



Universidade Presbiteriana  
**Mackenzie**

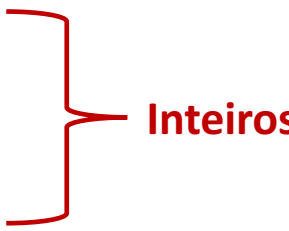
Faculdade de Computação e Informática  
Hardware para Computação

# Representação de Números

(Inteiros, Frações – positivos e negativos)

*Prof. Jean M. Laine*

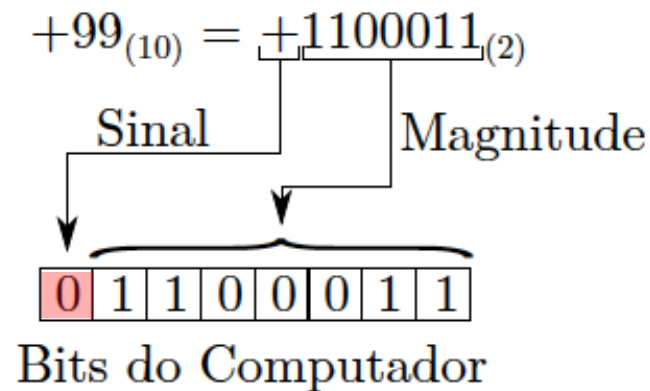
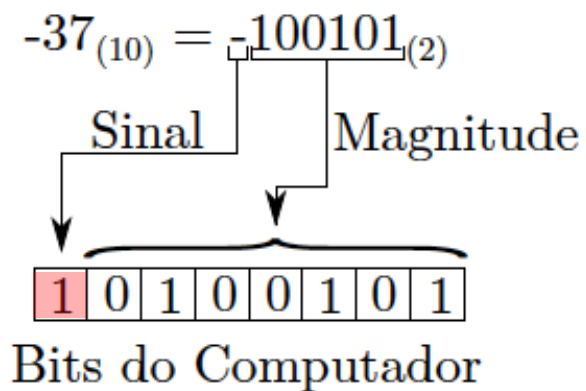
# Representação de Dados

- Sinal e Magnitude
  - Complemento de Dois
- 
- Inteiros
- **Representação em Ponto Flutuante – IEEE 754**
    - Padrão criado em 1985 pela IEEE
    - Passou a ser adotado por todos os computadores

# Sinal e Magnitude

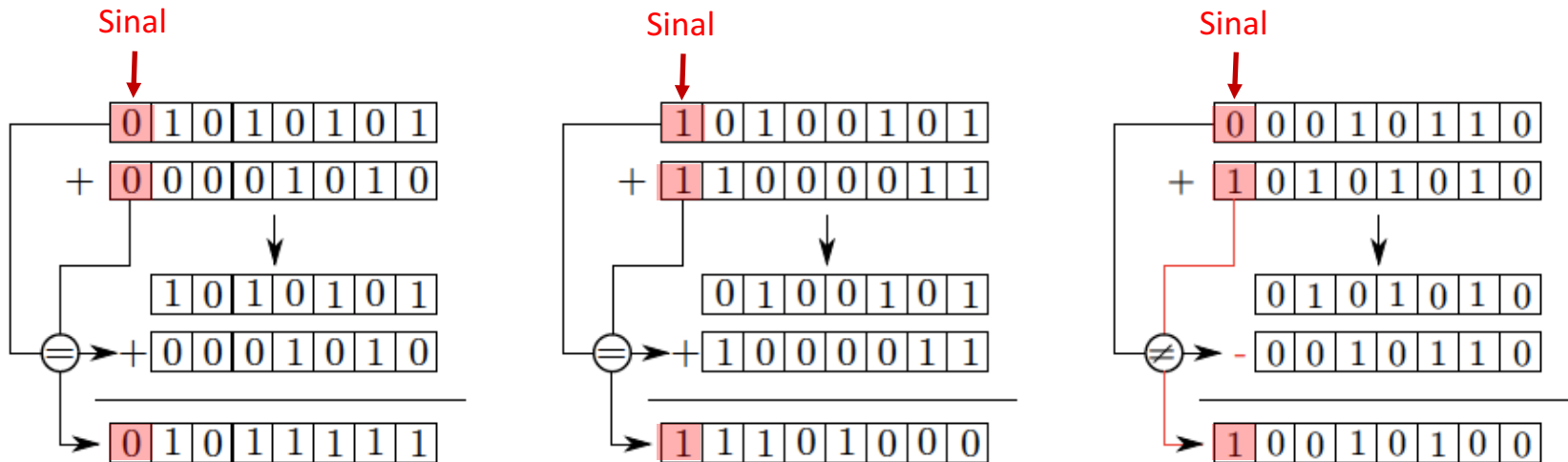
- Representação para números inteiros positivos e negativos
  - Foi usada em computadores antigos
- Os números são organizados em 2 partes:
  - **Sinal** (bit usado para representar o sinal do número (0: + e 1: -))
  - **Magnitude** (módulo ou valor absoluto do dado)
- O bit reservado para o sinal é sempre o mais significativo: o bit mais à esquerda no número
- A magnitude é simplesmente representada em base 2 com  $n - 1$  bits
  - $n$  é o número de bits usado pela máquina para representar o número.

