Algoritmos e Estrutura de Dados I

Aula 03

Introdução a Computação – Parte 2

Prof. Dr. Dilermando Piva Jr

1º Semestre - CDN

PROBLEMA DO MUNDO REAL

Imagine que você é um engenheiro de dados trabalhando em uma empresa de telecomunicações. A empresa está enfrentando desafios na transmissão eficiente de dados através de sua rede de comunicação. A largura de banda é limitada, e há uma necessidade urgente de otimizar a comunicação para garantir a transmissão rápida e confiável de informações.

Sua tarefa como um dos engenheiros da empresa, é buscar uma solução para comprimir os dados antes de transmiti-los pela rede. A ideia é representar as informações de uma maneira mais eficiente, minimizando o uso de recursos de transmissão.

DETALHAMENTO

Representação Binária e Hexadecimal:

- **Converta um conjunto de dados textuais em representação binária e hexadecimal.**
- **Explore como diferentes tipos de dados são representados em sistemas binários e hexadecimais.**

Codificação ASCII/Unicode:

- **Analise como os caracteres são codificados usando os padrões ASCII ou Unicode.**
- **Considere a eficiência desses sistemas na representação de informações textuais.**

Compressão de Dados:

- **Pesquise e aplique técnicas básicas de compressão de dados.**
- **Compare a eficácia da compressão em diferentes tipos de dados.**

Sistemas de Numeração:

- **Converta números entre sistemas numéricos, incluindo binário, decimal e hexadecimal.**
- **Explique como os computadores utilizam sistemas de numeração na representação de dados.**

DETALHAMENTO

1. Imersão na Experiência Concreta:

- **Os alunos recebem um conjunto de dados para converter e analisar.**
- **Experimentam a compressão de dados manualmente.**

2. Observação Reflexiva e Obtendo Feedback:

- **Os alunos compartilham suas observações sobre a eficiência das representações.**
- **Recebem feedback sobre as técnicas de compressão utilizadas.**

3. Conceituação Abstrata e Correção de Erros:

- **Aulas expositivas sobre sistemas numéricos e codificação de dados.**
- **Correção de erros com base nas observações anteriores.**

4. Experimentação Ativa e Utilização de Metacognição:

- **Os alunos aplicam técnicas de compressão avançadas.**
- **Refletem sobre o impacto das diferentes representações.**

5. Reprocessamento e Aprendizado Contínuo:

- **Discussão sobre a evolução das técnicas de compressão.**
- **Conexão com futuros conceitos, como representação de números reais.**

6. Colaboração e Envolvimento no Mundo Real:

- **Os alunos colaboram para identificar soluções eficientes.**
- **Compartilham resultados em portfólios individuais.**

REFLEXÃO E FEEDBACK

Após a imersão no problema, os alunos devem refletir sobre os desafios encontrados durante a investigação.

Encoraje discussões em grupo para que compartilhem suas experiências, erros e insights.

Ofereça feedback construtivo sobre as abordagens utilizadas pelos alunos, destacando pontos positivos e áreas de melhoria.

A Informação e sua Representação

- O computador, sendo um equipamento eletrônico, armazena e movimenta as informações internamente sob forma eletrônica; tudo o que faz é reconhecer dois estados físicos distintos, produzidos pela eletricidade, pela polaridade magnética ou pela luz refletida – em essência, eles sabem dizer se um “interruptor” está ligado ou desligado.

- O computador, por ser uma máquina eletrônica, só consegue processar duas informações: a presença ou ausência de energia.

- Para que a máquina pudesse representar eletricamente todos os símbolos utilizados na linguagem humana, seriam necessários mais de 100 diferentes valores de tensão (ou de corrente).

A Informação e sua Representação

Tipos de grandezas

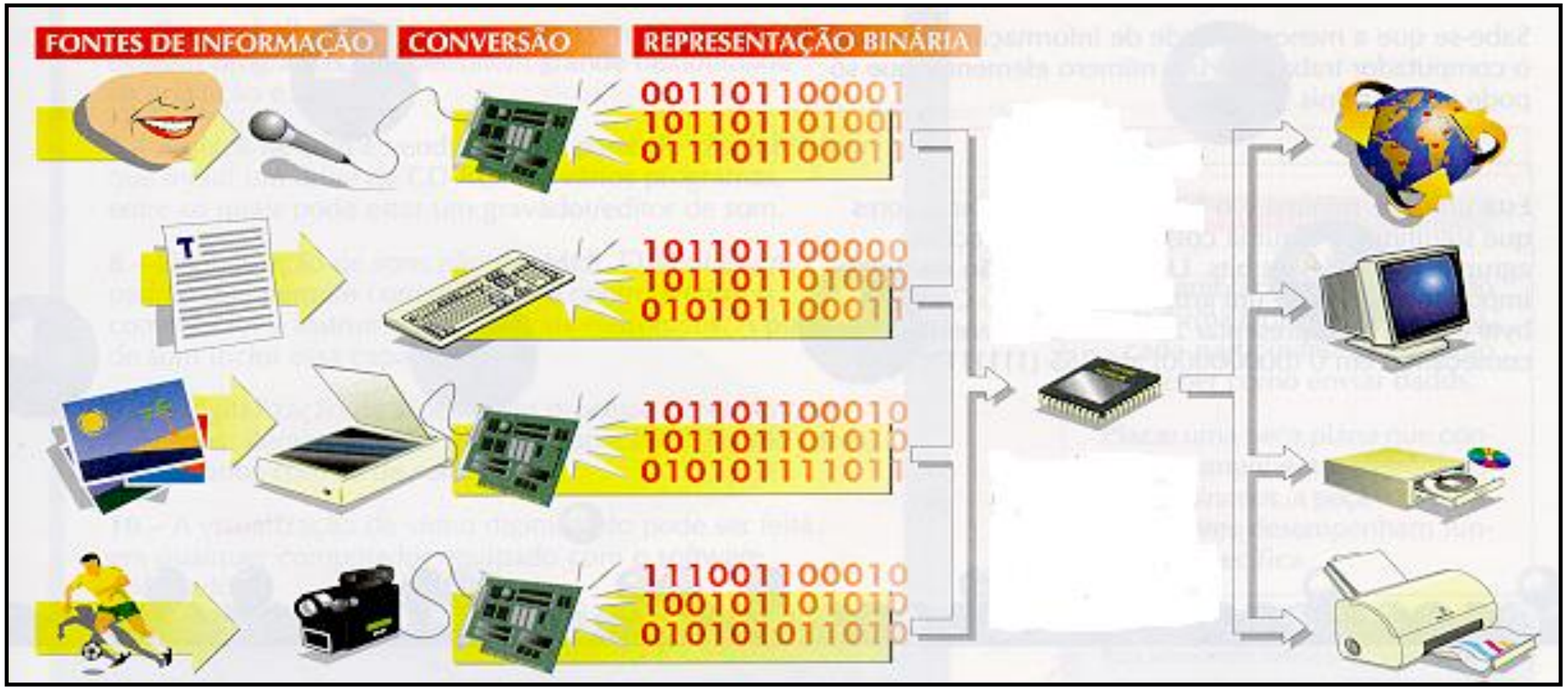
- **Analógica** ≡ contínua
- **Digital** ≡ discreta (passo a passo)

- **Computadores analógicos** – Trabalham com sinais elétricos de infinitos valores de tensão e corrente (modelo continuamente variável, ou **analogia**, do que quer que estejam medindo).

