Unichristus
Centro Universitário Christus

INTRODUÇÃO À COMPUTAÇÃO E SISTEMAS DE INFORMAÇÃO

Arquitetura de Computadores
&
Sistemas de Numeração

Lattes - linkedin
euristenhojr@gmail.com
http://www.unichristus.edu.br/

Euristenho Queiroz de Oliveira **Júnior**
Especialista em Engenharia de Software
MSc em Engenharia de Software

AGENDA

1. Apresentação
2. Livros
3. Acordo de Convivência
4. Arquitetura de Computadores
5. Sistemas de Numeração
6. Conhecimento - Competência
7. Próxima Aula
8. Referências

Unichristus
Centro Universitário Christus

Apresentação

FORMAÇÃO ACADÊMICA

- Graduado em Telemática/Telecomunicações - IFCE (2002 - 2008)
- Especialista em Engenharia de Software - FA7 (2011 - 2013)
- MSc em Engenharia de Software - UFPE (2011 - 2015)

CURRÍCULO PROFISSIONAL

- Atuei 4 anos na empresa privada
- 10 anos no ambiente Público
- Atualmente Líder Técnico de 45 Projetos de Tecnologia na SEPOG/PMF

Apresentação

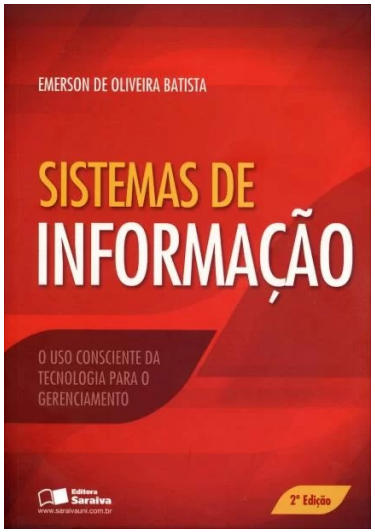
DOCÊNCIA

- Professor Substituto das Disciplinas de Sistemas de Informação – FA7 (2011 - 2012)
- Professor da Especialização em Sistemas WEB – FJN (2011 - 2012)
- Professor de Bancas de graduação em Sistemas de Informações – FA7 (2012)
- Professor dos Cursos de Tecnologia da Unifanor (2015 – 2018)
- Professor do Curso de Tecnologia da Unichristus (2018 - Atual)

Livros

- **Sistemas de Informação Gerenciais** - 11ª Ed. 2014 - Laudon,Kenneth C.; Jane P. Laudon - Pearson

- **Sistemas de Informação - o Uso Consciente da Tecnologia Para o Gerenciamento** - 2ª Ed. 2012 - Batista,Emerson de O.



Ementa

- Fundamentos de computação, arquiteturas de computadores e sistemas operacionais. Ambientes de processamento automatizado de informações.
- Evolução das profissões e características do profissional de sistemas de informação.
- Conceitos básicos: dado, informação e conhecimento.
- Computador e seus elementos básicos.
- Internet e Redes de Computadores – modelos e usos.
- Fundamentos de sistemas de informação.
- Classificação dos sistemas de informação.
- Conceitos e usos da Segurança de dados.
- Etapas da Especificação e do projeto de sistemas de informação.

Dicas de Convivência

- Horários
- Conversas
- Dúvidas
- Celular
- Avaliações

Questionamentos

Arquitetura de Computadores (ou Organização de Computadores) é estrutura como são organizados os módulos funcionais de um computador, como:

1- Processador

2- Memória

3- entrada/saída

4- etc.

Pode também denotar as propriedades lógicas e abstratas dos computadores, em conjunto com os métodos de projeto utilizados para implementar estas características

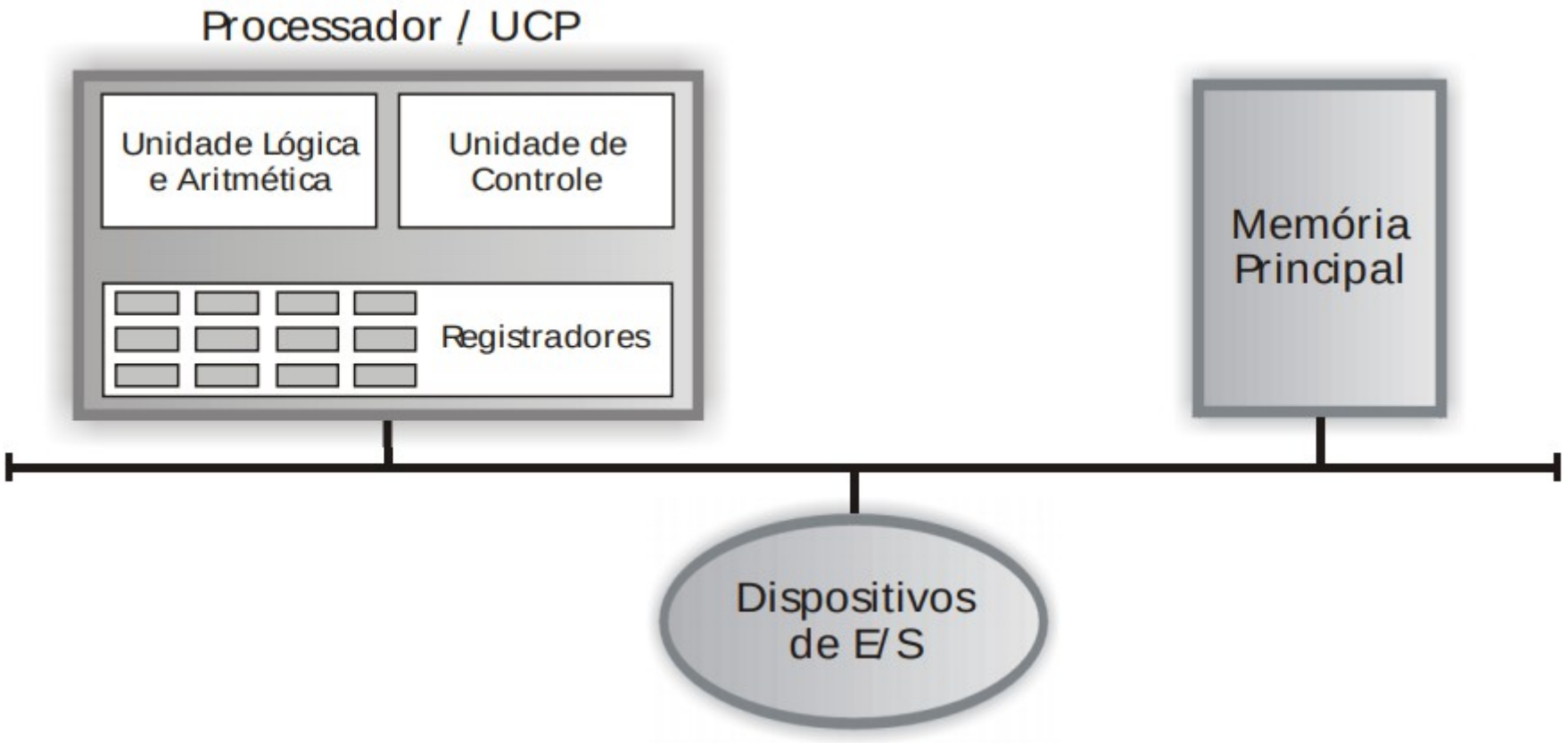
Módulos básicos:

