WEBCONFERÊNCIA I e II

ser educacional
gente criando o futuro

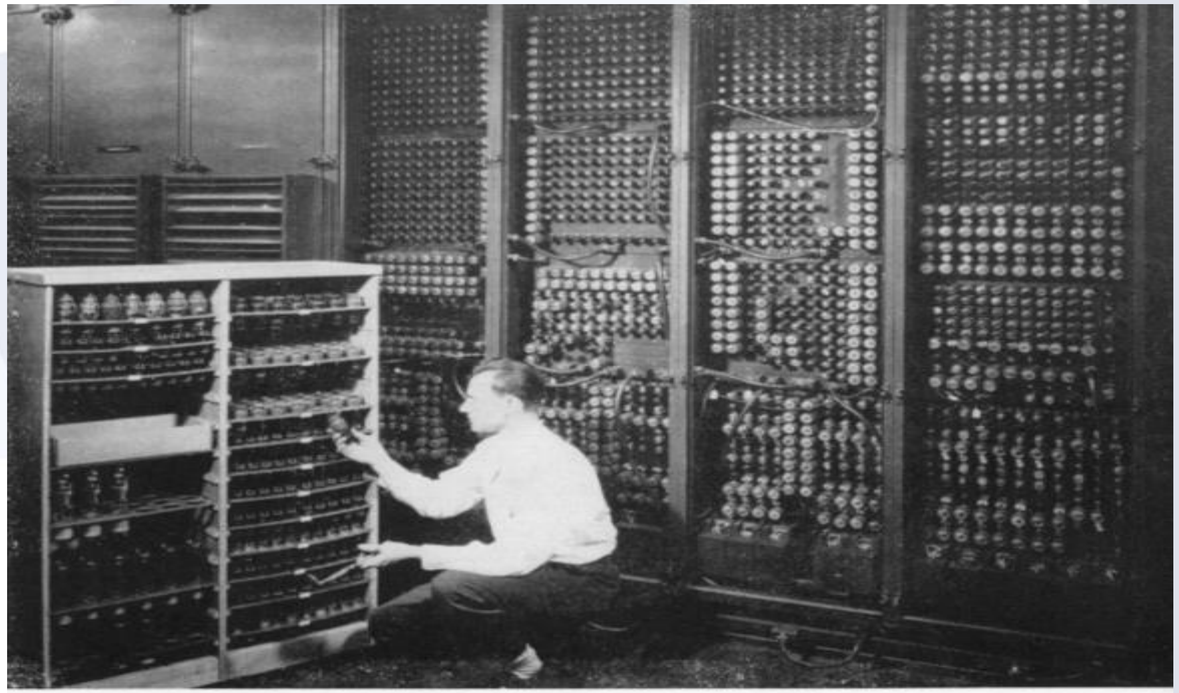
Organização e Arquitetura de Computadores

Leopoldo França

História

- Desenvolvimento da Arquitetura de Computadores
- **1ª Geração 1945-1955: Válvulas e Painéis de Programação**
  - A primeira geração de computadores digitais utilizava válvulas (o Eniac possuía 17.468 válvulas)
  - Características:
    - Grande tamanho físico (equiv. a área de um pequeno apartamento!)
    - Consumiam muita energia elétrica

http://www.fookunity.com/fook_team/HugePinball/img/components_concept/ist2_1296830-vacuum-tube.jpg



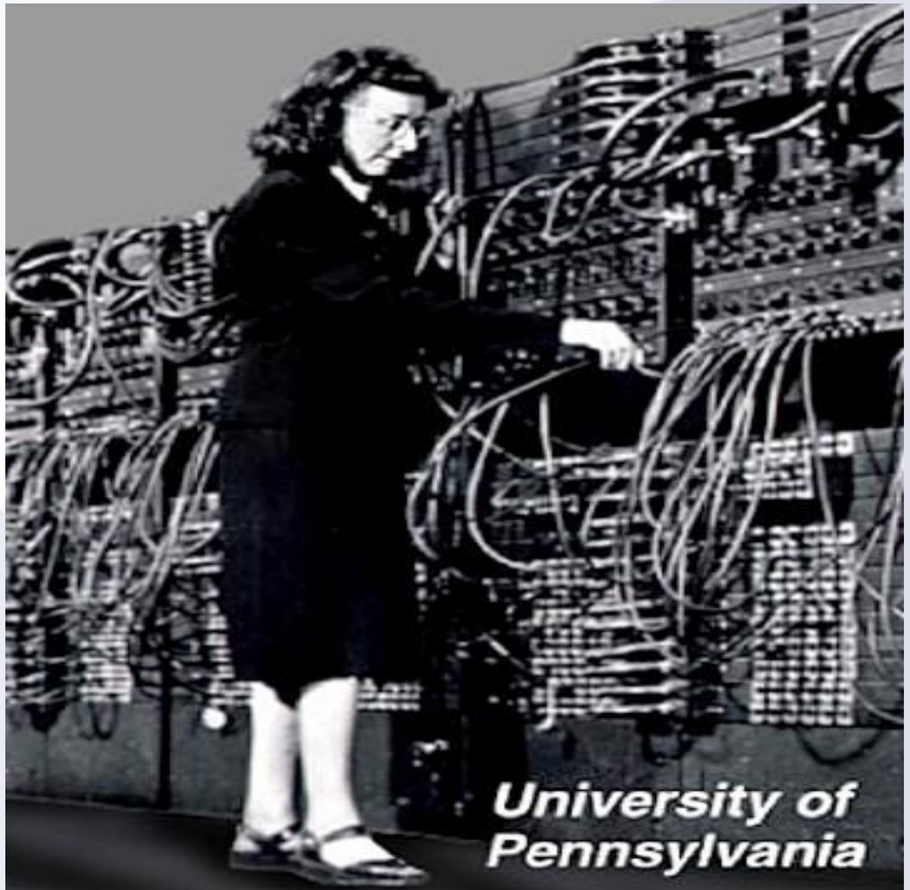
Replacing a bad tube meant checking among ENIAC's 19,000 possibilities.

http://ftp.arl.army.mil/ftp/historic-computers/gif/eniac3.gif

História

- **1ª Geração 1945-1955: Válvulas e Painéis de Programação (cont.)**
  - A “programação” era realizada em linguagem de máquina (0s e 1s)
  - Os computadores eram programado fisicamente através de painéis com plugs (no Eniac eram 6.000 conectores)

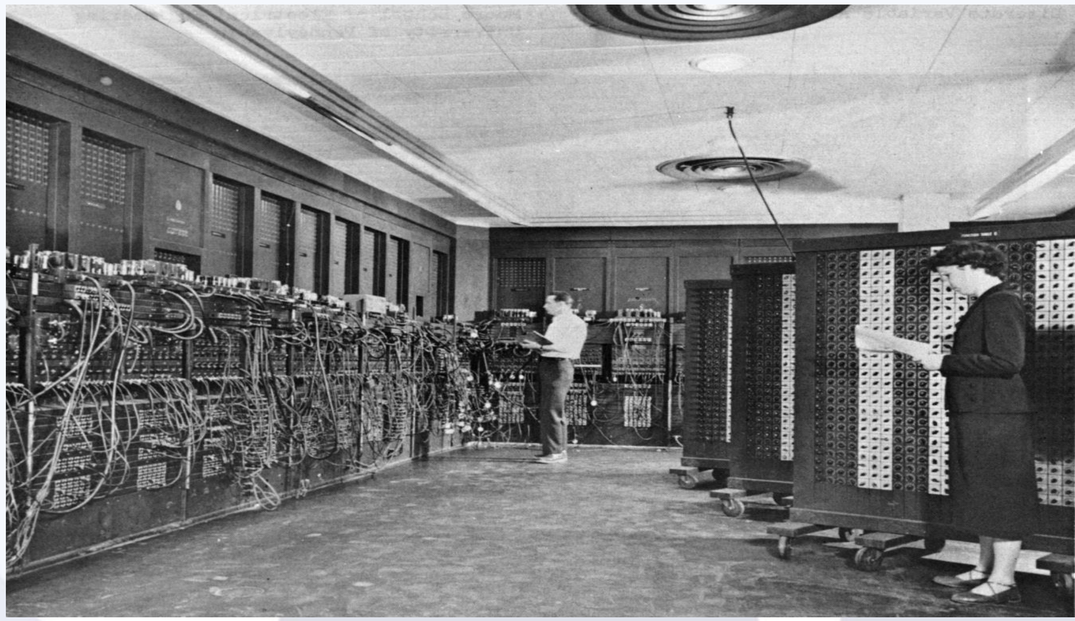
http://ftp.arl.army.mil/ftp/historic-computers/gif/eniac4.gif



http://www.plyojump.com/classes/images/computer_history/eniac_with_programmer2.jpg