- **Computadores digitais** – Trabalham com dois níveis de sinais elétricos: **alto e baixo**. Representam dados por meio de um símbolo facilmente identificado (***dígito***).

A Informação e sua Representação

Como os computadores modernos representam as informações?



A Informação e sua Representação

- **Para o computador, tudo são números.**

- **Computador Digital** ⇒ Normalmente a informação a ser processada é de forma numérica ou texto ⇒ codificada internamente através de um **código numérico**.

- Código mais comum ⇒ **BINÁRIO**

Por que é utilizado o sistema binário ?

A Informação e sua Representação

- Como os computadores representam as informações utilizando apenas dois estados possíveis, eles são totalmente adequados para números binários.

O – desligado

1 – ligado

- Número binário no computador: **bit** [de "**B**inary dig**IT**"]
  - A unidade de informação.
  - Uma quantidade computacional que pode tomar um de dois valores, tais como verdadeiro e falso ou 0 e 1.

Um bit está ligado (*set*) quando vale 1, desligado ou limpo (*reset* ou *clear*) quando vale 0; comutar, ou inverter (*toggle* ou *invert*) é passar de 0 para 1 ou de 1 para 0. (lógica positiva)

A Informação e sua Representação

- Um bit pode representar apenas **2** símbolos (0 e 1)

- **Necessidade** - unidade maior, formada por um conjunto de bits, para representar números e outros símbolos, como os caracteres e os sinais de pontuação que usamos nas linguagens escritas.

- Unidade maior (**grupo de bits**) - precisa ter bits suficientes para representar todos os símbolos que possam ser usados:
  - dígitos numéricos,
  - letras maiúsculas e minúsculas do alfabeto,
  - sinais de pontuação,
  - símbolos matemáticos e assim por diante.

A Informação e sua Representação

Necessidade:

Caracteres alfabéticos maiúsculos	**26**
Caracteres alfabéticos minúsculos	**26**
Algarismos	**10**
Sinais de pontuação e outros símbolos	**32**
Caracteres de controle	**24**
Total	**118**

A Informação e sua Representação

Capacidade de representação:

Bits	Símbolos
2	4
3	8
4	16
5	32
6	64
7	128
8	**256**
9	512
10	1024

A Informação e sua Representação

- **BYTE (BInary TErm)**
  - Grupo ordenado de 8 bits, para efeito de manipulação interna mais eficiente
  - Tratado de forma individual, como unidade de armazenamento e transferência.
  - Unidade de memória usada para representar um caractere.

Com 8 bits, podemos arranjar 256 configurações diferentes: dá para 256 caracteres, ou para números de 0 a 255, ou de –128 a 127, por exemplo.

O termo *bit* apareceu em 1949, inventado por John Tukey, um pioneiro dos computadores. Segundo Tukey, era melhor que as alternativas *bigit* ou *binit*.

O termo *byte* foi criado por Werner Buchholz em 1956 durante o desenho do computador IBM Stretch. Inicialmente era um grupo de 1 a 6 *bits*, mas logo se transformou num de 8 *bits*. A palavra é uma mutação de *bite*, para não confundir com *bit*.

A Informação e sua Representação

- Todas as letras, números e outros caracteres são codificados e decodificados pelos equipamentos através dos bytes que os representam, permitindo, dessa forma, a comunicação entre o usuário e a máquina.

- Sistemas mais importantes desenvolvidos para representar símbolos com números binários (bits):
  - ***EBCDIC*** (*Extended Binary Coded Decimal Interchange Code* – Código Ampliado de Caracteres Decimais Codificados em Binário para o Intercâmbio de Dados).
  - ***ASCII*** (*American Standard Code for Information Interchange* – Código Padrão Americano para o Intercâmbio de Informações).
  - ***UNICODE*** (Unicódigo).

A Informação e sua Representação

- ***EBCDIC***
  - Código de 8 bits (256 símbolos).
  - Usado em *mainframe* IBM e em sistemas de médio porte, raramente encontrado em microcomputadores.

- ***ASCII***
  - Padrão definido pela organização ANSI.
  - Código de 7 bits (128 combinações de caracteres).
  - No PC existe o ASCII Estendido (utiliza outros 128 códigos para símbolos gráficos, e línguas diferentes do inglês).

- ***UNICODE***
  - Novo padrão para representação de dados, oferecerá 2 bytes para a representação de símbolos (mais de 65.000 símbolos)

1 byte = 8 bits = 1 caractere (letra, número ou símbolo)

Podemos definir a **palavra** como um conjunto de bits que representa uma informação útil para os computadores. A palavra nos computadores é um valor fixo e constante para um dado processador (p.ex.: 32 bits, 64 bits).

A Informação e sua Representação

Partes do conjunto de caracteres ASCII

Binário	Caractere
0100 0001	A
0100 0010	B
0110 0001	a
0110 0010	b
0011 1100	<
0011 1101	=
0001 1011	ESC
0111 1111	DEL

Como os principais códigos de representação de caracteres utilizam grupos de 8 bits por caractere, os conceitos byte e caractere tornam-se semelhantes, e as, palavras, quase sinônimas. O termo caractere é mais usado para fins comerciais e o termo byte é empregado mais na linguagem técnica de profissionais da área.

A Informação e sua Representação

Indicações numéricas dos computadores:

Bit - 2 estados: 0 e 1

Byte	b	8 bits	
Quilobyte (ou Kilobyte)	Kb	1.024 bytes	2^{10}=1.024
Megabyte	Mb	1.024 Kb	2^{20}=1.048.576
Gigabyte	Gb	1.024 Mb	2^{30}=1.073.741.824
Terabyte	Tb	1.024 Gb	2^{40}=1.099.511.627.776
Petabyte	Pb	1.024 Tb	2^{40}=1.125.899.906.842.264

Peta (Pb) → Exabyte (Eb) → Zettabyte (Zb) → Yottabyte (Yb)

A Informação e sua Representação

- Os computadores manipulam **dados** (sinais brutos e sem significado individual) para produzir **informações**.

- A conversão de dados em informações, e estas novamente em dados, é uma parte tão fundamental em relação ao que os computadores fazem que é preciso saber como a conversão ocorre para compreender como o computador funciona.

- Infelizmente os computadores não usam nosso sistema de numeração.

