Universidade Federal de Santa Catarina
Centro Tecnológico
Departamento de Informática e Estatística
Curso de Graduação em Ciências da Computação

# Sistemas Digitais

## INE 5406

## Aula 10-P

### Refinamento das especificações do Trabalho Prático:

### Tema 4 - Representação e adição de números em ponto flutuante segundo o padrão IEEE 754.

Prof. José Luís Güntzel
guntzel@inf.ufsc.br

www.inf.ufsc.br/~guntzel/ine5406/ine5406.html

## Números Fracionários

**Além de inteiros com e sem sinal, as linguagens de programação suportam também números fracionários.**

**Exemplos:**

**$3,14159265..._{10}$** ($\pi$)

**$2,71828..._{10}$** (e)

**$0,000000001_{10} = 1,0_{10} \times 10^{-9}$** (segundos em um nanossegundo)

**$3.155.760.000_{10} = 3,15576_{10} \times 10^{9}$** (segundos em um século)

**Notação científica normalizada**

# Números Fracionários

**Exemplo de Número em Notação Científica Normalizada:**

$1{,}0_{10} \times 10^{-9}$

**Exemplos de Número em Notação Científica Não-Normalizada:**

$0{,}1_{10} \times 10^{-8}$

$10{,}0_{10} \times 10^{-10}$

## Números Fracionários

**Números binários também podem ser representados em Notação Científica. Exemplo:**

$\mathbf{1{,}0_2 \times 2^{-1}}$

Para manter um número binário normalizado deve-sem ultiplicá-lo por uma base tal que ele fique com um dígito diferente de “0” imediatamente à direita da vírgula.

## Binários em Notação Científica

**O Formato utilizado é:**

$$\mathbf{1{,}xxxxxxxxx_2 \times 2^{yyyy}}$$

**Muitas vezes, o expoente é mostrado em decimal, embora na máquina, fique armazenado em binário…**

## Vantagens da Notação Científica Padronizada

1. **Facilita a troca de dados entre diferentes computadores**
2. **Facilita os algoritmos aritméticos envolvendo números em ponto flutuante**
3. **Melhora a precisão dos números armazenados em uma palavra de memória** (pois os algarismos não-significativos à direita da vírgula são substituídos por algarismos significativos)

## Formato para Precisão Simples

**Os números em ponto flutuante normalmente são representados como múltiplos do tamanho da palavra da máquina. Exemplo:**

| 31 | 30 29 28 27 26 25 24 23 | 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
|---|---|---|
| S | expoente | mantissa |
| | 8 bits | 23 bits |

$$(-1)^S \times F \times 2^E$$

**Onde:**

- **F representa o valor do campo da mantissa**
- **E representa o valor do campo do expoente**

# Intervalo de Representação

- **O número de bits da palavra da máquina é fixo** (definido quando de seu projeto)
- **Um bit a mais na mantissa corresponde a um bit a menos no expoente e *vice-versa***
- **Mais bits na mantissa => maior precisão do número** ("passo")
- **Mais bits no expoente => maior intervalo de representação**

# Formato para Precisão Simples

| 31 | 30 29 28 27 26 25 24 23 | 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
|---|---|---|
| S | expoente | mantissa |
| | 8 bits | 23 bits |

**Exemplos de números que podem ser representados:**

$\mathbf{2{,}0_{10} \times 10^{-38}}$

$\mathbf{2{,}0_{10} \times 10^{38}}$

**O que significa *overflow*, no caso desta representação?**

**Significa que o expoente é muito grande (em módulo) para ser armazenado no campo a ele reservado** (8 bits, para o caso acima)

## Formato para Precisão Simples

| 31 | 30 29 28 27 26 25 24 23 | 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
|---|---|---|
| S | expoente | mantissa |
| | 8 bits | 23 bits |

**E o que acontece se o número fracionário tiver muitos dígitos à direita da vírgula, de modo que não pode ser representado no campo da mantissa?**