O **Processador**, também conhecido como CPU (Central Processing Unit, ou Unidade Central de Processamento), é responsável pela realização de todo o processamento

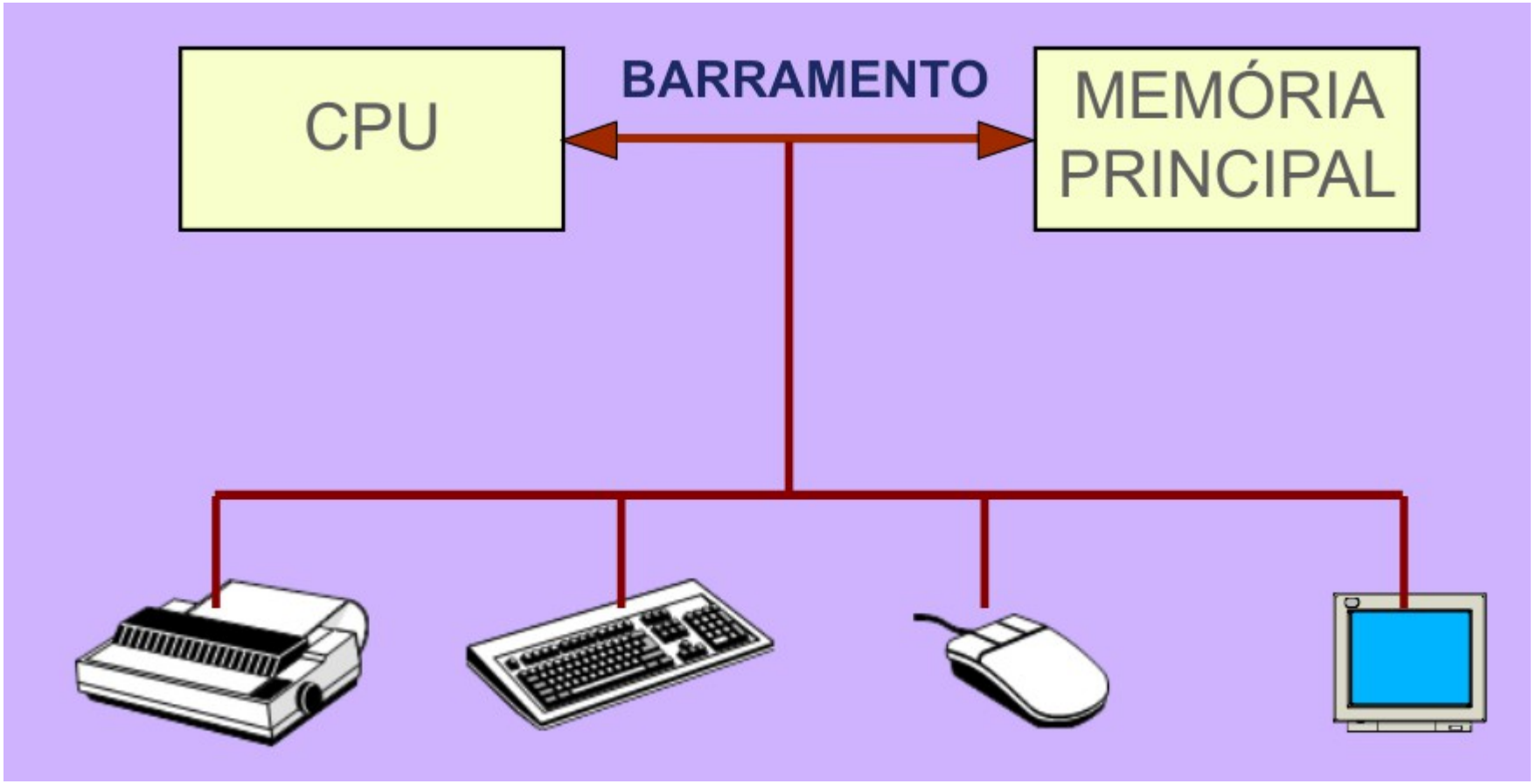
A **Memória** principal, utilizada para armazenar programas e dados.

O **Barramento**, que é o canal de comunicação entre o processador e a memória

Arquitetura Von Neumann



Arquitetura Von Neumann



- UCP – Unidade Central de Processamento
- A CPU é o “cérebro” do computador
- Executa programas armazenados na memória principal, buscando as instruções, examinando-as, e então executando uma após outra
- **ATENÇÃO:** O gabinete do computador, onde ficam localizados, placa-mãe, fonte, discos, além de outros componentes é erroneamente chamado de CPU