Embora os códigos de caracteres sejam úteis para representar dados textuais e números inteiros (0 a 9), eles não são úteis para números que possuem pontos fracionários, como 1,25. Para representar números com frações, bem como números extremamente grandes, por exemplo, os computadores utilizam a **notação de ponto flutuante** (a ser vista posteriormente).

A Informação e sua Representação

Sistema de Numeração

- Conjunto de símbolos utilizados para representação de quantidades e de regras que definem a forma de representação.
- Cada sistema de numeração é apenas um método diferente de representar quantidades. As quantidades em si não mudam; mudam apenas os símbolos usados para representá-las.
- A quantidade de algarismos disponíveis em um dado sistema de numeração é chamada de **base**.
- Representação numérica mais empregada: ***notação posicional***.

A Informação e sua Representação

Notação Posicional

- Valor atribuído a um símbolo *dependente* da posição em que ele se encontra no conjunto de símbolos que representa uma quantidade.
- O valor total do número é a soma dos valores relativos de cada algarismo (decimal).

Sistema de numeração decimal

A Informação e sua Representação

Notação Não Posicional

- Valor atribuído a um símbolo é *inalterável*, independente da posição em que se encontre no conjunto de símbolos que representam uma quantidade.

Sistema de Numeração *Romano*

A Informação e sua Representação

Sistema de Numeração

- Sistema de numeração – **código**
- Operação básica – **contagem**
- Grupo com um determinado número de objetos – **base (raiz)**
- **Sistemas de numeração básicos**:
  - Decimal
  - Binário
  - Octal
  - Hexadecimal

A Informação e sua Representação

Exemplos de Sistemas de Numeração

Sistema	Base	Algarismos
Binário	2	0,1
Ternário	3	0,1,2
Octal	8	0,1,2,3,4,5,6,7
Decimal	10	0,1,2,3,4,5,6,7,8,9
Duodecimal	12	0,1,2,3,4,5,6,7,8,9,A,B
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Como os números representados em base 2 são muito extensos e, portanto, de difícil manipulação visual, costuma-se representar externamente os valores binários em outras bases de valor mais elevado (octal ou hexadecimal). Isso permite maior compactação de algarismos e melhor visualização dos valores.

A Informação e sua Representação

Sistema de Numeração

Padrões de Representação

- Letra após o número para indicar a base;
- Número entre parênteses e a base como um índice do número.

- **Exemplo:**
  - Sistema Decimal – 2763**D** ou $(2763)_{10}$ ou 2763_{10}

A Informação e sua Representação

Sistema Decimal (Base 10)

- Sistema mais utilizado.
- 10 símbolos para representar quantidades.

0 1 2 3 4 5 6 7 8 9

- **Peso** – representar quantidades maiores que a base.
- Peso trouxe: **unidade**, **dezena**, (dez unidades), **centena** (cem unidades), **milhar** (mil unidades), **dezena de milhar**, **centena de milhar**, etc.
- **Exemplo**: 2574 é composto por 4 unidades, 7 dezenas, 5 centenas e 2 milhares, ou 2000 + 500 + 70 + 4 = 2574

A Informação e sua Representação

Sistema Binário (Base 2)

- Utiliza dois símbolos para representar quantidades.

0 e 1

- Segue as regras do sistema decimal - válidos os conceitos de **peso** e **posição**. Posições não têm nome específico.
- Cada algarismo é chamado de **bit**. Exemplo: 101_2
- **Expressão oral** - diferente dos números decimais.
  - Caractere mais à esquerda - *Most-Significative-Bit* - "**MSB**".
  - Caractere mais à direita - *Least-Significative-Bit* - "**LSB**".

A Informação e sua Representação

Sistema Octal (Base 8)

- Utiliza 8 símbolos.

0	1	2	3	4	5	6	7

- Exemplo: 563_8

- **Expressão oral** - similar ao sistema binário.

A Informação e sua Representação

Sistema Hexadecimal (Base 16)

- Possui 16 símbolos (algarismos) para representar qualquer quantidade.

0 1 2 3 4 5 6 7 8 9 A B C D E F

- Uso das letras - **facilidade de manuseio**.
- Exemplo: $5A3_{16}$
- **Expressão oral** - similar ao sistema binário.

A Informação e sua Representação

Ao trabalhar com sistemas de numeração, em qualquer base, deve-se observar o seguinte:

- O número de dígitos usado no sistema é igual à base.
- O maior dígito é sempre menor que a base.
- O dígito mais significativo está à esquerda, e o menos significativo à direita
- Um “vai-um” de uma posição para outra tem um peso igual a uma potência da base.
- Em geral se toma a base decimal como referência.

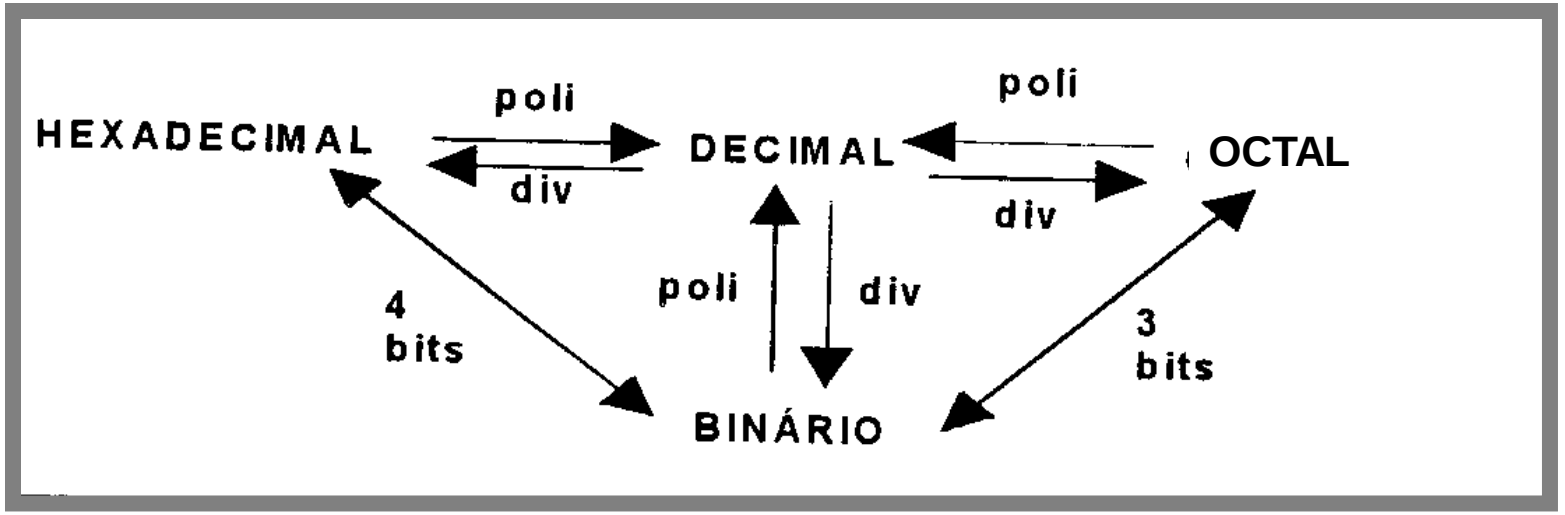
A Informação e sua Representação

Decimal	Binário	Octal	Hexadecimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
. . .	. . .	. . .	. . .