**Exemplo: $111111111111111111111111_2$ (24 “1”s)**

- **Este “evento exepcional” é denominado de *underflow***
- **Exige que o número seja truncado ou arredondado**

# Formato para Precisão Dupla

## Reduzindo as chances de ocorrência de *overflow* e de *underflow*

| 31 | 30 29 28 27 26 25 24 23 22 21 20 | 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
|---|---|---|
| S | expoente | mantissa |
| | 11 bits | 20 bits |

| 31 30 29 28 27 ….. 4 3 2 1 0 |
|---|
| Continuação da mantissa |
| 32 bits |

**2 palavras de 32 bits**

$$(-1)^S \times F \times 2^E$$

**Onde:**

- **F representa o valor do campo da mantissa**
- **E representa o valor do campo do expoente**

## Formato para Precisão Dupla

### Reduzindo as chances de ocorrência de *overflow* e de *underflow*

| 31 | 30 29 28 27 26 25 24 23 22 21 20 | 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
|---|---|---|
| S | expoente | mantissa |
| | 11 bits | 20 bits |

| 31 30 29 28 27 ….. 4 3 2 1 0 |
|---|
| Continuação da mantissa |
| 32 bits |

### Exemplos de números que podem ser representados:

$2{,}0_{10} \times 10^{-308}$

$2{,}0_{10} \times 10^{308}$

# Padrão IEEE 754

## Adotado pelos fabricantes de processadores a partir de 1980

| 31 | 30 29 28 27 26 25 24 23 | 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
|---|---|---|
| S | expoente | mantissa |
| | 8 bits | 23 bits |

- **Valor "1" à esquerda da vírgula está implícito** (assim, a mantissa fica com 24 bits na precisão simples e 53 bits na precisão dupla)
- **Quando o expoente valer 0, o hardware desconsidera o "1" implícito** ($000\ldots00_2$ representa o valor zero)
- **Os demais números são representados por:**

$$(-1)^S \times (1+ \text{matissa}) \times 2^E$$

## Padrão IEEE 754

- **Facilitou a portabilidade dos programas que trabalham com números em ponto flutuante**
- **Melhorou a qualidade das operações aritméticas realizadas pelos computadores**

## Padrão IEEE 754

| 31 | 30 29 28 27 26 25 24 23 | 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
|---|---|---|
| S | expoente | mantissa |
| | 8 bits | 23 bits |

- **Enumerando os bits da mantissa da esquerda para a direita por $m_1$, $m_2$, $m_3$ , …, então o valor do número será dado por:**

$$(-1)^S \times (1 + (m_1 \times 2^{-1}) + (m_2 \times 2^{-2}) + (m_3 \times 2^{-3}) + \ldots) \times 2^E$$

# Padrão IEEE 754

**Características Marcantes do Padrão:**

- **Bit de Sinal aparece mais à esquerda (uso de sinal-magnitude)**
- **O expoente aparece à esquerda da mantissa**
- **Os bits da mantissa aparecem em ordem inversa**

**Motivos para estas características:**

- **Permitir o processamento rápido de números por meio de instruções de comparações inteiras (inclusive para o ordenamento de números)**

# Padrão IEEE 754

**Problema: como comparar os expoentes?**

**Exemplo, usando complemento de dois:**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**expoente = +1**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**expoente = -1**

## Padrão IEEE 754

**Solução: aplicar um deslocamento positivo, de modo que o menor expoente (i.e., o mais negativo) seja representado por zero**

**Isto é chamado de notação com peso ou notação em excesso de (valor)**

**Padrão IEEE 754 usa excesso de 127:**

- **Expoente -1 => $-1 + 127_{10} = 126_{10}$ => $0111\ 1110_2$**
- **Expoente +1 => $+1 + 127_{10} = 128_{10}$ => $1000\ 0000_2$**