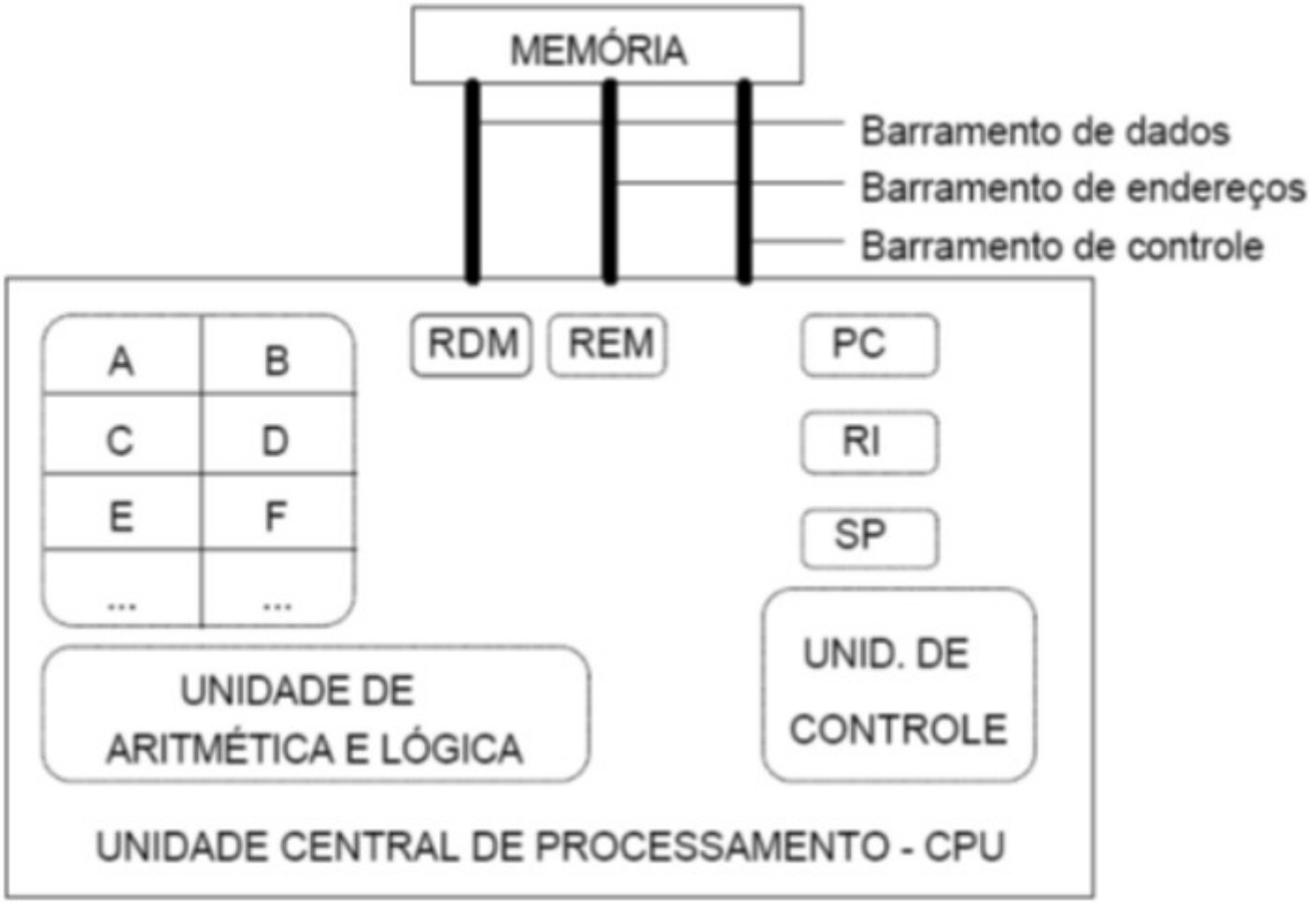
Componentes da CPU

- **Unidade de Controle:** busca as instruções da memória e determina o seu tipo (adição, subtração, comparação, etc.)

- **Unidade Lógica e Aritmética (ULA ou ALU):** executa efetivamente as operações lógicas (comparações) e aritméticas (adição, etc.)

- **Registradores:** formam uma memória pequena, de alta velocidade, usada para armazenar resultados temporários e informações de controle

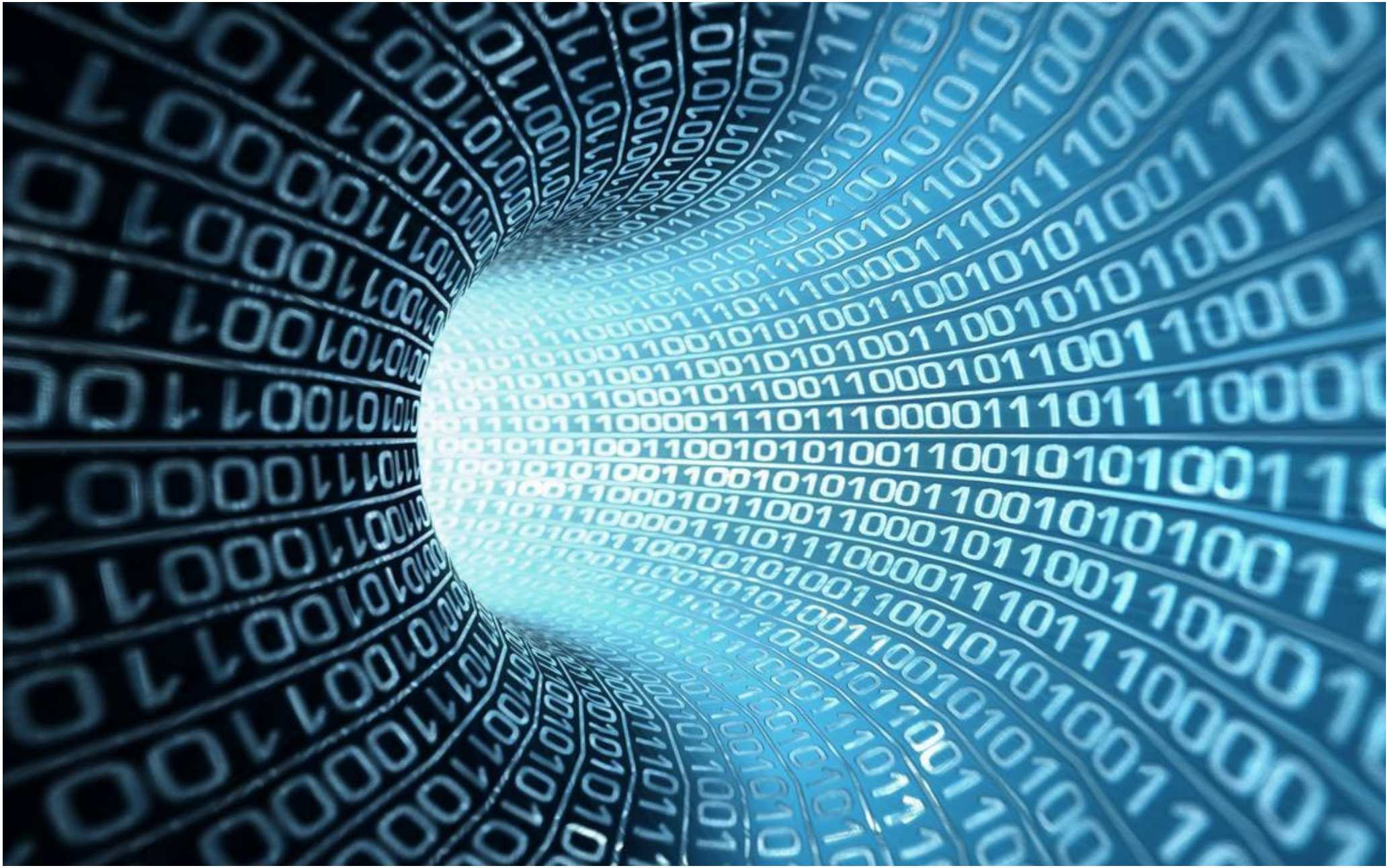
Execução de um programa na CPU



Conjunto de instruções

- A coleção de todas as instruções disponíveis ao programador (a nível de máquina) é chamada de conjunto de instruções daquela CPU
- O número de instruções varia de máquina para máquina, dependo das decisões do seu projeto
- Nem todos os processadores são CPUs
- Alguns processadores possuem conjuntos de instruções com finalidades específicas, como processadores aritméticos ou gráficos

Sistema de Numeração



Unichristus
Centro Universitário Christus

Prof. Euristenho Júnior

Conceito de bit e byte

Um **bit** ou dígito binário (*binary digit*), é a unidade básica que os computadores e sistemas digitais utilizam para trabalhar, ele pode assumir apenas dois valores, **0** ou **1**. Um **byte** é uma sequência de 8 bits.

Fisicamente, um bit pode ser representado de várias formas: através de dois valores de voltagem aplicados num fio, diferentes direções de magnetização em uma fita magnética, entre outras. O importante é que seja possível identificar dois estados diferentes.

O byte é a menor unidade de armazenamento utilizada pelos computadores. Isto quer dizer que, nós nunca conseguiremos salvar menos do que 8 bits.

Na próxima seção iremos estudar como os bits e bytes são utilizados na representação de dados e mídias.

Possibilidades de Representação

Como um bit só pode assumir dois valores (**0** ou **1**), só será possível representar exatamente dois estados distintos. Na Tabela 2.1, “Representações com um bit.” nós temos exemplos de como podemos associar significados aos valores do bit.

Por exemplo, em um sistema com trava eletrônica, o valor **0** poderia indicar que a porta estava fechada, enquanto **1** indicaria que a porta está aberta. Em outro sistema que registra o estado civil, **0** poderia representar *Solteiro*, enquanto **1** seria *Casado*.