## Exemplos usando 8 bits: -37 e +99



# Operação de Soma

- Se os **bits de sinal** dos dois números são **iguais**:
  - Separe as magnitudes e as some
  - Bit de sinal do resultado é igual ao bit de sinal dos operandos
- **Caso contrário**
  - Separe as magnitudes e subtraia a menor da maior.
  - O bit de sinal do resultado é igual ao bit de sinal do operando de maior magnitude.



## Outras operações

- Para **subtração**, pode-se trocar o sinal do subtraendo (inverter bit de sinal) e transformar em soma a operação. Exemplo:

$$8 - 3 = 8 + (-3)$$

- Para **divisão** e **multiplicação**, opera-se apenas sobre a magnitude
  - Se operandos têm mesmo sinal, resultado terá bit de sinal 0.
  - Se operandos têm sinal diferente, resultado terá bit de sinal 1.

# Limites de Representação do Sinal e Magnitude

- Suponha 8 bits para representar um número em Sinal e Magnitude
- Qual o maior número (maior positivo) que pode ser representado?
  - Primeiro bit 0 e todos os outros iguais a 1.
  - Para 8 bits:  $0\ 1111111_2 = 127_{10}$ .
- Qual o menor número (negativo) que pode ser representado?
  - Primeiro bit é 1 e todos os outros iguais a 1.
  - Para 8 bits:  $1\ 1111111_2 = -127_{10}$ .

E se usarmos 4 bits?

Qual o limite de representação?



# Generalizando...

- **Maior número:**

$$2^{(n-1)} - 1$$

- **Menor número:**

$$-2^{(n-1)} - 1$$

- **Exemplos:**

- Para  $n = 8$ : de -127 a 127
- Para  $n = 16$ : de -32767 a 32767.
- Para  $n = 32$ : de -2147483647 a 2147483647

# Problema

- Duplicidade do valor 0. Considere os números:

00000000 e 10000000

Assim, ambos são 0 (zero): **positivo** e **negativo**

# Exercícios

1. Represente os números seguintes em Sinal e Magnitude usando 8 bits de representação:
  - a. - 80
  - b. 75
  - c. - 103
  - d. 121
  - e. - 65
2. Represente os números seguintes em Sinal e Magnitude usando 16 bits de representação:
  - a. - 198
  - b. 246
  - c. - 12561
  - d. 23890

# Exercícios

3. Faça as seguintes operações, considerando que os números estão representados em sinal em magnitude, usando 8 bits:

- a)  $10011001 + 00100110$
- b)  $00011000 + 00110011$
- c)  $10100000 - 10001100$
- d)  $00011000 - 00000111$

# Complemento de 2 (C2)

## Representação de números inteiros

- **Números positivos:**
  - Idêntica ao número escrito em base 2.
  - Zeros a esquerda são adicionados para que número fique com  $n$  bits.
  - **Bit mais significativo tem que ser igual a 0**
- **Números negativos:**
  - Escreve-se o número em binário
  - **Invertem-se os bits**
  - **Soma-se um ao número**
- O Complemento de Dois é o esquema de representação mais popular nos computadores modernos.

- Exemplos (com 5 bits):

- . Complemento a dois de 01011:

- 1º inverte: 10100 ; 2º soma-se 1 =  $10100 + 1 = 10101$ .

- . Complemento a dois de 11001 é  $00110 + 1 = 00111$ .

- . Complemento a dois de 10010 é  $01101 + 1 = 01110$ .

- . Complemento a dois de 10101 é  $01010 + 1 = 01011$ .

# Propriedades

- Dado um número representado em Complemento de Dois com  $n$  bits, podemos estendê-lo para mais bits:
  - Se o número positivo, basta adicionar zeros à esquerda.
  - Se o número é negativo, basta adicionar uns.
  - De forma geral, repete-se o bit mais significativo tantas vezes quanto necessário.
- Exemplos:
  - 01101 com cinco bits tem o mesmo valor de 00001101 com 8 bits.
  - 10001 com cinco bits tem o mesmo valor de 11110001 com 8 bits

# Como determinar o valor de número?

- Se o número é positivo (bit mais significativo é 0), basta convertê-lo para decimal
- Se o número é negativo (bit mais significativo é 1):
  - Invertem-se os bits, e soma-se um
- Exemplo:

$$\begin{array}{rcl} 1101 & & (-3) \\ \underline{1100}+ & & (-4) \\ 1001 & & (-7) \end{array}$$



# Subtração

Transforma o subtraendo em negativo e faz a soma. Só inverter os bits e somar 1.

