultado;
- Normalizar o valor do resultado, se necessário;
- Arredondar o valor do resultado, se necessário;
- Verificar se houve *overflow/underflow*.

Cálculo Numérico Computacional

Divisão

- Colocar o expoente do resultado igual à diferença dos expoentes de x (dividendo) e y (divisor);
- Executar a divisão das mantissas e determinar o sinal do resultado;
- Normalizar o valor do resultado, se necessário;
- Arredondar o valor do resultado, se necessário;
- Verificar erros.

EXERCÍCIOS

Cálculo Numérico Computacional

Próxima aula:

Aula 03

- Erros absolutos e relativos;
- Arredondamento e truncamento;
- Análise de erros em aritmética de ponto flutuante;
- Instabilidade numérica.