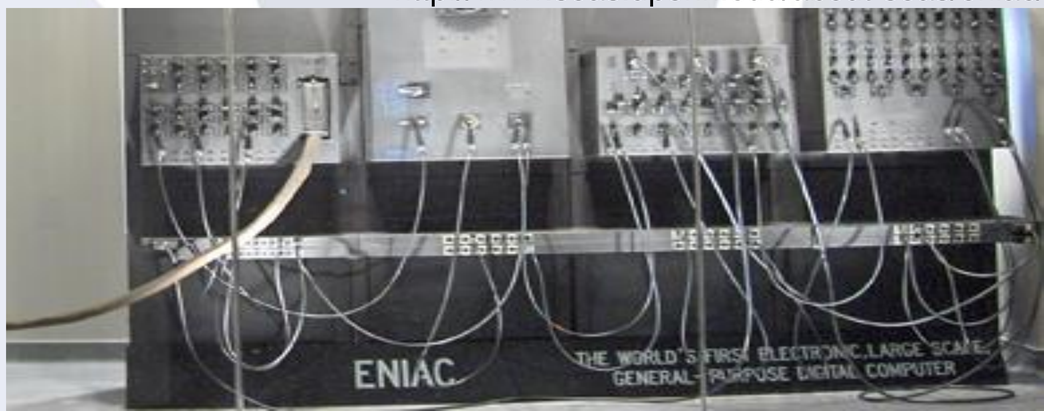
História

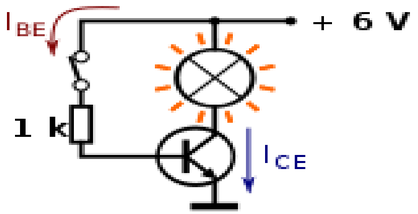
- **1ª Geração 1945-1955: Válvulas e Painéis de Programação (cont.)**
  - ENIAC (*Electronic Numerical Integrator ad Computer*, 1946)
    - 17.468 válvulas
    - 27 toneladas
    - 150 kW
    - 62m^2 de área
    - US$ 500.000,00 (6 milhões considerando a inflação até 2008)



http://upload.wikimedia.org/wikipedia/commons/4/4e/Eniac.jpg

http://www.seas.upenn.edu/about-seas/eniac/



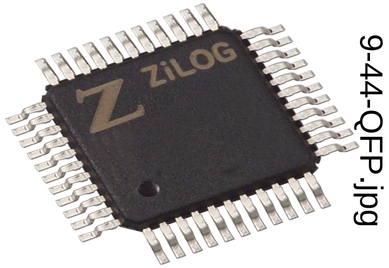


http://en.wikipedia.org/wiki/File:Transistor_as_switch.svg

História

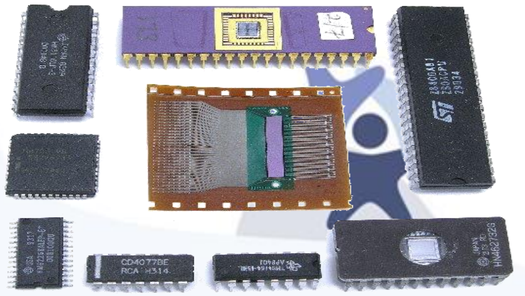
http://tojovemo.files.wordpress.com/2009/10/transistor-2.png?w=267&h=322

- **2ª Geração 1955-1965: Transistores e Sistemas em Lote (Batch)**
  - A segunda geração de computadores digitais utilizava transistores na implementação das portas lógicas
  - Os programas (**jobs**) eram perfurados em cartões, e memória magnética (fita) era utilizada para armazenar os programas
    - Inicialmente, o operador, através de uma leitora de cartões, transferia os programas para uma fita magnética
    - A fita magnética era lida pelo computador, que executava um programa de cada vez e gravava o resultado em uma fita de saída



http://media.digikey.com/photos/Zilog%20Photos/269-44-QFP.jpg

História



http://www.uobkupartnership.talktalk.net/med%20integ3.jpg

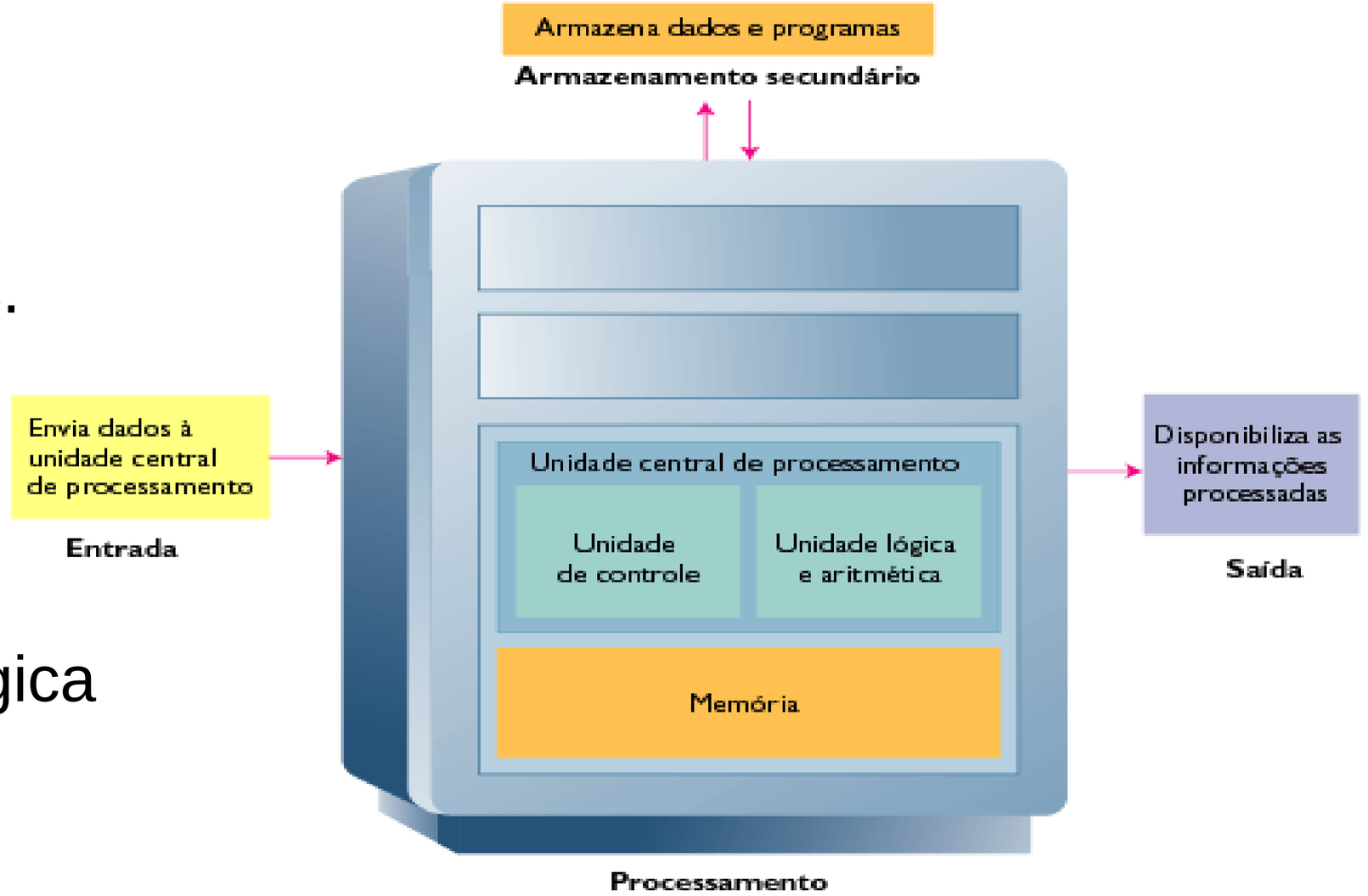
- **3ª Geração 1965-1980: Circuito Integrado (CI) e Multiprogramação**
  - Utilização de CIs na construção dos computadores
  - Desenvolvimento da **multiprogramação** através de **máquina virtual** – o objetivo era manter o processador “ocupado” executando um programa, enquanto outro programa aguarda pela conclusão de uma operação de entrada/saída
  - Desenvolvimento de **multiprocessamento** – permitindo a execução de mais de um programa simultaneamente

História

- **4ª Geração 1980-2000: LSI, VLSI, ULSI e Microprocessador**
  - Desenvolvimento de CI com integração em larga escala (**LSI** – *large scale integration,* 1.000+ componentes), muito larga escala (**VLSI** – *very large scale integration,* 10.000+ componentes) e ultra larga escala (**ULSI** – *ultra large scale integration,* 100.000+ componentes), contendo atualmente milhões de componentes/transistores por cm^2 do **chip**
  - Desenvolvimento do **microprocessador**, que deu origem ao **computador pessoal** (**microcomputador**)
  - Desenvolvimento da interface gráfica (**GUI**)
  - Desenvolvimento da arquitetura **RISC**

Unidade Central de Processamento

- Conjunto complexo de circuitos eletrônicos.
- Executa instruções de programa armazenadas.
- Duas partes:
- Unidade de controle
- Unidade aritmética e lógica (ALU)



Unidade de Controle

- Direciona o sistema do computador a executar instruções de programa armazenadas.
- Deve comunicar-se com a memória e com a ALU.
- Envia dados e instruções do armazenamento secundário para a memória, quando necessário.

Unidade Aritmética e Lógica

- Executa todas as operações aritméticas e lógicas.
- Operações aritméticas:
  - Adição, subtração, multiplicação, divisão.
- Operações lógicas:
  - Compara números, letras ou caracteres especiais.
  - Testa de condições (operações relacionais)

Registradores

- Áreas de armazenamento temporário de alta velocidade.
  - Localizações de armazenamento situadas dentro da CPU.
- Funcionam sob direção da unidade de controle:
  - Recebem, guardam e transferem instruções ou dados.
  - Controlam onde a próxima instrução a ser executada ou os dados necessários serão armazenados.

Bit

- Abreviação de binary digit (dígito binário).
  - Dois valores possíveis: 0 e 1.
  - Nunca pode estar vazio.
- Unidade básica para armazenar dados:
  - 0 significa desligado; 1 significa ligado.