A Informação e sua Representação

Conversão entre Sistemas de Numeração

- **Procedimentos básicos**: **- divisão**
  (números inteiros) **- polinômio**
  - agrupamento de bits

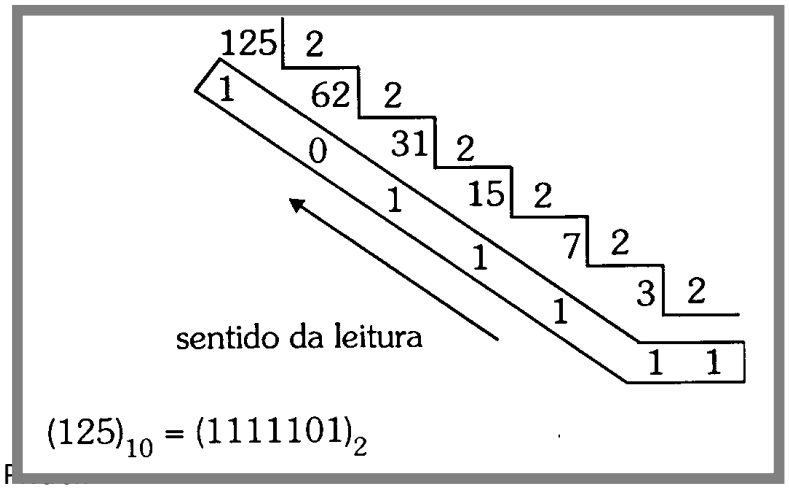


A Informação e sua Representação

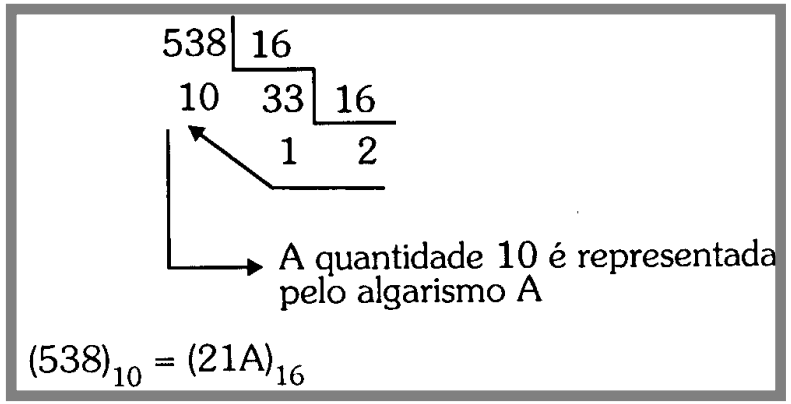
Conversão entre Sistemas de Numeração

- **Divisão** (Decimal → outro sistema)
- Dividir o número por **b** (base do sistema) e os resultados consecutivas vezes.

Ex.: $(125)_{10} = (?)_2$



$(538)_{10} = (?)_{16}$



A Informação e sua Representação

Conversão entre Sistemas de Numeração

Notação Polinomial ou Posicional

- Válida para qualquer base numérica.

- LEI DE FORMAÇÃO
  (**Notação ou Representação Polinomial**):

Número =

$$a_n b^n + a_{n-1} b^{n-1} + a_{n-2} b^{n-2} + \ldots + a_0 b^0$$

a_n = algarismo, b = base do número
n = quantidade de algarismo - 1

A Informação e sua Representação

Conversão entre Sistemas de Numeração

Ex.:

a) $(1111101)_2 = (?)_{10}$

$$(1111101)_2 = 1x2^6 + 1x2^5 + 1x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 = \mathbf{125_{10}}$$

b) $(21A)_{16} = (?)_{10}$

$$(21A)_{16} = 2x16^2 + 1x16^1 + 10x16^0 = \mathbf{538_{10}}$$

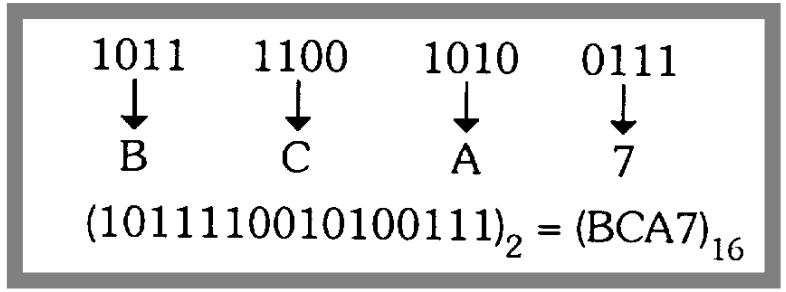
A Informação e sua Representação

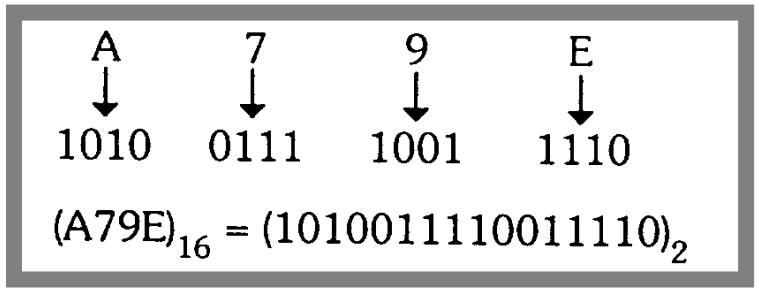
Conversão entre Sistemas de Numeração

Agrupamento de Bits

- Sistemas octal e hexa → binário (e vice versa)
- associando 3 bits ou 4 bits (quando octal ou hexadecimal, respectivamente) e vice-versa.

Ex.: $(1011110010100111)_2 = (\ ?\ )_{16}$ **$(A79E)_{16} = (\ ?\ )_2$**





A Informação e sua Representação

Conversão entre Sistemas de Numeração

Conversão octal → hexadecimal

- Não é realizada diretamente - não há relação de potências entre as bases oito e dezesseis.
- Semelhante à conversão entre duas bases quaisquer - **base intermediária** (base binária)
- Conversão em duas etapas:

1 - número: base octal (hexadecimal) → binária.

2 - resultado intermediário: binária → hexadecimal (octal).