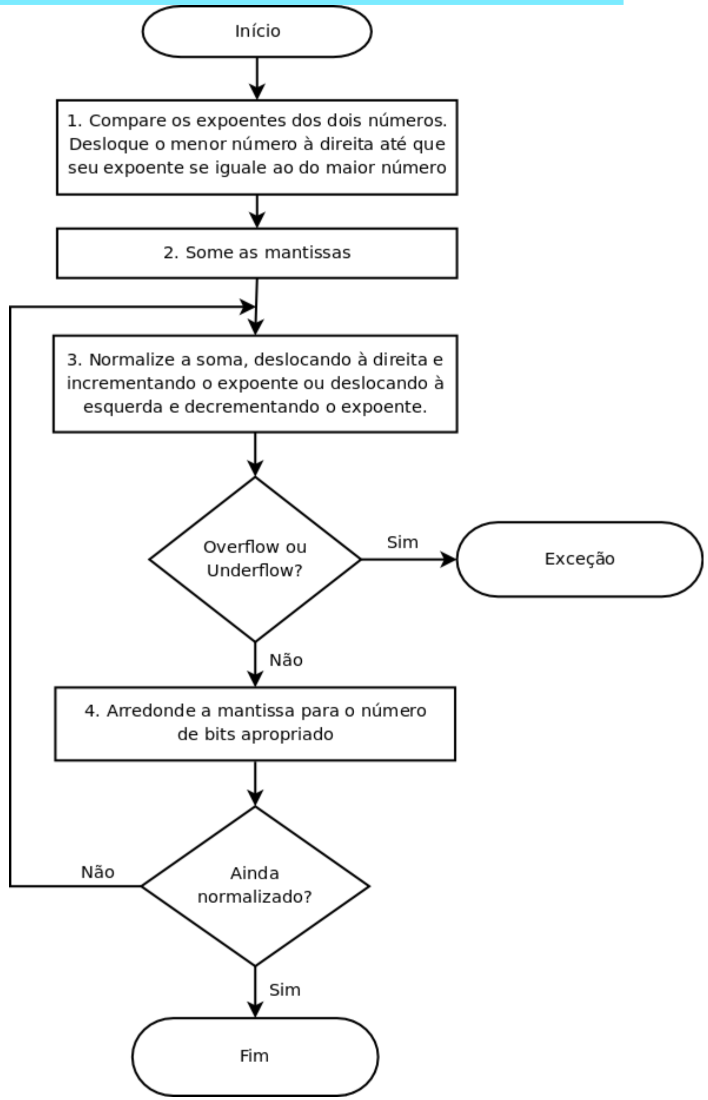
# Padrão IEEE 754

| | | |
|---|---|---|
| 31 | 30 29 28 27 26 25 24 23 | 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
| S | expoente | mantissa |
| | 8 bits | 23 bits |

$$(-1)^{S} \times (1+ \text{matissa}) \times 2^{(E\ -\ \text{excesso})}$$

- **Para precisão simples, expoente em excesso de 127**
- **Para dupla precisão, expoente em excesso de 1023**

## Adição de Números em Ponto Flutuante



## Referências Bibliográficas

PATTERSON, David A.; HENNESSY, John L. , "Computer Organization and Design: The Hardware/Software Interface", 3rd edition, Morgan Kaufmann Publishers, San Francisco, California, USA, 2007. chapter 3.

PATTERSON, David A.; HENNESSY, John L. "Organização e Projeto de Computadores: a interface hardware/software." 3ª Edição. Rio de Janeiro: LTC, 2005. capítulo 3

PATTERSON, David A.; HENNESSY, John L. "Organização e projeto de computadores: a interface hardware/sofware." 2ª edição Rio de Janeiro: LTC, 2000. capítulo 4.

STALLINGS, William. Arquitetura e Organização de Computadores. 5ª edição. São Paulo: Prentice-Hall, 2002.

GOLDBERG, David. What Every Computer Scientist Should Know About Floating-Point Arithmetic. ACM Computing Surveys. 23(1). P.5-48.

http://grouper.ieee.org/groups/754/

http://steve.hollasch.net/cgindex/coding/ieeefloat.html