Byte

- Um grupo de 8 bits.
  - Cada byte tem 256 (2^8) valores possíveis.
- Dispositivos de memória e armazenamento são medidos em número de bytes.

Palavra

- O número de bits que a CPU processa como uma unidade.
  - Tipicamente, um número inteiro de bytes.
  - Quanto maior a palavra, mais potente é o computador.
  - Computadores pessoais tipicamente têm 32 ou 64 bits de extensão de palavras.

O Barramento (*Bus*) do Sistema

- Percursos elétricos paralelos que transportam dados entre a CPU e a memória.
- Largura de barramento:
  - O número de percursos elétricos para transportar dados.
  - Medida em bits.
- Velocidade de barramento:
  - Medida em megahertz (MHz).

Largura de Barramento

- Tipicamente, a mesma largura do tamanho de palavra da CPU.
- Com um tamanho de barramento maior, a CPU pode:
  - Transferir mais dados simultaneamente:
    - Torna o computador mais rápido.
  - Referenciar números de endereço de memória maiores:
    - Permite mais memória.
  - Suportar um número e uma variedade maiores de instruções.

Velocidade de Barramento

- Quanto maior a velocidade de barramento, mais rapidamente os dados viajarão por meio do sistema.

Velocidades dos Microprocessores

- **Medida da velocidade de *clock* do sistema:**
  - Quantos pulsos eletrônicos o clock produz por segundo.
  - Usualmente, expressa em gigahertz (GHz).
    - Billhões de ciclos de máquina por segundo.
    - Alguns PCs antigos mediam em megahertz (MHz).
- **Uma comparação de velocidades de *clock* somente é significativa entre microprocessadores idênticos.**

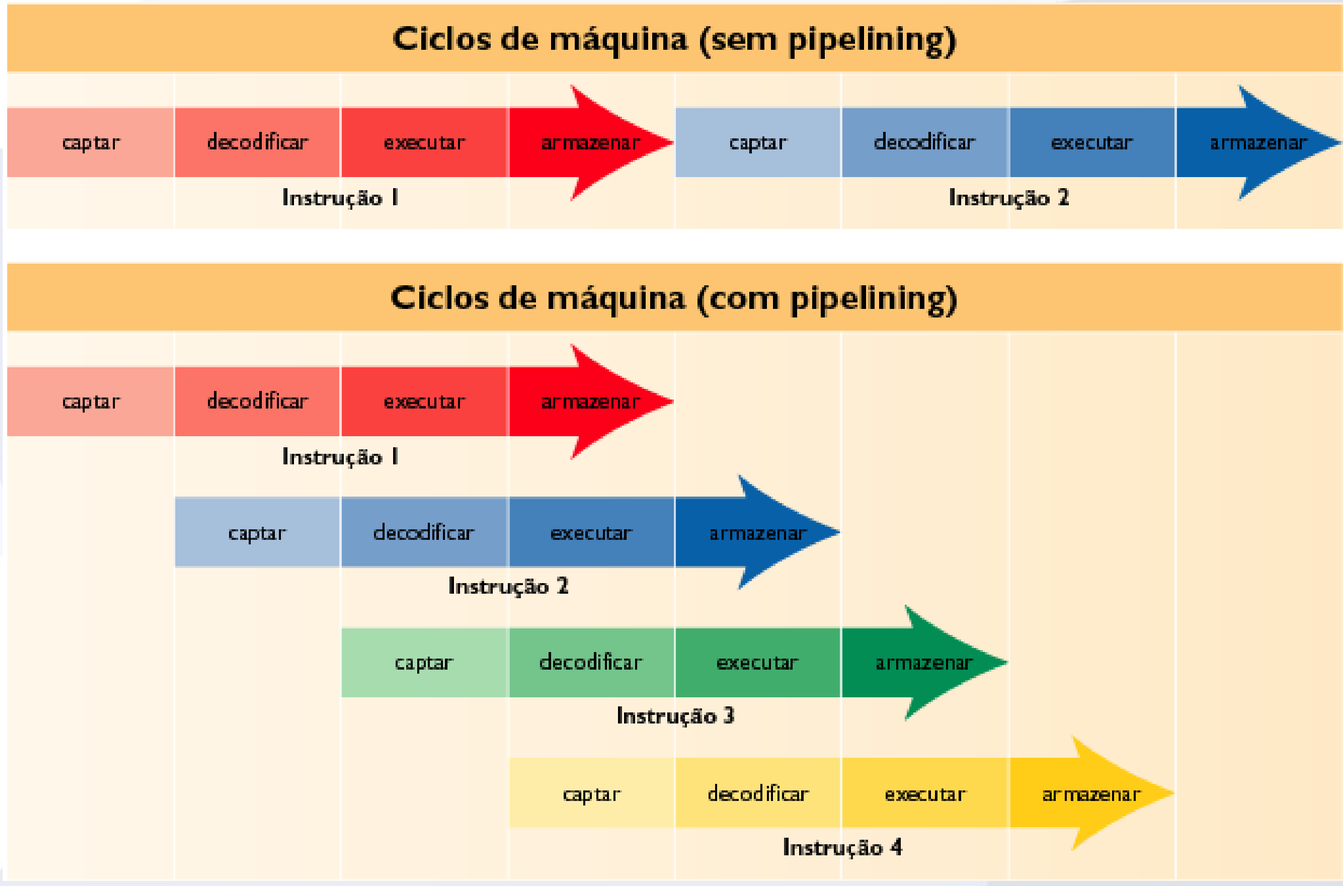
Tecnologia RISC

- Computação com um Conjunto Reduzido de Instruções – *Reduced Instruction Set Computing*
  - Usa um pequeno subconjunto de instruções.
  - Um menor número de instruções aumenta a velocidade.
  - Inconveniente: operações complexas têm de ser divididas em uma série de instruções de tamanho menor.

- Computação com um Conjunto Complexo de Instruções – *Complex Instruction Set Computing* (CISC)

Pipelining

- **Introduz uma nova instrução na CPU a cada etapa do ciclo de máquina.**
  - A instrução 2 é captada quando a instrução 1 é decodificada, em vez de esperar até que o ciclo se complete.



Processamento Paralelo

- O processador de controle divide o problema em partes:
  - Cada parte é enviada a um processador distinto.
  - Cada processador tem sua própria memória.
  - O processador de controle monta os resultados.
- Alguns computadores que usam processamento paralelo operam em termos de *teraflops*: trilhões de instruções com ponto flutuante por segundo.

Representação de números binárias

- As palavras de um computador são compostas por bits.

Ex.

$$10110100_2$$

- Os números naturais podem ser representados tanto em decimal quanto em binário.

Ex.

$$0100_2 = (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) = 4_{10}$$

Representação de números negativos

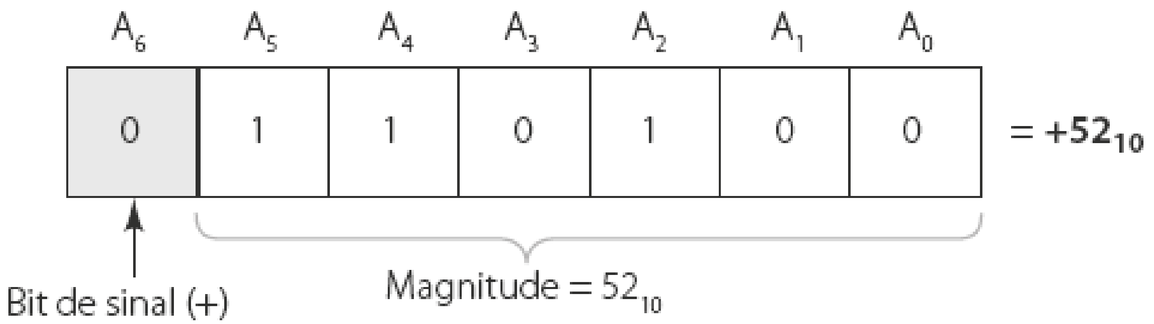
- **Notação sinal/magnitude**
  - Cada número possui um bit adicional que representa o sinal.

Ex.

$$\mathbf{1}10100_2 = -4$$

Bit de sinal

Figura 2.2 Representação de números com sinal na forma sinal-magnitude.



Fonte: Tocci, Widmer e Moss (2011, p. 254).

Representação de números negativos

- **Notação complemento de dois**
  - Números com zero (0) à esquerda são considerados positivos, números com um (1) à esquerda são considerados negativos.

$$000_2 = (\mathbf{0} \times 2^2) + (\mathbf{0} \times 2^1) + (\mathbf{0} \times 2^0) = 0$$

$$001_2 = (\mathbf{0} \times 2^2) + (\mathbf{0} \times 2^1) + (\mathbf{1} \times 2^0) = 1$$

$$010_2 = (\mathbf{0} \times 2^2) + (\mathbf{1} \times 2^1) + (\mathbf{0} \times 2^0) = 2$$

$$011_2 = (\mathbf{0} \times 2^2) + (\mathbf{1} \times 2^1) + (\mathbf{1} \times 2^0) = 3$$

$$100_2 = (\mathbf{1} \times -(2^2)) + (\mathbf{0} \times 2^1) + (\mathbf{0} \times 2^0) = -4$$

$$101_2 = (\mathbf{1} \times -(2^2)) + (\mathbf{0} \times 2^1) + (\mathbf{1} \times 2^0) = -3$$

$$110_2 = (\mathbf{1} \times -(2^2)) + (\mathbf{1} \times 2^1) + (\mathbf{0} \times 2^0) = -2$$

$$111_2 = (\mathbf{1} \times -(2^2)) + (\mathbf{1} \times 2^1) + (\mathbf{1} \times 2^0) = -1$$

Representação de números negativos

- **Notação complemento de dois**

Figura 2.3 Representação de números com sinal na forma de complemento de 2.