A Informação e sua Representação

Conversão entre Sistemas de Numeração

Ex.:

a) $(175)_8 = (\ ?\ )_{16}$

$(175)_8 = (1111101)_2 = \mathbf{(7D)_{16}}$

b) $(21A)_{16} = (?\ )_8$

$(21A)_{16} = (001000011010)_2 = \mathbf{(1032)_8}$

A Informação e sua Representação

Conversão entre Sistemas de Numeração

Conversão de Números Fracionários

- Lei de Formação ampliada:

$$\text{Número} = \underbrace{a_n.b^n+a_{n-1}.b^{n-1}+a_{n-2}.b^{n-2}+\ldots+a_0.b^0}_{\text{parte inteira}}+\underbrace{a_{-1}.b^{-1}+a_{-2}.b^{-2}+\ldots+a_{-m}.b^{-m}}_{\text{parte fracionária}}$$

A Informação e sua Representação

Conversão entre Sistemas de Numeração

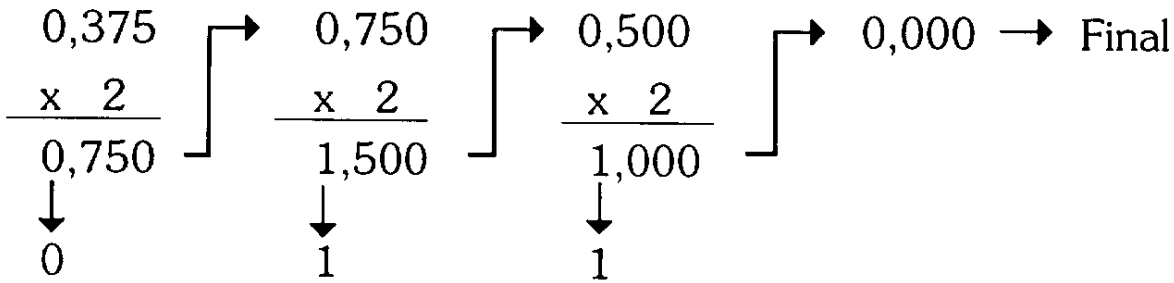
Exemplos:

a) $(101{,}110)_2 = (\ ?\ )_{10}$

$$1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} = (5{,}75)_{10}$$

b) $(8{,}375)_{10} = (\ ?\ )_2$

- parte inteira: $(8)_{10} = (1000)_2$
- parte fracionária:



$(8{,}375)_{10} = (1000{,}011)_2$

A Informação e sua Representação

- **Mostre que:**
  - **$5{,}8_{10} = 101{,}11001100\ldots_{2}$ (uma dízima).**
  - **$11{,}6_{10} = 1011{,}10011001100\ldots_{2}$**
    - a vírgula foi deslocada uma casa para a direita, pois 11,6 = 2 x 5,8 .

A Informação e sua Representação

- Em um computador são armazenados e processados apenas dados e instruções.

- Um computador executa operações sobre dados numéricos (os números) ou alfabéticos (letras e símbolos).

- É preciso definir uma forma de representar os dados, codificados em uns e zeros, que possam ser interpretados pelo computador, de forma correta e eficiente (com bom desempenho e pouco consumo de memória).

A Informação e sua Representação

Os dados podem ser:

- **alfabéticos**
  - letras, números e símbolos (codificados em ASCII e EBCDIC)
- **numéricos**
  - ponto fixo (números inteiros)
  - ponto flutuante (números reais ou fracionários)
  - BCD (representação decimal codificada em binário)
- **Lógicos**
  - Variáveis que possuem apenas dois valores para representação (FALSO e VERDADEIRO).

Representação de Números Inteiros

- Todos os dados numéricos são representados em um computador como uma seqüência de 0s e 1s.
- Os números podem ser positivos ou negativos. As operações aritméticas, em particular a subtração, podem originar resultados negativos.
- Um aspecto primordial a ser definido seria então como representar o sinal.
- **Como é que um computador sabe que um dado número é negativo?**

Representação de Números Inteiros

- A resposta a esta pergunta é que isso depende da convenção usada na representação de números.

- As convenções mais usuais são as seguintes:
  - **Representação de grandeza com sinal (sinal e magnitude)**
  - **Representação em complemento de 2**

Outras formas de representação:

Complemento de 1: para negar o valor de um número deve-se inverter os bits do sinal (obsoleta) e **Excesso de 2^{m-1}**: representação do número é dada pela soma de seu valor absoluto com 2^{m-1}. Exemplo: Um sistema de 8 bits é chamado de excesso de 128 e um número é armazenado com seu valor real somado a 128. Ex.:-3=01111101_2 (-3+128=125)

Representação de Grandeza com Sinal

- O bit mais significativo representa o sinal:
  - **0** (indica um **número positivo**)
  - **1** (indica um **número negativo**)

- Os demais bits representam a **grandeza (magnitude)**.

sinal	magnitude

- O valor dos bits usados para representar a magnitude independe do sinal (sendo o número positivo ou negativo, a representação binária da magnitude será a mesma).

Exemplos: (8 bits)

- $00101001_2 = +41_{10}$
- $10101001_2 = -41_{10}$

Representação de Grandeza com Sinal

- **Exemplos: (8 bits)**

Valor decimal	Valor binário com 8 bits (7 + bit de sinal)
+9	00001001
-9	10001001
+127	01111111
-127	11111111

Assim, uma representação em binário com **n** bits teria disponível para a representação do número ***n-1*** bits (o bit mais significativo representa o sinal).

Representação de Grandeza com Sinal

- Apresenta uma grande **desvantagem**: ela exige um grande número de testes para se realizar uma simples soma de dois números inteiros.

- Requer que na UAL existam dois circuitos distintos para a adição e a subtração.

- Existem **duas representações** para o zero.

Representação de complemento de 2

- **Representação de números inteiros positivos**
  - igual à representação de grandeza com sinal.

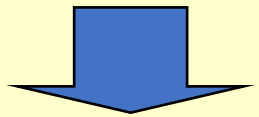
- **Representação de números inteiros negativos**

(a partir da representação em sinal-magnitude)

  - mantém-se o bit de sinal;
  - mantém-se os bits menos significativos da direita para a esquerda até à ocorrência do primeiro bit igual a 1 (inclusive), sendo os bits restantes complementados de 1 (à exceção do bit de sinal).

Exemplo : (8 bits)

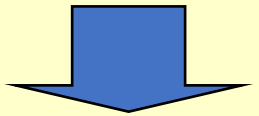
$10001100_2 = -12_{10}$



11110100_{c2}

Exemplo : (8 bits)

$10101001_2 = -41_{10}$



11010111_{c2}

Representação de complemento de 2

- **Exemplo:** Números inteiros codificados em **binário de 8 bits** em um sistema que utiliza complemento de 2:

(-127, ..., -2. -1, 0, +1, +2,..., +127)

{10000001, ..., 11111110, 11111111, 00000000, 00000001, 00000010, ..., 01111111}

- Bit mais significativo ➡ informação de sinal
  (0 = positivo e 1 = negativo)

Representação de complemento de 2

- Requer **um só circuito** (somador) para fazer a adição e a subtração.
- Há apenas uma representação para o valor **0** (disponibilidade para **mais uma representação)** - mais um número negativo pode ser representado (para 8 bits, pode-se representar o número $-128_{10} \Rightarrow 10000000_2$) .
- A quantidade de números positivos é **diferente** da quantidade de números negativos.