Tabela 2.1. Representações com um bit.

Bit	Porta	Lâmpada	Sexo	Detector de movimento	Estado civil
0	Fechada	Desligada	Masculino	Sem movimento	Solteiro
1	Aberta	Ligada	Feminino	Com movimente	Casado

Possibilidades de Representação

Para representar mais de dois valores distintos nós precisamos de uma sequência de bits maior. Na Tabela 2.2, “Representações com dois bits.” nós temos exemplos de representações utilizando sequências com **2** bits, obtendo **4** possibilidades. Nesta caso, o estado civil *Casado* passou a ser representado pela sequência `01`.

Tabela 2.2. Representações com dois bits.

Sequencia de Bits	Lâmpada	Estado civil
00	Desligada	Solteiro
01	Ligada com intensidade **baixa**	Casado
10	Ligada com intensidade **alta**	Divorciado
11	*Não utilizado*	Viúvo

Possibilidades de Representação

Observe que o número de possibilidades diferentes que podemos representar depende do tamanho da sequência de bits que estamos utilizando, mais precisamente: $2^{tamanho}$.

$2^1 = 2$	$2^2 = 4$	$2^3 = 8$	$2^4 = 16$
$2^5 = 32$	$2^6 = 64$	$2^7 = 128$	$2^8 = 256$ possibilidades (um byte)
16 bits = 65.535	32 bits = 4.294.967.295		64 bits = 18.446.744.073.709.551.615

As tabelas são bastante utilizadas para representar informações. Em uma coluna colocamos o que desejamos representar e na outra sua representação binária. Não há uma ordem particular, por exemplo, na Tabela 2.2, "Representações com dois bits." *Solteiro* era representado por `00`, mas poderíamos construir outra tabela em que seria **codificado** como `11` (lê-se um-um).

O termo **codificar** significa traduzir um conteúdo para a sua representação binária.

Percebam também que, quando todas as informações desejadas já foram representadas, podem existir sequências binárias que não possuem significado definido, que foi o caso da sequência `11` para a lâmpada (Tabela 2.2, "Representações com dois bits.").

Possibilidades de Representação

Tabela 2.3. Representação de números utilizando um byte.

Num	Byte	Num	Byte	Num	Byte	Num	Byte	Num	Byte
0	00000000	8	00001000	16	00010000	24	00011000	248	11111000
1	00000001	9	00001001	17	00010001	25	00011001	249	11111001
2	00000010	10	00001010	18	00010010	26	00011010	250	11111010
3	00000011	11	00001011	19	00010011	27	00011011	251	11111011
4	00000100	12	00001100	20	00010100	28	00011100	252	11111100
5	00000101	13	00001101	21	00010101	29	00011101	253	11111101
6	00000110	14	00001110	22	00010110	30	00011110	254	11111110
7	00000111	15	00001111	23	00010111	31	00011111	255	11111111

Possibilidades de Representação

Caractere	Byte	Caractere	Byte	Caractere	Byte	Caractere	Byte	Caractere	Byte
a	01100001	A	01000001	n	01101110	N	01001110	0	00110000
b	01100010	B	01000010	o	01101111	O	01001111	1	00110001
c	01100011	C	01000011	p	01110000	P	01010000	2	00110010
d	01100100	D	01000100	q	01110001	Q	01010001	3	00110011
e	01100101	E	01000101	r	01110010	R	01010010	4	00110100
f	01100110	F	01100110	s	01110011	S	01010011	5	00110101
g	01100111	G	01100111	t	01110100	T	01010100	6	00110110
h	01101000	H	01101000	u	01110101	U	01010101	7	00110111
i	01101001	I	01101001	v	01110110	V	01010110	8	00111000
j	01101010	J	01101010	w	01110111	W	01010111	9	00111001
k	01101011	K	01101011	x	01111000	X	01011000		
l	01101100	L	01001100	y	01111001	Y	01011001		
m	01101101	M	01001101	z	01111010	Z	01011010		

Possibilidades de Representação

Percebam que neste sistema os caracteres são representados por exatamente um byte, que é o tamanho mínimo possível de ser salvo no computador.

Talvez você tenha percebido a ausência dos *caracteres especiais*, como o "ç", "ß", além dos caracteres acentuados como "ã", "ô", "é", etc. Isto porque o padrão ASCII foi criado por americanos para **codificar** as mensagens escritas no idioma inglês, que não possuem tais caracteres. Por esta razão, existem vários outros sistemas de codificação para melhor representar as mensagens do idioma que se deseja utilizar, alguns exemplos são: **Unicode**, **UTF-8** e **ISO 8859-1** (padrão latino-americano).

Faça um teste! Abra um editor de texto como o *bloco de notas*, *gedit* ou *kate* (não use o *Word*). Digite *abc* no documento em branco e salve-o. Em seguida, verifique o tamanho do arquivo, dependendo da codificação utilizada pelo seu editor o arquivo poderá ter de 3 a 8 bytes.

Imagens

Uma das formas possíveis para representar imagens é tratá-las como grades de pontos (ou *pixels*).

Ao atribuir uma cor para cada ponto, podemos então pintar a imagem. Na Figura 2.1, “Fotografia evidenciando a grade de pontos.” nós temos uma imagem e um recorte em destaque, mostrando a grade de pontos com suas respectivas cores.

Além das cores dos pontos também é necessário definir o tamanho da grade (quantos pontos teremos na horizontal e na vertical), também conhecida como *resolução da imagem*. Sem a resolução teríamos apenas um linha de pontos coloridos.

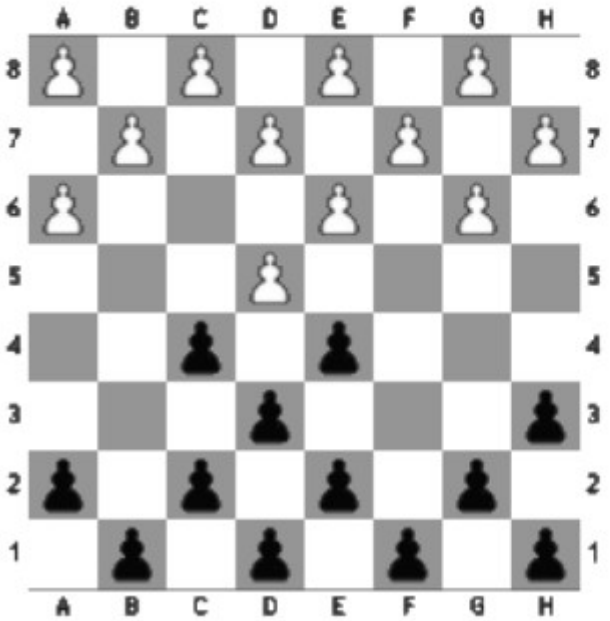
Um sistema popular de representação de cores é o **RGB**, onde é reservado um byte para os tons de cada uma das cores primárias: vermelho, verde e azul. Como um byte permite representar 256 tons de uma cor, ao total são possíveis representar 16 milhões (256x256x256) de cores.

Através do sistema **RGB** podemos representar as três cores primárias e as suas derivadas, que são as cores resultantes das misturas das cores primárias. Neste sistema, o branco é interpretado como sendo a união de todas as cores, e o preto a ausência de cor. Este sistema é utilizado pelo formato de imagem **BMP**.