Fonte: Tocci, Widmer e Moss (2011, p. 255).

Regra prática para negação

- **Considere**
  - x é um número em complemento a dois.
  - $\overline{x}$ é a representação invertida de x.

    Ex. $x = 011_2$ então $\overline{x} = 100_2$
    - A soma $x + \overline{x} = -1$, portanto

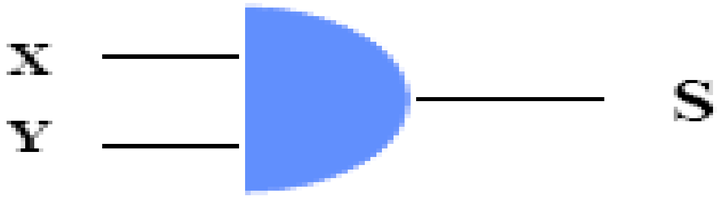
      $x + \overline{x} + 1 = 0$
- Então $-x = \overline{x} + 1$;

  Ex. $x = 011_2$ (3) então $-x = 100_2 + 001_2 = 101_2$ (-3)

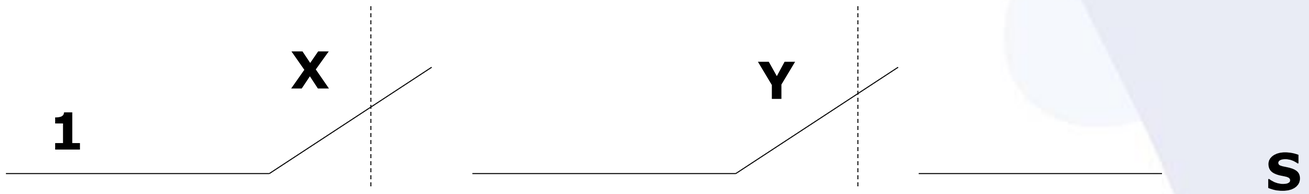
Portas lógicas e álgebra booleana

- AND (e)

$$S = X \cdot Y$$



X	Y	S
0	0	0
0	1	0
1	0	0
1	1	1



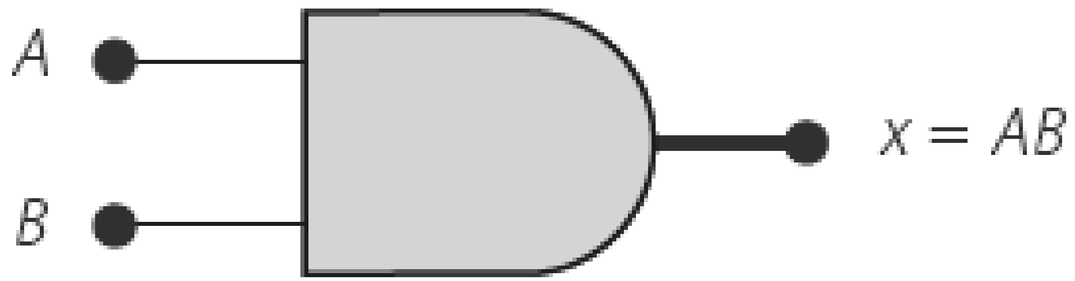
Portas lógicas e álgebra booleana

Figura 2.6 (a) Tabela-verdade para a operação AND; (b) símbolo da porta AND.

AND

A	B	$x = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

(a)



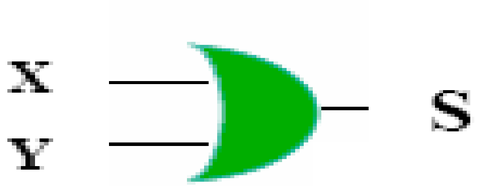
(b)

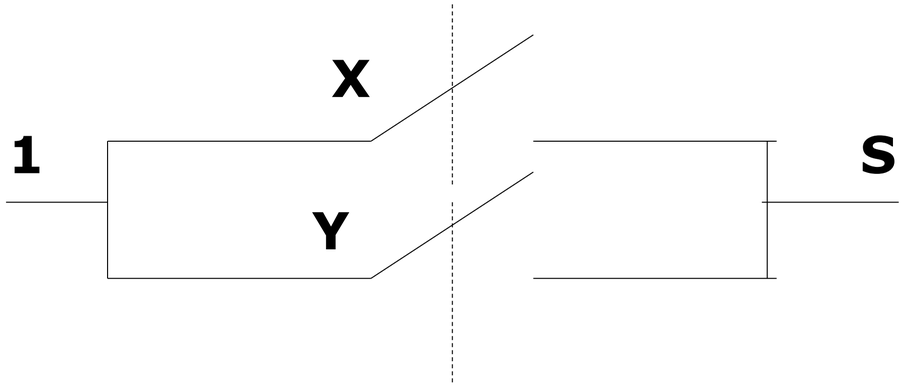
Fonte: Tocci, Widmer e Moss (2011, p. 54).

Portas lógicas e álgebra booleana

- OR (ou)

$$S = X + Y$$





X	Y	S
0	0	0
0	1	1
1	0	1
1	1	1

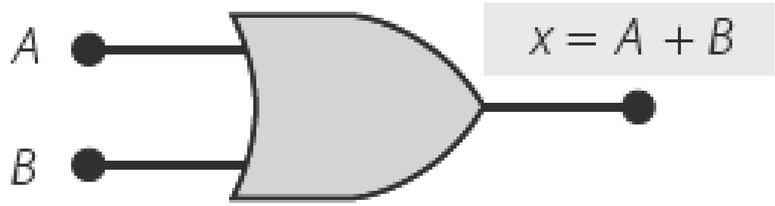
Portas lógicas e álgebra booleana

Figura 2.5 (a) Tabela-verdade de operação OR; (b) símbolo de uma porta OR de duas entradas.

OR

A	*B*	*x* = *A* + *B*
0	0	0
0	1	1
1	0	1
1	1	1

(a)



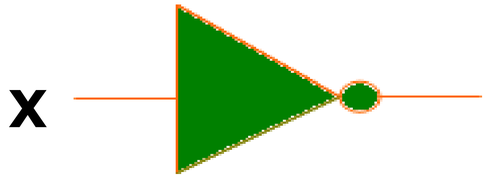
Porta OR

(b)

Fonte: Tocci, Widmer e Moss (2011, p. 51).

Portas lógicas e álgebra booleana

- Inversor – porta NOT (não)

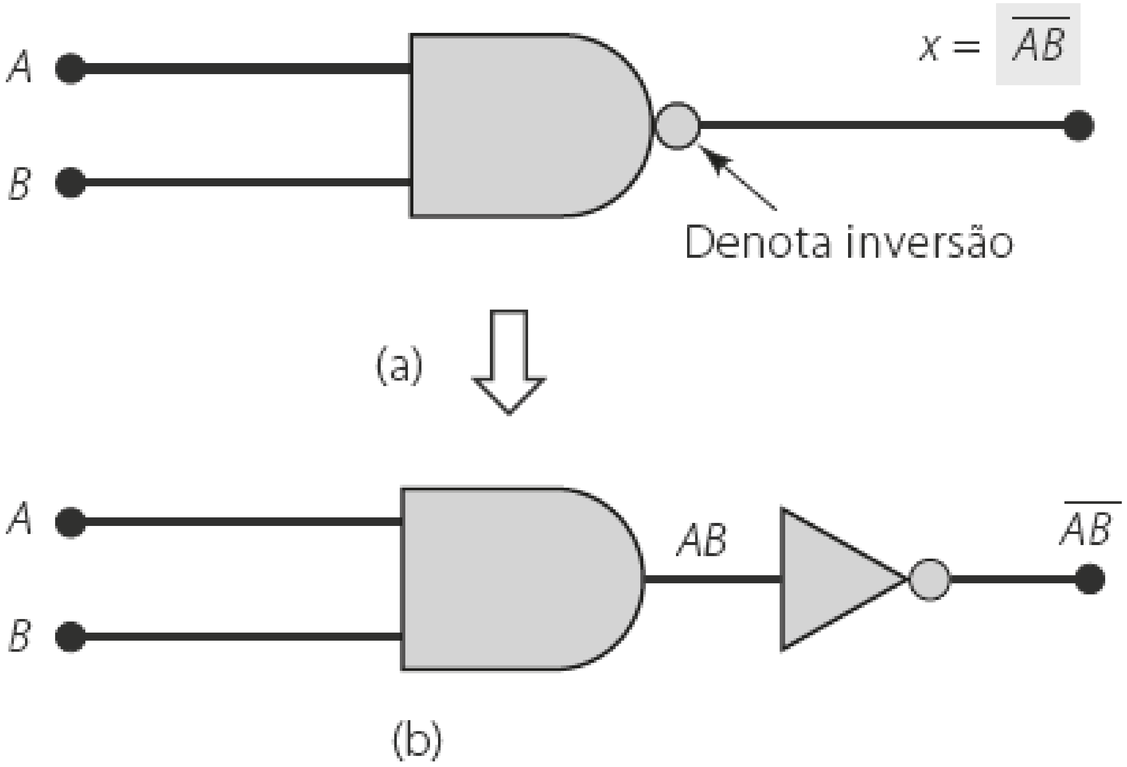


x	$\overline{x}$
0	1
1	0

Portas lógicas e álgebra booleana

- Podemos fazer associações das portas lógicas e formar as portas: NAND (não-e)

Figura 2.8 (a) Símbolo da porta NAND; (b) circuito equivalente; (c) tabela-verdade.