$$\begin{array}{rcl} 0111 & (7) \\ \underline{0011} - & (3) & \longrightarrow \begin{array}{rcl} 0111 & (7) \\ \underline{1101} + & (-3) \\ 0100 & (4) \end{array} \end{array}$$

# Exercícios

1. Represente os números seguintes em C2 usando 8 bits de representação:
  - a. - 80
  - b. - 75
  - c. - 103
  - d. 121
  - e. - 65
2. Represente os números seguintes em C2 usando 16 bits de representação:
  - a. - 198
  - b. - 246
  - c. - 12561



Universidade Presbiteriana  
**Mackenzie**

Faculdade de Computação e Informática  
Hardware para Computação

# Ponto Flutuante

[IEEE 754](#)

(float e double)

*Prof. Jean M. Laine*

# Os números

- Semelhante a notação científica:
  - $-3,45 \times 10^{32}$
  - $+1,2 \times 10^{-14}$
- Números binários =  $\pm 1,bbbbbb \times 2^e$ 
  - **Exemplo:**  $+1,0011 \times 2^{12}$
- Tipos de dados em linguagens:
  - **Float** (4 Bytes) e **double** (8 Bytes)
- Padrão especificado e definido por IEEE 754 (1985)
- 2 representações possíveis:
  - Precisão simples (32 bits): **float**
  - Precisão dupla (64 bits): **double**

# Tipos de dados

tipo	descrição	número de bits
byte	inteiro	8
short	inteiro	16
int	inteiro	32
long	inteiro	64
float	vírgula flutuante	32
double	vírgula flutuante	64
char	caracter	16
logical	booleano	8



# Ponto Flutuante: notação

single: 8 bits

single: 23 bits

double: 11 bits

double: 52 bits

S	Exponent	Fraction
---	----------	----------

$$x = (-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} + \text{Bias})}$$

S: sign bit (0  $\Rightarrow$  non-negative, 1  $\Rightarrow$  negative)

# Ponto Flutuante

- Os números devem ser representados na **forma normalizada** (1, alguma coisa):

$\pm 1, \dots$

- Parte inteira do número **sempre igual a 1**

Exemplos:

- 130,75 = ? Binário
- 0,125 = ? Binário

E na forma normalizada?

# Ponto Flutuante

single: 8 bits  
double: 11 bits

single: 23 bits  
double: 52 bits

S	Exponent	Fraction
---	----------	----------

$$x = (-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} + \text{Bias})}$$

S: sign bit (0  $\Rightarrow$  non-negative, 1  $\Rightarrow$  negative)

- Precisão Simples:

- Bias = **127**

- Precisão Dupla:

- Bias = **1023**



# Valores Especiais – convenção

Valor	Sinal	Expoente	Mantissa
Zero	0	0s	0s
+ Infinito	0	1s	0s
- Infinito	1	1s	0s
NaN	0	1s	Diferente de 0s

No padrão IEEE 754, os **NaN** (Not a Number), possuem sinal 0, expoente 1 e mantissa com qualquer valor, exceto tudo 0s, pois isso caracteriza +infinito, e representam exceções como divisão por zero, raiz de negativos etc.

Exemplo: -9,5 (PS)

Sinal negativo  $\rightarrow 1$

9,5 para binário  $\rightarrow 1001,1$

Forma Normal =  $1,0011 \times 2^3$

Agora que temos o expoente = 3, devemos normalizá-lo  
(**e+bias**):

$$3 + 127 = 130$$

Em binário temos  $3 = 11$  e  $127 = 1111111$ , somando os dois  
temos  $\rightarrow 10000010$  (130)

**Resultado** (sinal, **expoente**, **fração**):

1 **10000010** 00110000000000000000000000000000

# Convertendo número binário para decimal

Para converter binários de ponto flutuante para decimal, devemos fazer o inverso do que fizemos antes. Devemos identificar os componentes e dividí-los em sinal, expoente (8 ou 11 bits) e o restante será a mantissa. Não devemos esquecer de recompor a parte inteira, ou seja o 1. Exemplo:

1	10000010	001100000000000000000000
sinal	expoente	fração

expoente (**e - bias**)  $\rightarrow 10000010 = 130$ ; logo  $130 - 127 = 3$

reconstituindo a **parte inteira** (1) e adicionando-a a **mantissa**  $\rightarrow$  **10011**

adicionando a vírgula e o expoente  $\rightarrow 1,0011 \times 2^3$

deslocando a vírgula, de acordo com o expoente  $\rightarrow 1001,1$

convertendo para decimal  $\rightarrow 9,5$

adicionando o sinal  $\rightarrow -9,5$

# Exercícios

1. Represente os números usando IEEE 754 com precisão simples e com precisão dupla:
  - a. 116,125
  - b. -32,75
  - c. -29,500
  - d. 75,250
2. Responda quais valores estão representados se os números a seguir estão na notação IEEE 754 precisão simples:
  - a. 1 10011010 001110000000000000000000
  - b. 0 10100010 101101000000000000000000