Complemento de 2 - PROCESSO

1. **Identifique o bit de sinal**: O bit mais à esquerda (o bit mais significativo) é o bit de sinal. Se for 0, o número é positivo. Se for 1, o número é negativo.

2. **Inverta os bits**: Para converter um número binário em complemento de 1, todos os bits devem ser invertidos. Ou seja, todos os 0s se tornam 1s e todos os 1s se tornam 0s.

3. **Adicione 1 ao número invertido**: Este é o passo final para obter a representação em complemento de 2. Adicione 1 ao número binário obtido após a inversão.

Complemento de 2 - PROCESSO

Por exemplo, vamos converter o número binário 10110 em complemento de 2:

1. Identifique o bit de sinal: O bit de sinal é 1, então o número é negativo.
2. Inverta os bits: 10110 torna-se 01001.
3. Adicione 1 ao número invertido: 01001 + 1 = 01010.

Portanto, 10110 em complemento de 2 é -01010.

Se o número binário for positivo, o processo é o mesmo, exceto que você não precisa inverter os bits, pois o complemento de 1 e o complemento de 2 serão iguais. Por exemplo, para o número binário 01101:

1. Identifique o bit de sinal: O bit de sinal é 0, então o número é positivo.
2. Não há inversão dos bits.
3. Adicione 1 ao número: 01101 + 1 = 01110.

- Portanto, 01101 em complemento de 2 é 01110.

Representação de Números Inteiros

Números negativos de 8 bits expressos em 4 sistemas diferentes

N (decimal)	N (binário)	-N (sinal-magnitude)	-N (comple-mento de 1)	-N (comple-mento de 2)	-N (excesso de 128)
1	00000001	10000001	11111110	11111111	01111111
2	00000010	10000010	11111101	11111110	01111110
3	00000011	10000011	11111100	11111101	01111101
4	00000100	10000100	11111011	11111100	01111100
10	00001010	10001010	11110101	11110110	01110110
20	00010100	10010100	11101011	11101100	01101100
100	01100100	11100100	10011011	10011100	00011100
127	01111111	11111111	10000000	10000001	00000001
128		Não existe represen-tação	Não existe represen-tação	10000000	00000000

Representação de Números Inteiros

- Haverá sempre um padrão de bits a mais ou a menos, não importa qual a representação escolhida.

- O padrão de bits extra pode ser usado como –0, como o menor número negativo da representação, ou algo assim, mas, independentemente de como esse padrão de bits for usado, ele poderá ser um estorvo.

Representação de Números Reais

- Em alguns tipos de cálculo, a faixa de variação dos números envolvidos é muito grande.

- **Exemplo**:
  - 1) Massa do elétron - da ordem de 9×10^{-28} gramas
  - 2) Massa do Sol - aproximadamente igual a 2×10^{33} gramas
  - Faixa de variação: $> 10^{60}$
  - Exemplo de representação (34 dígitos à esquerda do ponto decimal e 28 dígitos à direita do mesmo)

1) 0000000000000000000000000000000000.000000000000000000000000009

2) 2000000000000000000000000000000000.000000000000000000000000000

- **Como representar esses números no computador?**

Representação de Números Reais

- Forma usual de representação de números reais: **parte inteira, vírgula (ou ponto), parte fracionária.**

- Esta representação, embora cômoda para cálculos no papel, não é adequada para processamento no computador.

- **Exemplo: 45,724**

Representação de Números Reais

- O número **45,724** pode ser expresso como:
  - **$45{,}724 \times 10^{0}$**
  - **45724×10^{-3}**
  - **$0{,}45724 \times 10^{2}$**

- É necessário o uso de um sistema de representação de números no qual a faixa de variação dos números seja independente do número de dígitos significativos dos números representados.

Representação de Números Reais

- Uma maneira de separar a faixa de variação dos números de sua precisão consiste em representá-lo na notação científica.

$$n = f \times 10^{e}$$

  - ***f*** - fração ou significando (ou mantissa)
  - ***e*** - expoente (inteiro positivo ou negativo)

- Qualquer número (inteiro ou fracionário) pode ser expresso no formato **número x base^{expoente}**, podendo-se variar a posição da vírgula e o expoente.

- Denominação (computacional): **representação em ponto flutuante** (o ponto varia sua posição, modificando, em conseqüência, o valor representado).

Representação de Números Reais

- Representação pode variar ("**flutuar**") a posição da vírgula, ajustando a potência da base.

- **Exemplos**:
  - $3{,}14 = 0{,}314 \times 10^{-1} = 3{,}14 \times 10^{0}$
  - $0{,}000001 = 0{,}1 \times 10^{-5} = 1{,}0 \times 10^{-6}$
  - $1941 = 0{,}1941 \times 10^{4} = 1{,}941 \times 10^{3}$

- A **faixa de variação** dos números é determinada pela quantidade de dígitos do expoente e a **precisão** é determinada pela quantidade de dígitos do significando.

Representação de Números Reais

- **Forma normalizada:** usa um único dígito antes da vírgula, diferente de zero (*).

- Na representação computacional de números em ponto flutuante, a representação normalizada é, em geral, melhor que a não-normalizada.
  - **Forma normalizada**: só existe uma forma de representar um número.
  - **Forma não normalizada**: um mesmo número pode ser representado de diversas maneiras.

(*) Padrão IEEE 754 para números em ponto flutuante – **significando (mantissa) normalizado** – começa com um bit 1, seguido de um ponto (vírgula) binário e pelo resto do significando (número = ± 1,_ _ ... x 2^{exp})

Mantissa normalizada - começa com o ponto (vírgula) binário seguido por um bit 1 e pelo resto da mantissa (bit antes da vírgula igual a zero).

Representação de Números Reais

Ilustração:

- No sistema binário:
  - **$110101 = 110{,}101 \times 2^3 = 1{,}10101 \times 2^5 = 0{,}0110101 \times 2^7$**

- Números armazenados em um computador - os expoentes serão também gravados na base dois
  - Como $3_{10} = 11_2$ e $7 = 111_2$
  - $110{,}101 \times (10)^{11} = 1{,}10101 \times (10)^{101} = 0{,}0110101 \times (10)^{111}$

- Representação normalizada - há apenas um “1” antes da vírgula
  - **Exemplo**: **$1{,}10101 \times (10)^{101}$**

Representação em Ponto Flutuante

Algumas definições

- No número **1,10101x**$(10)^{\mathbf{101}}$:
  - **1,10101 = significando**
  - **101 = expoente**

- **OBS:**
  - a base binária não precisa ser explicitada
    (o computador usa sempre esta)
  - O “1” antes da vírgula, na representação normalizada – se esta for adotada, também pode ficar implícito, economizando um bit (“**bit escondido**”).