		AND	NAND
A	B	AB	$\overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

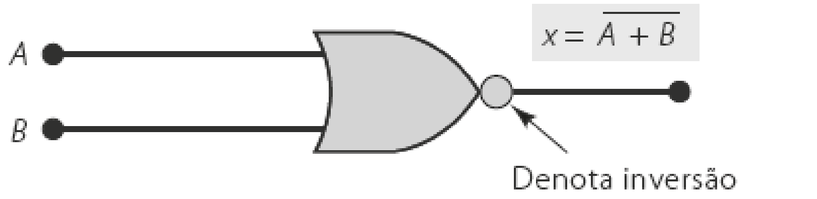
(c)

Fonte: Tocci, Widmer e Moss (2011, p. 66).

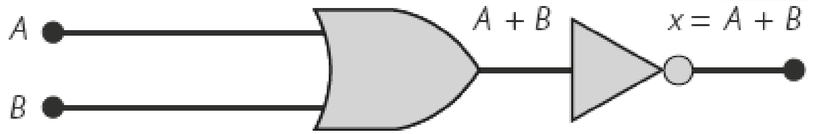
Portas lógicas e álgebra booleana

- Podemos fazer associações das portas lógicas e formar as portas: NOR (não-ou)

Figura 2.7 (a) Símbolo da porta NOR; (b) circuito equivalente; (c) tabela-verdade.



(a)



(b)

		OR	NOR
A	B	$A + B$	$\overline{A + B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

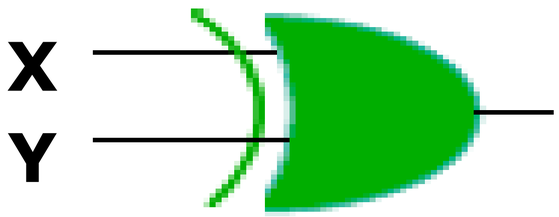
(c)

Fonte: Tocci, Widmer e Moss (2011, p. 64).

Portas lógicas e álgebra booleana

- XOR (Ou-exclusivo)

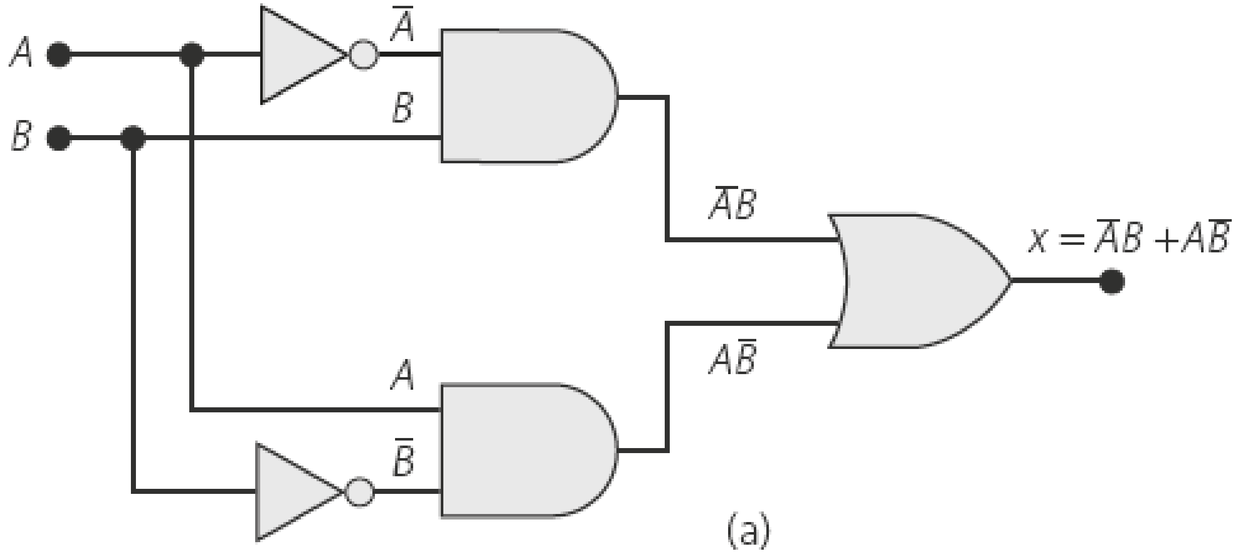
$$S = X \oplus Y$$



X	Y	S
0	0	0
0	1	1
1	0	1
1	1	0

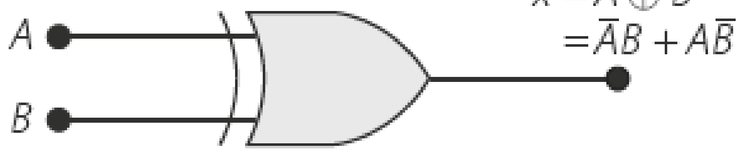
Portas lógicas e álgebra booleana

Figura 2.9 (a) Circuito XOR e tabela-verdade; (b) símbolo tradicional para a porta XOR.



(a)

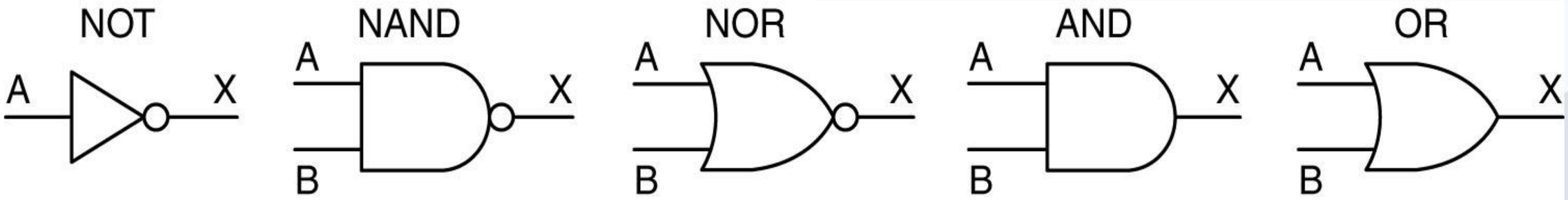
A	B	x
0	0	0
0	1	1
1	0	1
1	1	0



(b)

Fonte: Tocci, Widmer e Moss (2011, p. 122).

Portas e Álgebra Booleana



A	X
0	1
1	0

(a)

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

(b)

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

(c)

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

(d)

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

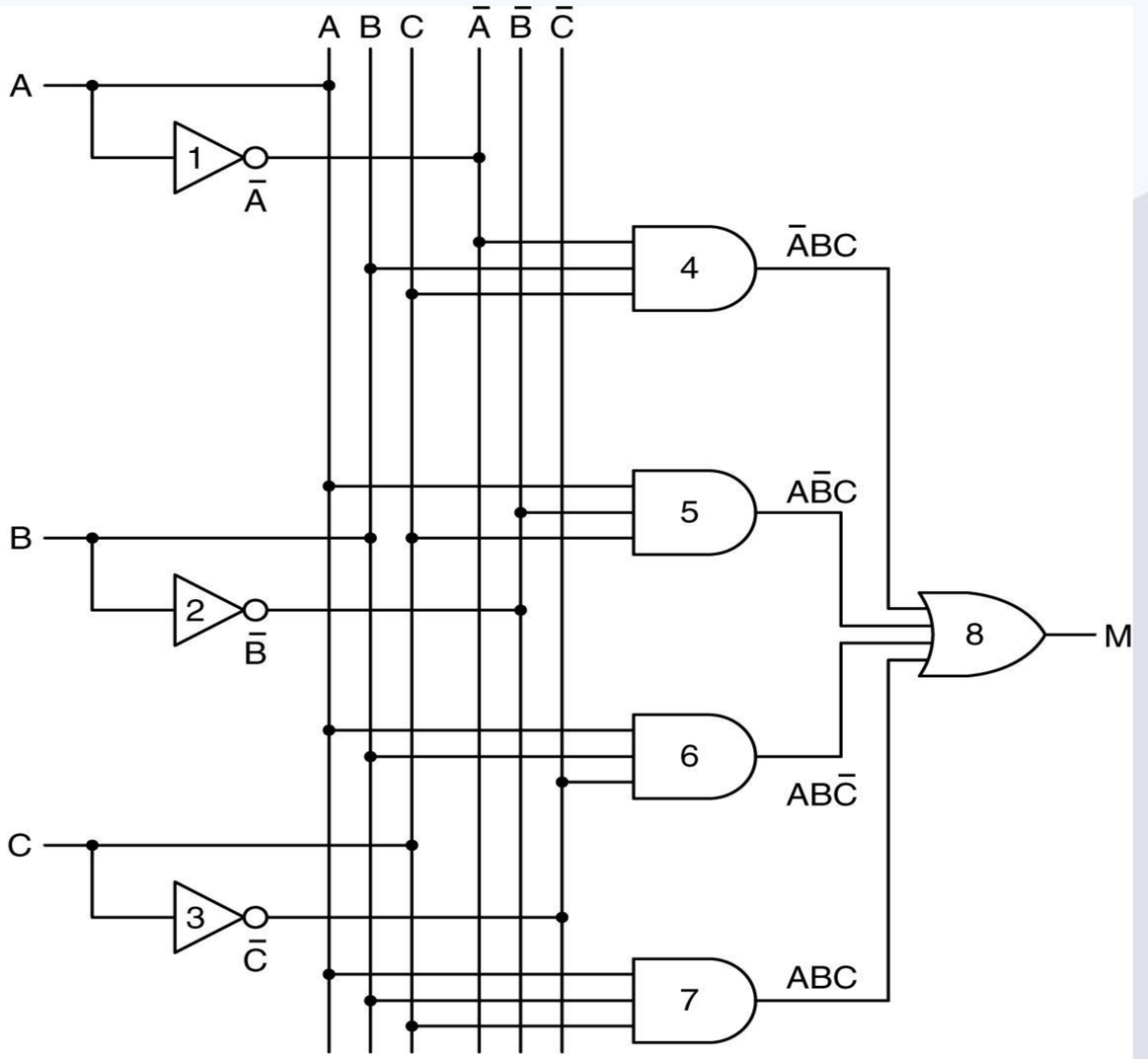
(e)

Portas lógicas e álgebra booleana

- (a) Tabela-verdade para a função majoritária de três variáveis.
- (b) Um circuito que implementa a função descrita em (a).