O sistema **RGBA** inclui também um *canal alpha*, responsável por representar a transparência do ponto, utilizado pelo formato de imagem **PNG**.

Atividades

1. Utilizando o método de representação de números estudado (Seção 2.2.1, “Números”), represente os números **32**, **64** e **128** em bytes.
2. Como você representaria todos os meses do ano em bits?
3. Quantos bits são necessários para representar 150 possibilidades diferentes?
4. ★ Como você representaria as horas em bits?
5. ★ Como você representaria o tabuleiro abaixo em binário? Quantos bytes sua estratégia utiliza?



Sistema de Numeração Posicional

O método de numeração de quantidades ao qual estamos acostumados, utiliza um sistema de numeração posicional. Isto significa que a posição ocupada por cada algarismo em um número altera seu valor de uma potência de 10 (na base 10) para cada casa à esquerda.

Por exemplo:

No sistema decimal (base 10), no número 125 o algarismo 1 representa 100 (uma centena ou 10^2), o 2 representa 20 (duas dezenas ou 2 x 10^1), o 5 representa 5 mesmo (5 unidades ou 5x10^0).

Assim, em nossa notação:

$$125 = 1\times10^2 + 2\times10^1 + 5\times10^0$$

Sistema de Numeração Posicional

A base de um sistema é a quantidade de algarismos disponíveis na representação. A base 10 é hoje a mais usualmente empregada, embora não seja a única utilizada. No comércio pedimos uma dúzia de rosas (base 12) e marcamos o tempo em minutos e segundos (base 60).

Quando lidamos com computadores, é muito comum e conveniente o uso de outras bases, as mais importantes são a binária (base 2), octal (base 8) e hexadecimal (base 16).

Um sistema numérico de base k precisa de k símbolos diferentes para representar seus dígitos de 0 a k-1. Os números decimais são formados a partir de 10 dígitos decimais:

```
0 1 2 3 4 5 6 7 8 9
```

Já os números na base binária são representados a partir de dois dígitos:

```
0 1
```

O octal necessita de oito:

```
0 1 2 3 4 5 6 7
```

Sistema de Numeração Posicional

No caso de números hexadecimais, são necessários 16 algarismos. Portanto, serão mais 6 símbolos além dos dez algarismos arábicos. Em geral, usam-se as letras maiúsculas de A a F:

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

A representação 318_{10} (base 10), significa:

$$318_{10} = 3\times10^2 + 1\times10^1 + 8\times10^0$$

Generalizando, representamos uma quantidade N qualquer, numa dada base b, com um número tal como segue:

$$N_b = a_0 \times b^n + a_1 \times b^{n-1} + \ldots + a_n \times b^0$$

Sistema de Numeração Posicional

Abaixo, o número 35 será expresso nas bases elencadas acima:

Decimal

$$35 = 3\times10^{1} + 5\times10^{0} = 30 + 5 = 35_{10}$$

Binário

$$35 = 1\times2^{5} + 0\times2^{4} + 0\times2^{3} + 0\times2^{2} + 1\times2^{1} + 1\times2^{0} = 32 + 0 + 0 + 0 + 2 + 1 = 100011_{2}$$

Octal

$$35 = 4\times8^{1} + 3\times8^{0} = 32 + 3 = 43_{8}$$

Hexadecimal

$$35 = 2\times16^{1} + 3\times16^{0} = 32 + 3 = 23_{16}$$

Conversões entre as bases 2, 8 e 16

As conversões mais simples são as que envolvem bases que são potências entre si. Vamos exemplificar com a conversão entre a base 2 e a base 8. Como $2_3 = 8$, então a conversão funciona da seguinte forma: separando os algarismos de um número binário (base 2) em grupos de três algarismos (começando sempre da direita para a esquerda) e convertendo cada grupo de três algarismos para seu equivalente em octal, teremos a representação do número em octal.

Por exemplo:

$10101001_2 = 010 \,.\, 101 \,.\, 001_2$

Tabela 3.1. Conversão direta de binário para octal e vice-versa

Binário	Octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Logo:

$010_2 = 2_8$	$101_2 = 5_8$	$001_2 = 1_8$	$10101001_2 = 251_8$

Conversões entre as bases 2, 8 e 16

Vamos agora exemplificar com uma conversão entre as bases 2 e 16. Como $2^4 = 16$, seguindo o processo anterior, basta separarmos em grupos de quatro algarismos e converter cada grupo seguindo a Tabela 3.2, "Conversão direta de binário para hexadecimal e vice-versa.".

Por exemplo:

$11010101101_2 = 0110 \,.\, 1010 \,.\, 1101_2$

Olhando a tabela de conversão direta temos:

Tabela 3.2. Conversão direta de binário para hexadecimal e vice-versa.

Binário	Hexadecimal	Binário	Hexadecimal
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

Conversões entre as bases 2, 8 e 16

Logo:

$0110_2 = 6_{16}$	$1010_2 = A_{16}$	$1101_2 = D_{16}$	$11010101101_2 = 6AD_{16}$

Conversões entre as bases 2, 8 e 16

Vamos agora analisar a conversão inversa.

Por exemplo:

$A81_{16} = A \,.\, 8 \,.\, 1_{16}$

Sabemos que:

$A_{16} = 1010_2$	$8_{16} = 1000_2$	$1_{16} = 0001_2$

Portanto:

$A81_{16} = 101010000001_2$

Conversão de números em uma base b qualquer para a base 10

Vamos lembrar a expressão geral já apresentada:

$$N_b = a_0 \times b^n + a_1 \times b^{n-1} + \ldots + a_n \times b^0$$

A melhor forma de fazer a conversão é usando essa expressão. Como exemplo, o número 101101_2 terá calculado seu valor na base 10:

$$101101_2 = 1\times2^5 + 0\times2^4 + 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 = 45_{10}$$

Outros exemplos:

Converter $A5_{16}$ para a base 10:

$$A5_{16} = 10\times16^1 + 5\times16^0 = 160 + 5 = 165_{10}$$

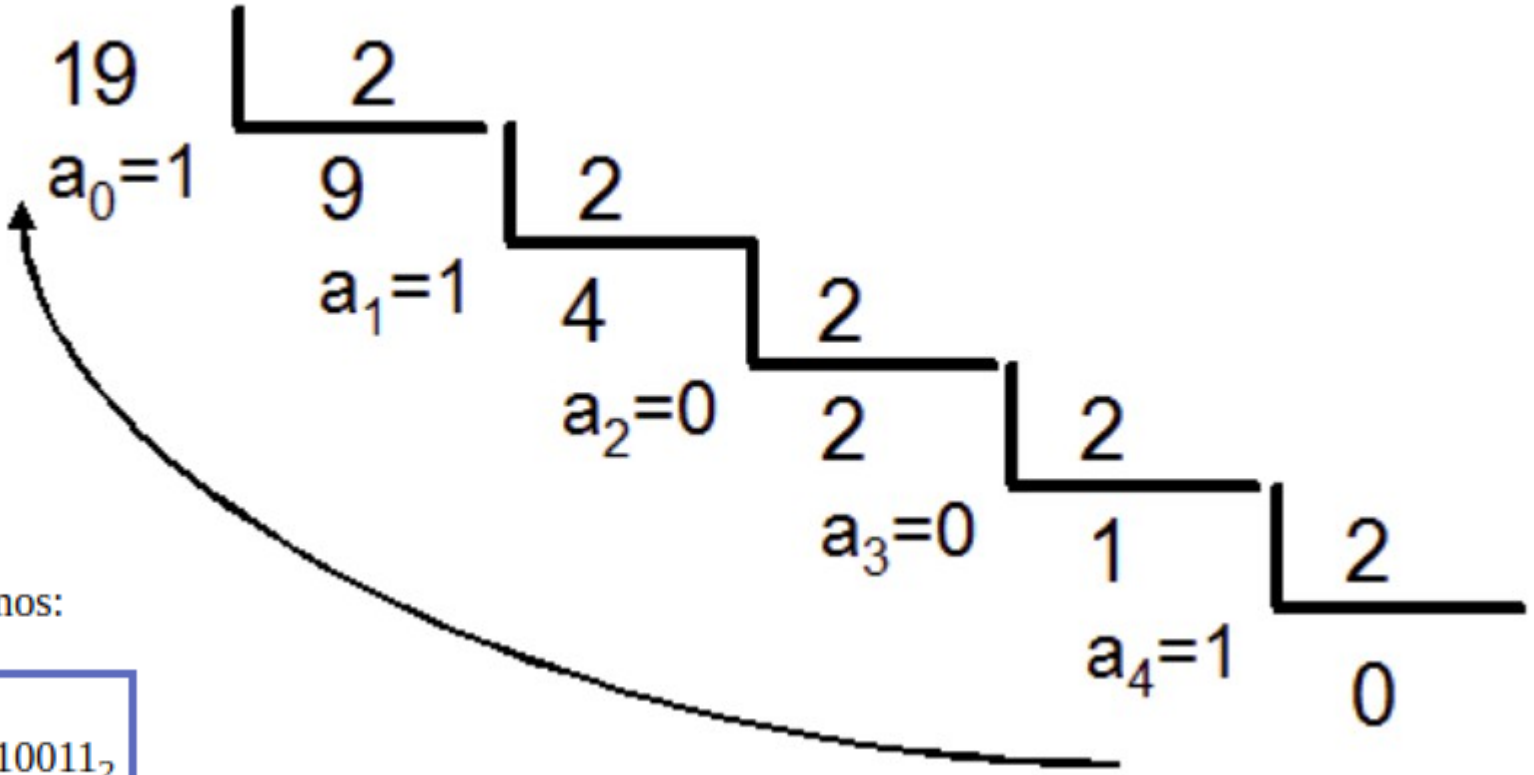
Converter 485_9 para a base 10:

$$485_9 = 4\times9^2 + 8\times9^1 + 5\times9^0 = 324 + 72 + 5 = 401_{10}$$

Conversão de números da base 10 para uma base b qualquer

A conversão de números da base 10 para uma base qualquer, emprega algoritmos que serão o inverso dos anteriores. O número decimal será dividido sucessivas vezes pela base, o resto de cada divisão ocupará sucessivamente as posições de ordem 0, 1, 2 e assim por diante, até que o resto da última divisão (que resulta em quociente 0) ocupe a posição de mais alta ordem.

- Conversão do número 19_{10} para a base 2:



Logo temos:

$19_{10} = 10011_2$

Conversão de números da base 10 para uma base b qualquer

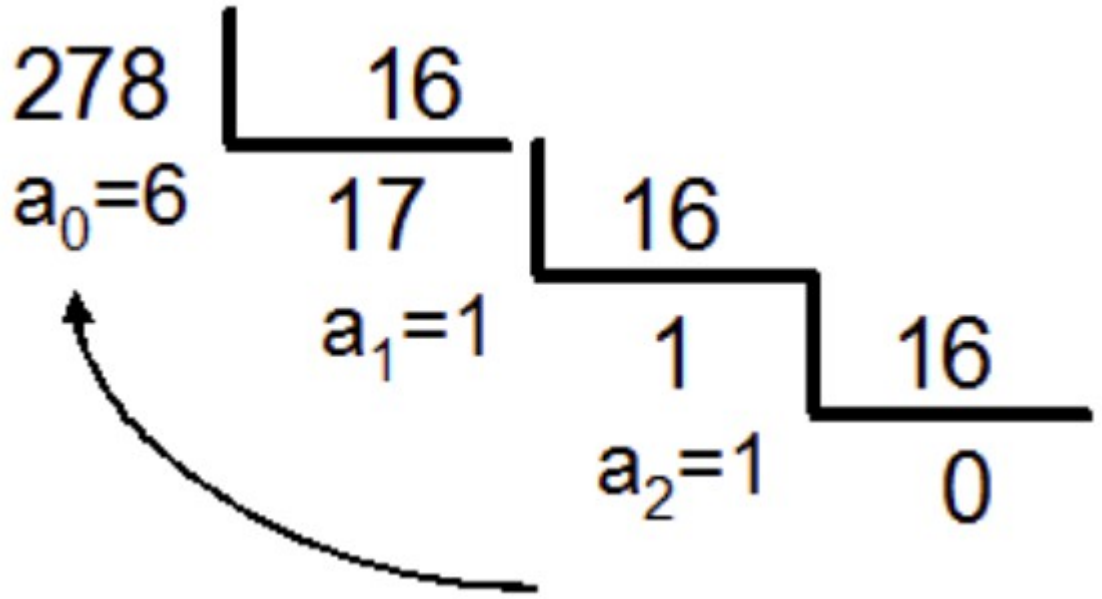
Usando a conversão anterior como prova real, temos:

$$10011_2 = 1\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 = 19_{10}$$

Conversão do número 278_{10} para a base 16?

Conversão de números da base 10 para uma base b qualquer

Conversão do número 278_{10} para a base 16?



Logo temos:

$$278_{10} = 116278_{16}$$

Como o computador manipula os dados (números) através de uma representação binária, iremos estudar agora a aritmética do sistema binário, a mesma usada pela ULA (Unidade Lógica e Aritmética) dos processadores.

Soma e Subtração Binária

A tabuada da soma aritmética em:

```
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 (e “vai um” para o dígito de ordem superior)
1 + 1 + 1 = 1 (e “vai um” para o dígito de ordem superior)
```

Por exemplo:

Efetuar $011100_2 + 011010_2$

Soma-se as posições da direita para esquerda, tal como uma soma decimal.

Solução:

$$\begin{array}{r} 0^{1}\ 1^{1}\ 1\ 1\ 0\ 0 \\ +\ 0\ 1\ 1\ 0\ 1\ 0 \\ \hline 1\ 1\ 0\ 1\ 1\ 0 \end{array} \quad \leftarrow \text{“vai um”}$$

A tabuada da subtração aritmética binária:

```
0 - 0 = 0
0 - 1 = 1 (“vem um do próximo”)
1 - 0 = 1
1 - 1 = 0
```

Como é impossível tirar 1 de 0, o artifício é "pedir emprestado" 1 da casa de ordem superior, ou seja, na realidade o que se faz é subtrair 1_2 de 10_2 e encontramos 1_2 como resultado, devendo então subtrair 1 do dígito de ordem superior. Este algoritmo é exatamente o mesmo da subtração em decimal.