Representação em Ponto Flutuante

- **Na organização/arquitetura do computador, deve-se definir**:
  - Número de bits do significando (precisão, ***p*** ou ***f***)
  - Número de bits do expoente (***e***)
  - Um bit (“**0**” para + e “**1**” para -) de sinal (tipicamente o primeiro, da esquerda)

O Padrão IEEE 754 para Números em Ponto Flutuante

- Até meados dos anos 1980, cada fabricante de computador tinha seu próprio formato para representar números em ponto flutuante.

- **Solução**: criação do **Padrão 754** (IEEE 1985).

- O Padrão IEEE 754 procurou uniformizar a maneira como as diferentes máquinas representam os números em ponto flutuante, bem como devem operá-los.

- O padrão IEEE 754 para ponto (vírgula) flutuante é a representação mais comum para **números reais** em computadores de hoje, incluindo PC's compatíveis com Intel, Macintosh, e a maioria das plataformas Unix/Linux.

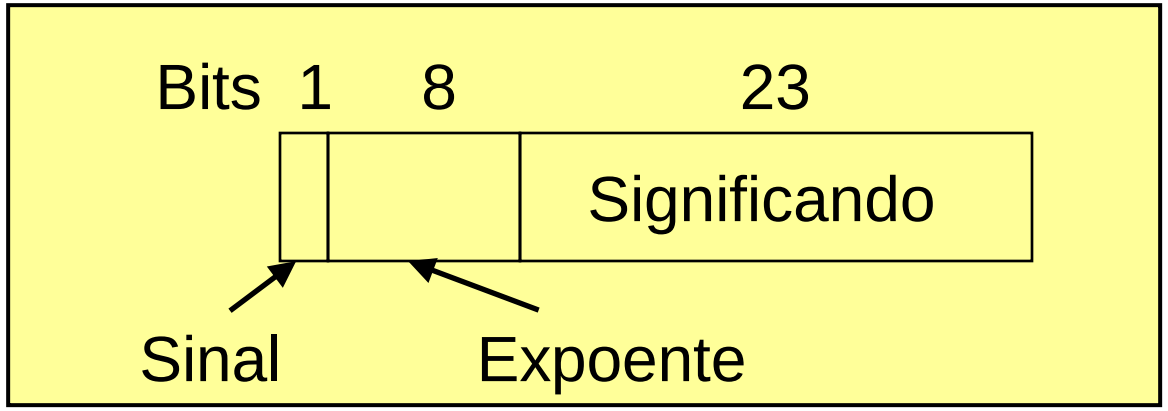
O Padrão IEEE 754 para Números em Ponto Flutuante

O padrão IEEE 754 define três formatos:

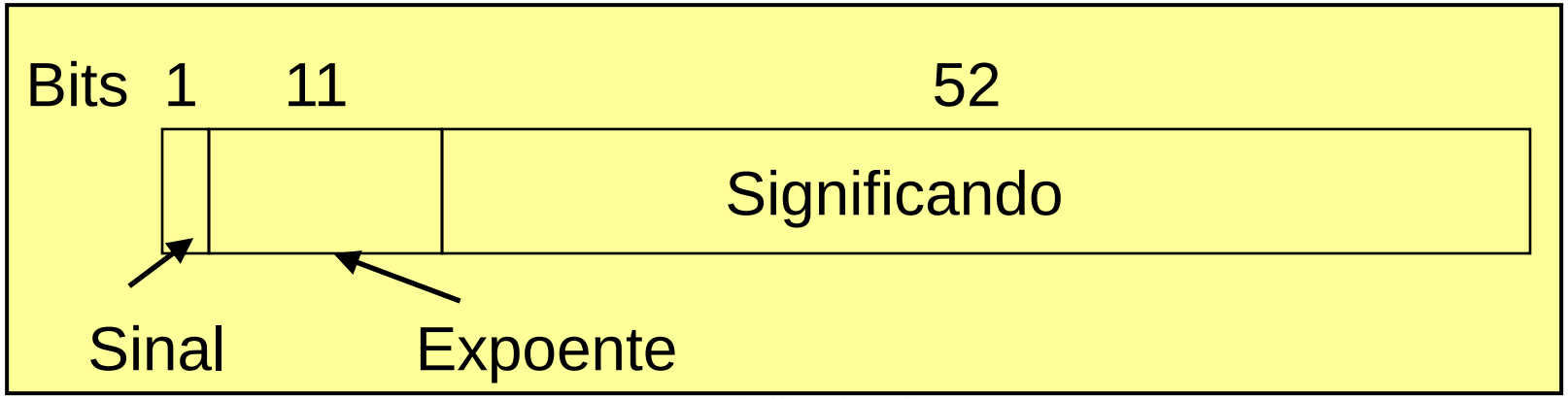
- **Precisão simples** (32 bits)
- **Precisão dupla** (64 bits)
- Precisão estendida (80 bits)

- Os formatos de precisão simples e precisão dupla usam a base 2 para o significando e a notação em excesso para o expoente.

O Padrão IEEE 754 para Números em Ponto Flutuante



Precisão simples



Precisão dupla

O Padrão IEEE 754 para Números em Ponto Flutuante

Precisão	Sinal	Expoente(+/-)	Significando
Simples (32bits)	**1** [bit31]	**8** [bits30-23]	**23** [bits22-00]
Dupla (64 bits)	**1** [bit63]	**11** [bits62-52]	**52** [bits51-00]

- **Sinal**: 0 = + e 1 = -
- **Combinações**: Sinal + Expoente + Significando
- **Notação em excesso de 127** (bit de polarização): precisão simples.
- **Notação em excesso de 1023** (bit de polarização): precisão dupla.