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

(a)



(b)

Equivalência de circuitos (1)

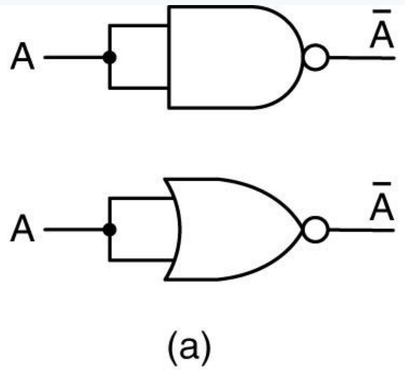
Construção de portas

(a) NOT

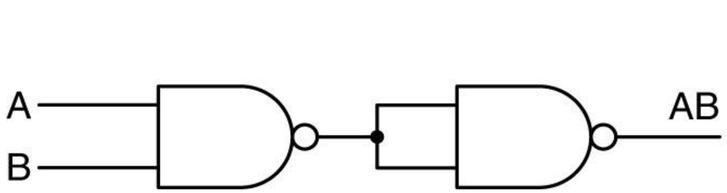
(b) AND

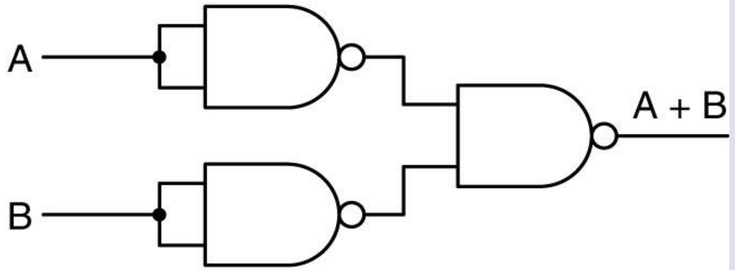
(c) OR

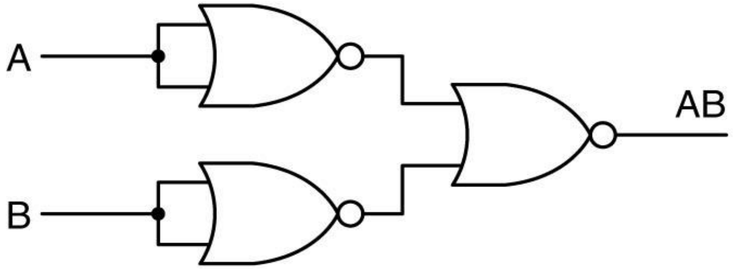
usando somente portas NAND ou somente portas NOR.



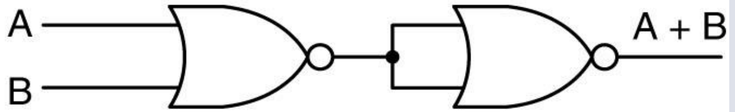
(a)





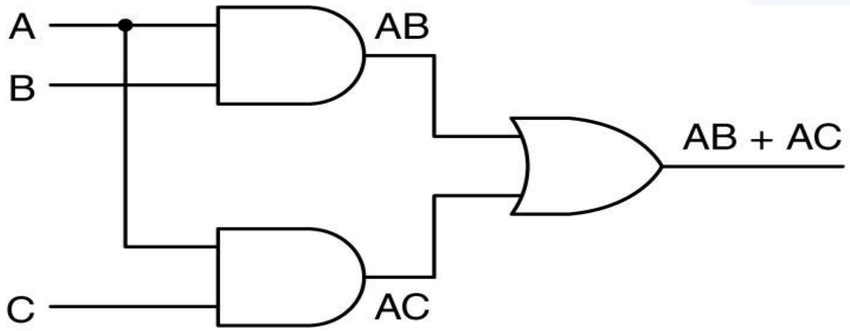


(b)



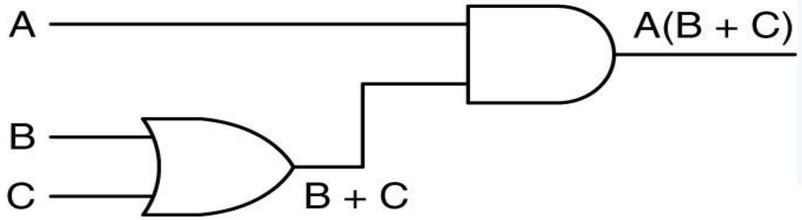
(c)

Equivalência de circuitos (2)



A	B	C	AB	AC	AB + AC
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

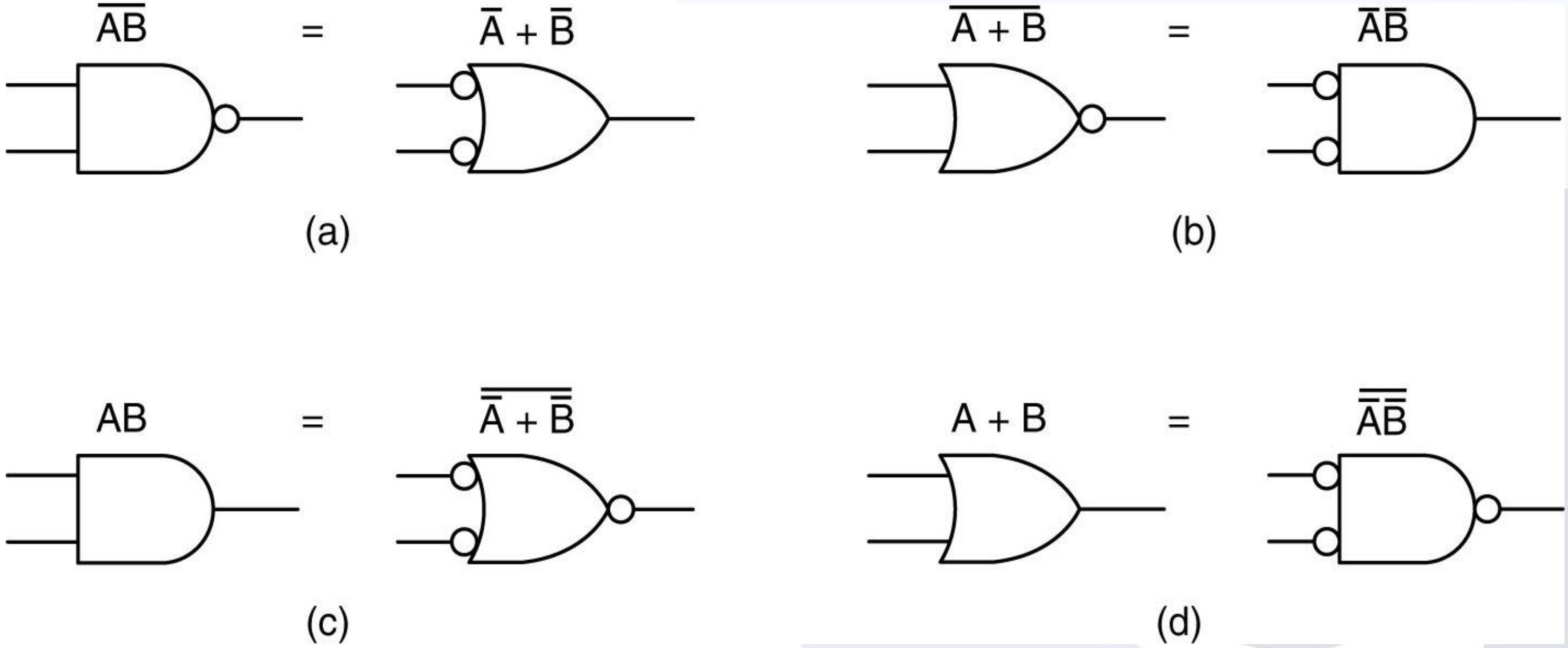
(a)



A	B	C	A	B + C	A(B + C)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

(b)

Equivalência de circuitos (3)



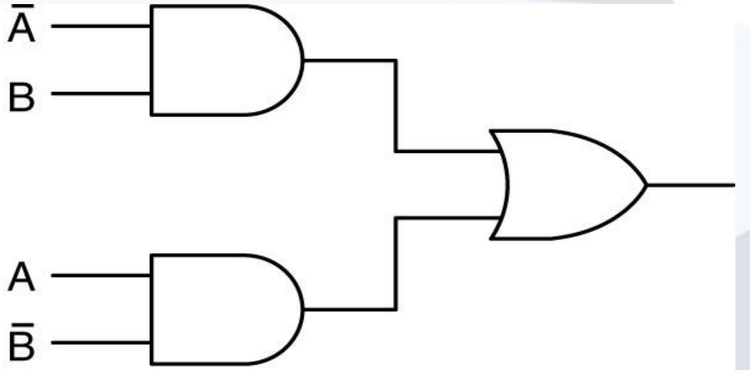
Símbolos alternativos para algumas portas:
(a) NAND, (b) NOR, (c) AND, (d) OR

Equivalência de circuitos (4)

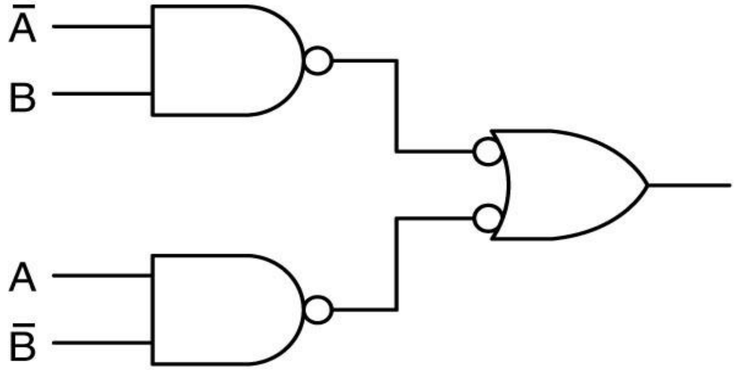
- (a) Tabela-verdade para a função XOR.
- (b-d) Três circuitos para calcular essa tabela.