Por exemplo: $111100_2 - 011010_2 = ?$

não esqueça, subtrai-se as colunas da direita para a esquerda, tal como uma subtração decimal.

Solução:

$$\begin{array}{r} 1\ 1^{0}1^{2}0\ 0 \\ -\ 0\ 1\ 0\ 1\ 0 \\ \hline 1\ 0\ 0\ 1\ 0 \end{array}$$

Multiplicação Binária

Vamos ver agora a tabuada da multiplicação:

```
0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1
```

O processo é idêntico à multiplicação entre números decimais.

Multiplicação Binária

Exemplo:

Efetuar: 101_2 x 110_2

Solução:

$$
\begin{array}{rcl}
1\ 0\ 1_2 & \rightarrow & 5_{10} \\
\times\ 1\ 1\ 0_2 & \rightarrow & 6_{10} \\
\hline
0\ 0\ 0 & & \\
1\ 0\ 1 & & \\
1\ 0\ 1 & & \\
\hline
1\ 1\ 1\ 1\ 0_2 & \rightarrow & 30_{10}
\end{array}
$$

Multiplicação Binária

No entanto, a multiplicação em computadores é feita, também, por um artifício: para multiplicar A por n somamos A com A (n-1) vezes.

Exemplo:

$4 \times 3 = 4 + 4 + 4 = 12$

E a divisão também pode ser feita por subtrações sucessivas, até o resultado zerar ou ficar negativo. Este artifício serve apenas para divisões inteiras.

Por exemplo:

$16 \div 4 \rightarrow 16 - \mathbf{4} = 12 \rightarrow 12 - \mathbf{4} = 8 \rightarrow 8 - \mathbf{4} = 4 \rightarrow 4 - \mathbf{4} = 0$

O número de subtrações indica o resultado da divisão inteira, neste caso, igual a 4.

Multiplicação Binária

$27 \times 2 = ?$

Multiplicação Binária

27 x 2 = ?

```
      1  1  0  1  1   (27)
            x 1  0   (2)
   ---------------------
      0  0  0  0  0
+  1  1  0  1  1
   ---------------------
1  1  0  1  1  0   (54)
```

Divisão Binária

- Não há tabela de referência;
  - a operação é feita de modo semelhante à divisão em decimais;
  - o valor do **divisor** deve ser igual ou menor que o do **dividendo** e, se for igual ou menor é escrito 1 no quociente. Esse valor é multiplicado pelo divisor e subtraído do dividendo, até atingir o valor zero, no caso da divisão exata.

Divisão Binária

- Exemplo 1:
  - 55/5 = 11
  - para confirmar faça a multiplicação do divisor pelo quociente.

```
110111 | 101
101    | 1011
-------
00111
  101
-----
 0101
  101
-----
 0000
```

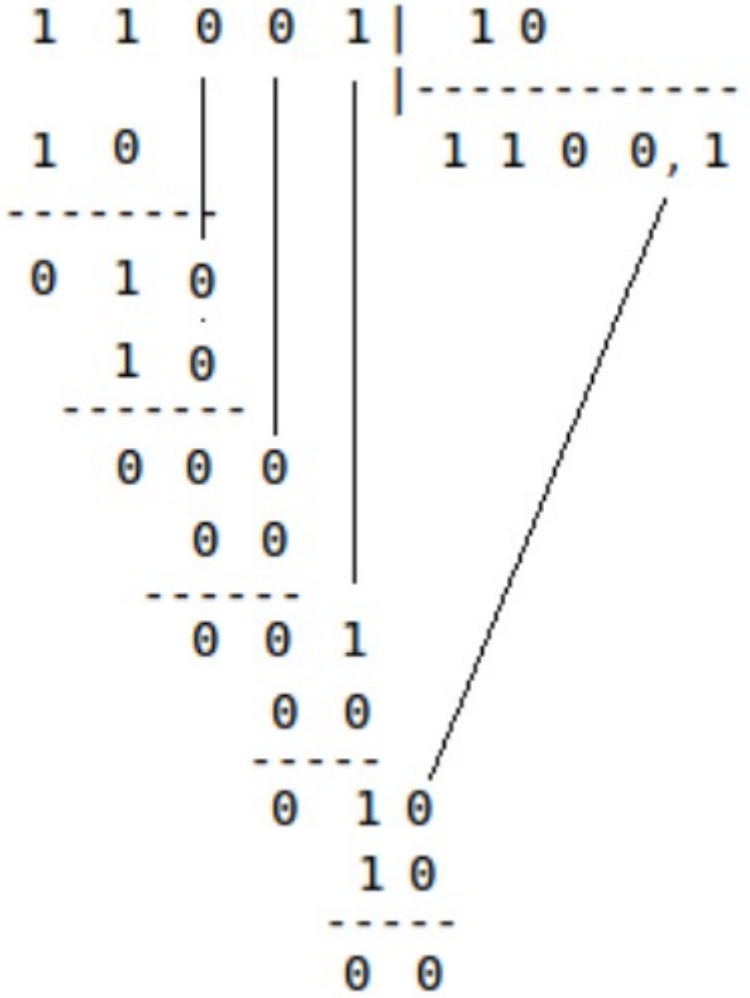
Divisão Binária

- Exemplo 2:
  - 27/3 = 9
  - para confirmar faça a multiplicação do divisor pelo quociente.

```
1  1  0  1  1  |  1  1
                |--------------
1  1              1  0  0  1
--------
0  0  0
   0  0
  --------
   0  0  1
      0  0
     ---------
      0  1  1
         1  1
        ------
         0  0
```

Divisão Binária

- Exemplo 3:
  - divisão não exata;
  - 25/2 = 12,5
  - para confirmar faça a multiplicação do divisor pelo quociente.



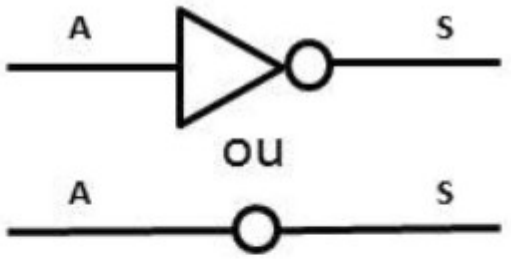
Definição

George Boole publicou a álgebra booleana (em 1854), sendo um sistema completo que permitia a construção de modelos matemáticos para o processamento computacional. O fascinante na lógica booleana é partir de três operadores básicos, que veremos a seguir, e construir Circuitos Lógicos capazes de realizar as diversas operações necessárias para um computador.

Operador NOT

O operador unário NOT, negação binária, resulta no complemento do operando, ou seja, será um bit 1 se o operando for 0, e será 0 caso contrário, conforme podemos confirmar pela tabela de verdade, onde A é o bit de entrada e S é a resposta, ou bit de saída.