O Padrão IEEE 754 para Números em Ponto Flutuante

Ilustração

- Expoentes na **precisão simples** c/256 combinações
  - **01111111** (127_{10}) = expoente zero (*bias* = polarização)
  - **00000001** = menor expoente = –126 (abaixo de zero)
  - **11111110** = maior expoente = +127 (acima de zero)

O Padrão IEEE 754 para Números em Ponto Flutuante

Exemplo: Realize as conversões abaixo:

- **$10{,}875_{10}$** = **(?)$_2$** (IEEE 754, com 32 bits)
- **$11000001110100000000000000000000_2$** (IEEE 754, com 32 bits) = **(?)$_{10}$**

O Padrão IEEE 754 para Números em Ponto Flutuante

Solução:

- **$10{,}875_{10}$** = $1010{,}111_2$ = $1{,}010111 \times 2^3$

  sinal: 0

  expoente: $3_{10} + 127_{10} = x_{10}$, $x_{10} = 130_{10} = 10000010_2$

  significando: $01011100000000000000000_2$

  Número (IEEE 754, com 32 bits):

  $01000001001011100000000000000000_2$

O Padrão IEEE 754 para Números em Ponto Flutuante

Solução:

- **$11000001110100000000000000000000_2$**

  (IEEE 754, com 32 bits)

  sinal: 1

  expoente: $10000011_2 = 131_{10}$, $x_{10} + 127_{10} = 131_{10}$,

  $x_{10} = 4_{10}$

  significando: $10100000000000000000000_2$ =

  Número:(negativo) $1{,}101_2 \times 2^4 = 11010_2 = \mathbf{-26_{10}}$

Representação de Números Decimais Codificados em Binário (BCD)

- A representação de números reais em ponto flutuante é perfeitamente adequada para fazer cálculos matemáticos, científicos, etc.

- Na representação em ponto flutuante pode-se ter perda de precisão do número representado ou mesmo haverá números que não podem ser representados por *overflow*.

- Para representação de números em que é necessário manter precisão até o último algarismo, não é admissível erro por aproximação.

- **Solução**: usar a representação **BCD** ou *Binary Coded Decimal* (Decimal Representado em Binário).

Representação de Números Decimais Codificados em Binário (BCD)

- A ideia do BCD é representar, em binário, cada algarismo de forma que o número original seja integralmente preservado.

- A codificação BCD não possui extensão fixa, possibilitando representar números com precisão variável - quanto maior o número de bits, maior será a precisão.

Representação de Números Decimais Codificados em Binário (BCD)

Tabela de Representação dos Números Decimais em BCD

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Continua ...

Representação de Números Decimais Codificados em Binário (BCD)

Tabela de Representação dos Números Decimais em BCD

Decimal	BCD
10	Inválido
11	Inválido
12	Inválido
13	Inválido
14	Inválido
15	Inválido

- **Exemplo: 239_{10} = (?) BCD**
  - $2 = 0010_2$
  - $3 = 0011_2$ e
  - $9 = 1001_2$, logo: **239 = 001000111001 (BCD).**

Representação de Números Decimais Codificados em Binário (BCD)

- A codificação de um dígito em BCD requer 4 bits.

- Como a utilização de apenas 4 bits por byte não é eficiente, normalmente são armazenados 2 dígitos BCD em um só byte. Esta representação é chamada **BCD comprimido ou compactado** ("***packed BCD***").

- Exemplo: **14239_{10} = (?) BCD**

1	42	39	número decimal
xxxx0001	**01000010**	**00111001**	representação BCD comprimido
a+2	a+1	a	endereço

Representação de Números Decimais Codificados em Binário (BCD)

- Entre os algarismos sem código válido em decimal (códigos representativos dos valores decimais de 10 a 15), é comum utilizar alguns deles para indicar o sinal do número.

- Há sistemas que adotam a seguinte convenção para o sinal dos números representados em BCD:
  - 1100 ⟶ representa o sinal positivo (“+”)
  - 1101 ⟶ representa o sinal negativo (“-”)

Representação de Números Decimais Codificados em Binário (BCD)

- Com nesta representação ainda há um desperdício de códigos; como BCD usa 4 bits (16 representações possíveis) para representar 10 algarismos, 6 (ou 4) códigos não são utilizados.

- Portanto, essa representação é menos eficiente em relação à utilização dos recursos do computador que a **representação em ponto flutuante**.

Experimentação ativa

Estudo de Caso: **"Gerenciamento Eficiente de Estoque em uma Cadeia de Suprimentos"**

Contexto:

Os alunos são contratados por uma empresa de logística que gerencia o estoque de produtos em uma extensa cadeia de suprimentos. A empresa enfrenta desafios na otimização do espaço de armazenamento e na transmissão eficiente de informações sobre os produtos entre diferentes locais de armazenamento.

Experimentação ativa

Estudo de Caso: **"Gerenciamento Eficiente de Estoque em uma Cadeia de Suprimentos"**

Desafio:

Os alunos precisam desenvolver um sistema de gerenciamento de estoque que otimize a representação e transmissão de dados relacionados aos produtos, levando em consideração diferentes tipos de informações, como códigos de barras, descrições de produtos e níveis de estoque.

Experimentação ativa

Estudo de Caso: **"Gerenciamento Eficiente de Estoque em uma Cadeia de Suprimentos"**

Tarefas:

Codificação de Produtos:

Desenvolva um sistema de codificação para os produtos que inclua códigos de barras e outras representações eficientes.
Considere a representação binária e hexadecimal para os códigos.

Representação de Números Inteiros e Reais:

Explore a representação de números inteiros e reais para quantidades de estoque e preços.
Considere a eficiência na transmissão de dados numéricos.

Codificação de Caracteres Especiais:

Considere a presença de caracteres especiais nas descrições dos produtos.
Avalie diferentes sistemas de codificação, como ASCII e Unicode.

Sistemas Numéricos e Conversões:

Converta as quantidades de estoque entre diferentes sistemas numéricos, como binário, decimal e octal.
Explique como essas conversões podem ser aplicadas na transmissão de dados.

Ponto Flutuante e Números Decimais:

Se os preços dos produtos envolvem valores decimais, explore a representação em ponto flutuante.
Compare a eficiência de representar preços em formato inteiro e ponto flutuante.

Experimentação ativa

Estudo de Caso: **"Gerenciamento Eficiente de Estoque em uma Cadeia de Suprimentos"**

Tarefas:

Codificação de Produtos:

Desenvolva um sistema de codificação para os produtos que inclua códigos de barras e outras representações eficientes.
Considere a representação binária e hexadecimal para os códigos.

Representação de Números Inteiros e Reais:

Explore a representação de números inteiros e reais para quantidades de estoque e preços.
Considere a eficiência na transmissão de dados numéricos.

Codificação de Caracteres Especiais:

Considere a presença de caracteres especiais nas descrições dos produtos.
Avalie diferentes sistemas de codificação, como ASCII e Unicode.

Sistemas Numéricos e Conversões:

Converta as quantidades de estoque entre diferentes sistemas numéricos, como binário, decimal e octal.
Explique como essas conversões podem ser aplicadas na transmissão de dados.

Ponto Flutuante e Números Decimais:

Se os preços dos produtos envolvem valores decimais, explore a representação em ponto flutuante.
Compare a eficiência de representar preços em formato inteiro e ponto flutuante.

Colaboração e envolvimento no Mundo Real

Divida em Grupos os estudantes para colaborarem com suas soluções.

O grupo deve sair com uma solução única e unificada.

Cada integrante do grupo, deve fazer a publicação em seu portfólio da solução a que chegou.