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

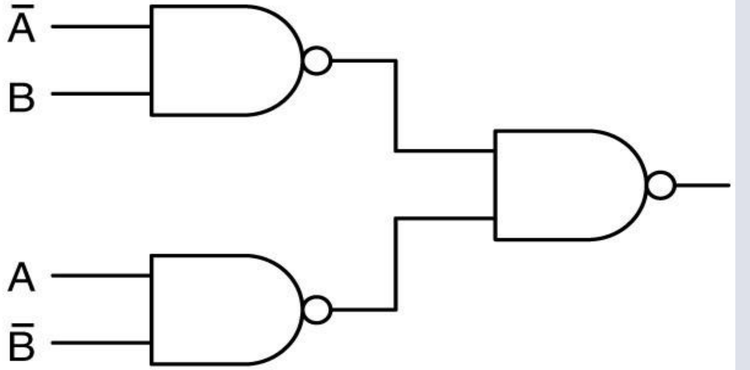
(a)



(b)



(c)



(d)

Equivalência de circuitos (5)

A	B	F
0^V	0^V	0^V
0^V	5^V	0^V
5^V	0^V	0^V
5^V	5^V	5^V

(a)

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

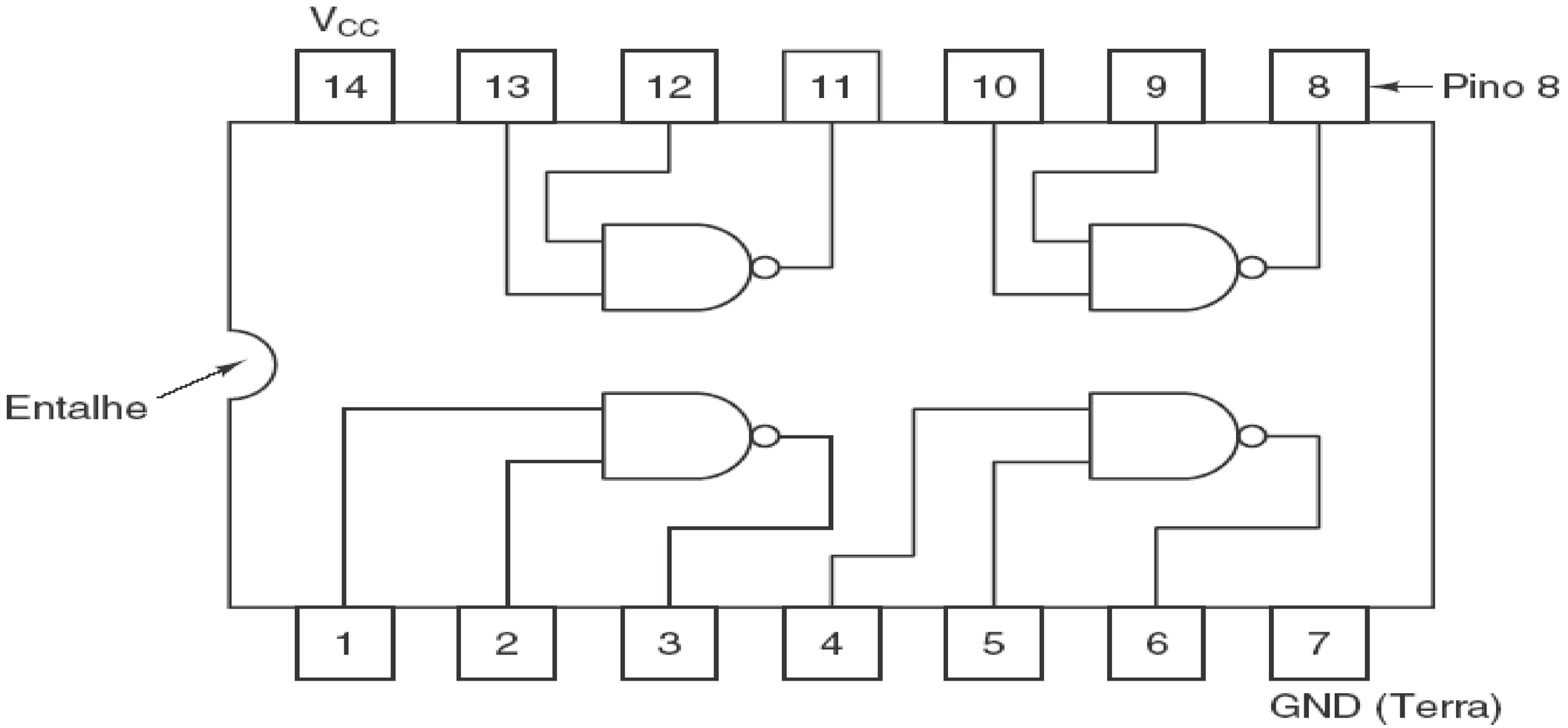
(b)

A	B	F
1	1	1
1	0	1
0	1	1
0	0	0

(c)

- (a) Características elétricas de um dispositivo.
- (b) Lógica positiva.
- (c) Lógica negativa.

Circuitos integrados



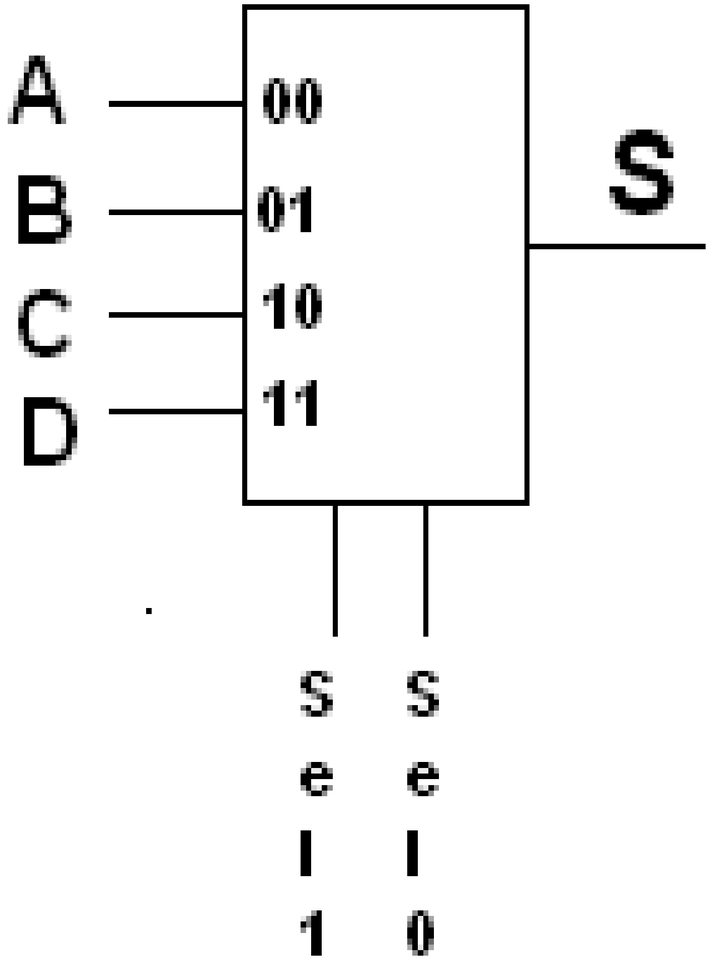
- Chip SSI que contém quatro portas.