NOT



A	S
0	1
1	0

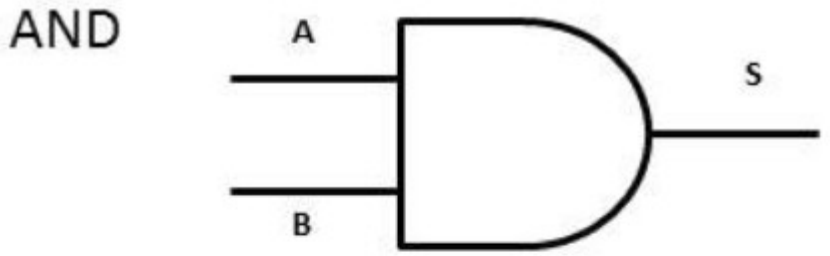
Operador NOT

O Céu é Azul — Verdade !

NOT O Céu é Azul — Falso!

Operador AND

O operador binário AND, ou conjunção binária devolve um bit 1 sempre que ambos operandos sejam 1, conforme podemos confirmar pela tabela de verdade, onde A e B são bits de entrada e S é o bit-resposta, ou bit de saída.



A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

Operador AND

O Céu é Azul. Verdade! | 1

O gelo é quente. Falso! | 0

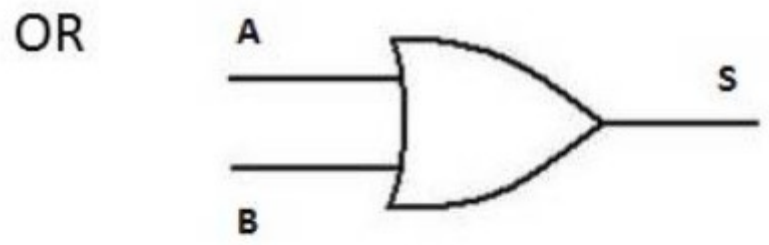
O Céu é Azul

AND Falso! | 0

O gelo é quente.

Operador OR

O operador binário OR, ou disjunção binária devolve um bit 1 sempre que pelo menos um dos operandos seja 1, conforme podemos confirmar pela tabela de verdade, onde A e B são os bits de entrada e S é o bit-resposta, ou bit de saída.



A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

Operador OR

O Céu é Azul. Verdade! | 1

O gelo é quente. Falso! | 0

O Céu é Azul

OR

O gelo é quente. Verdade! | 1

A soma de um Computador

Aprendemos como funciona a aritmética binária, soma, subtração, representação negativa dos números, entre outros. Em seguida, foi apresentada a lógica binária e seus três comandos básicos (AND, OR, NOT). Mas como um computador soma de fato?

Primeiramente, precisamos abordar as portas lógicas, elas são a base para as outras operações. A construção de uma porta lógica, utiliza conhecimentos de circuitos eletrônicos formados por diodos, resistências, capacitores entre outros que são abordados em cursos avançados da Eletrônica Analógica

Circuito somador de 2 bits

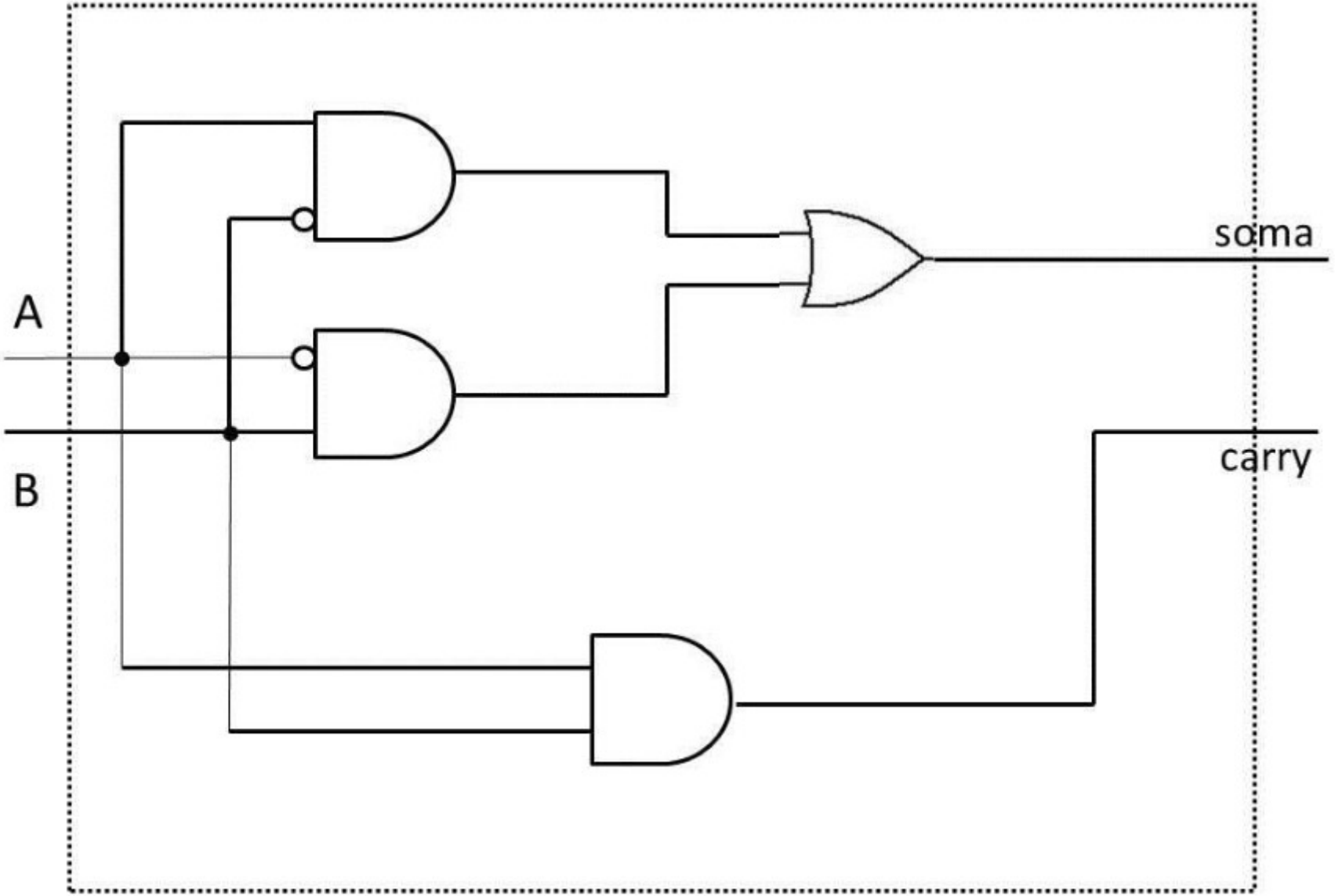


Tabela de Valores da operação de soma de 2 bits

A	B	soma	carry (vai um)
0	0	0	0
0	1	1	0
1	0	1	0
1	**1**	**0**	**1**

Atividades

1- Efetue as seguintes conversões:

a) Converta para decimal 110101_2 e 1001_2

b) Converta para octal 110111011101_2 e 1111111_2

c) Converta para hexadecimal 101100101100_2

d) Converta para binário $FF1F_{16}$ e ABC_{16}

Referências

- TANENBAUM, ANDREW S. Organização Estruturada de Computadores. 5 ta. Edição. Editora Pearson. 2011.
- STALLINGS WILLIAM. Estrutura e Organização de Computadores. 5 ta. Edição. Editora Pearson. 2004.
- BROOKSHEAR GLENN J. Ciência da Computação – Uma visão abrangente. 5 ta. Edição. Editora Bookman. 2000.