Multiplexador

- O multiplexador é um dispositivo que possui:
  - 2^n entradas
  - 1 saída
  - n sinais de controle

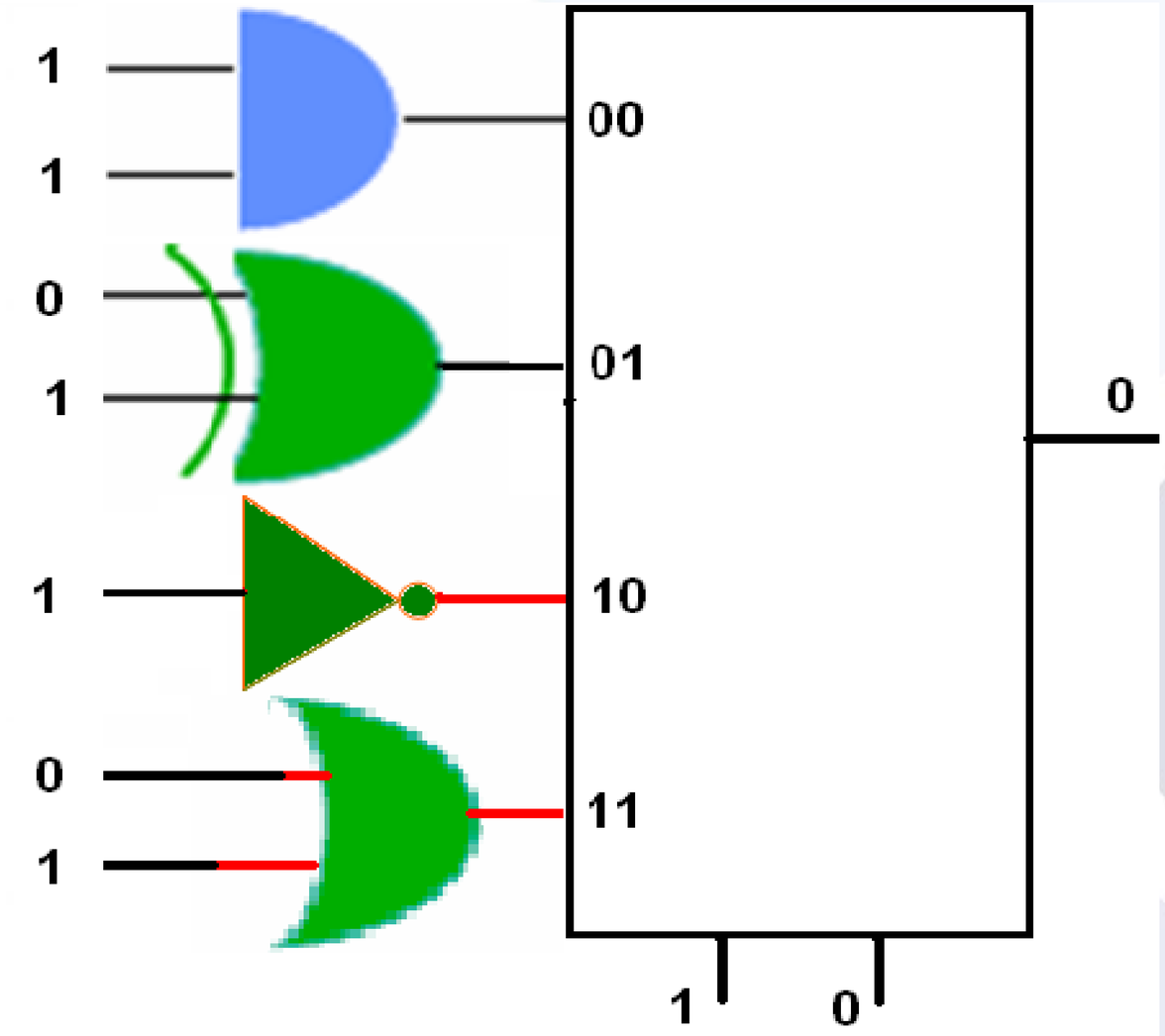
Multiplexador

- Ex. mux 4 para 1

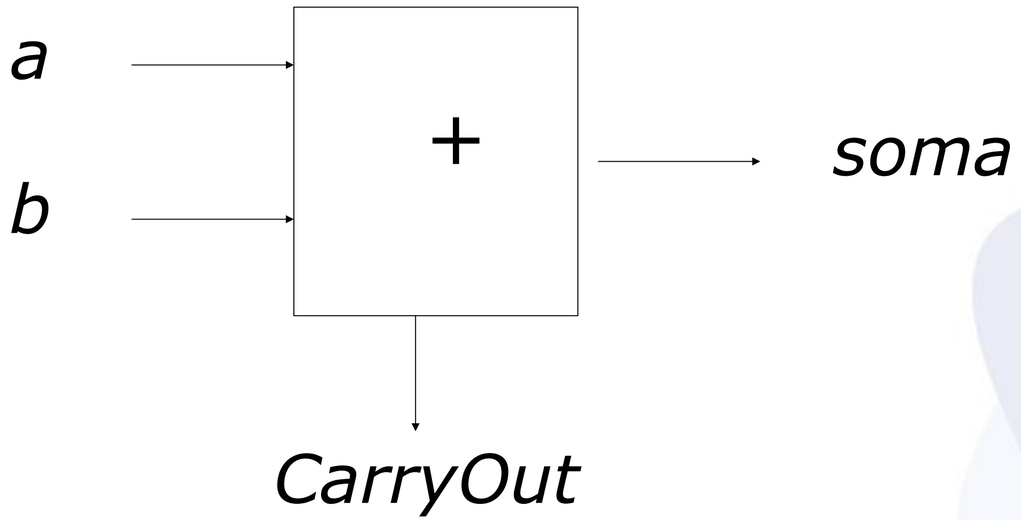


Sel1	Sel0	S
0	0	A
0	1	B
1	0	C
1	1	D

Exemplos

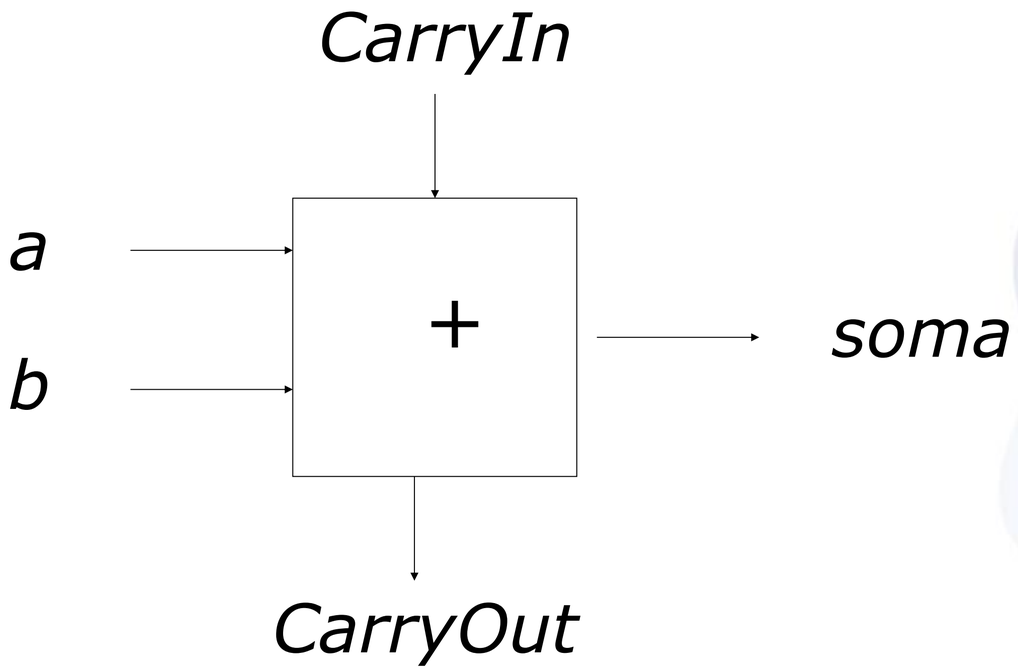


O meio somador de 1 bit



a	*b*	*Soma*	*CarryOut*
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

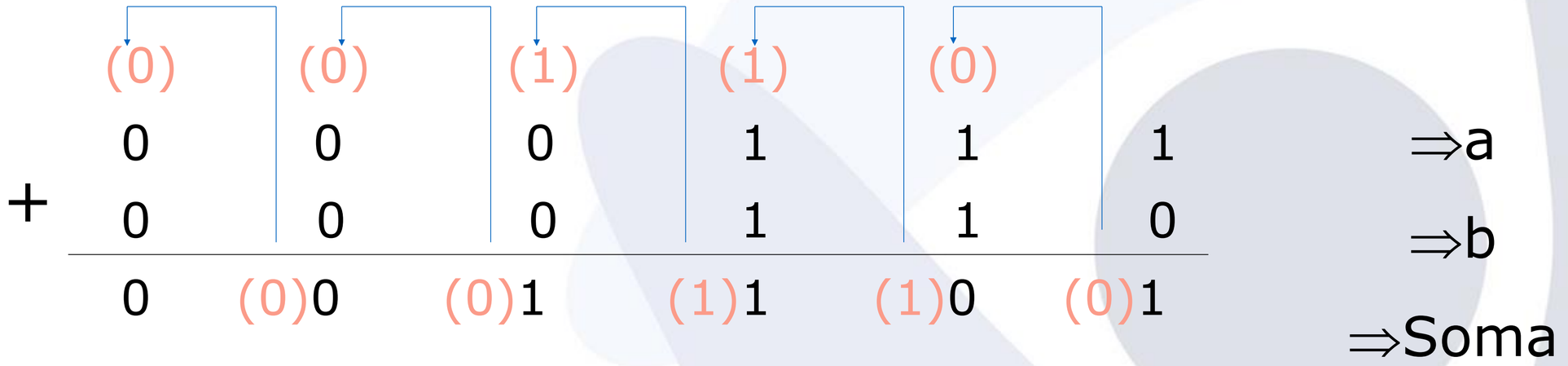
O somador completo de 1 bit



a	*b*	*CarryIn*	*Soma*	*CarryOut*
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Soma binária

- Os bits são somados um a um, da direita para a esquerda, com os ***carries*** sendo passados para o próximo bit à esquerda.



Subtração binária

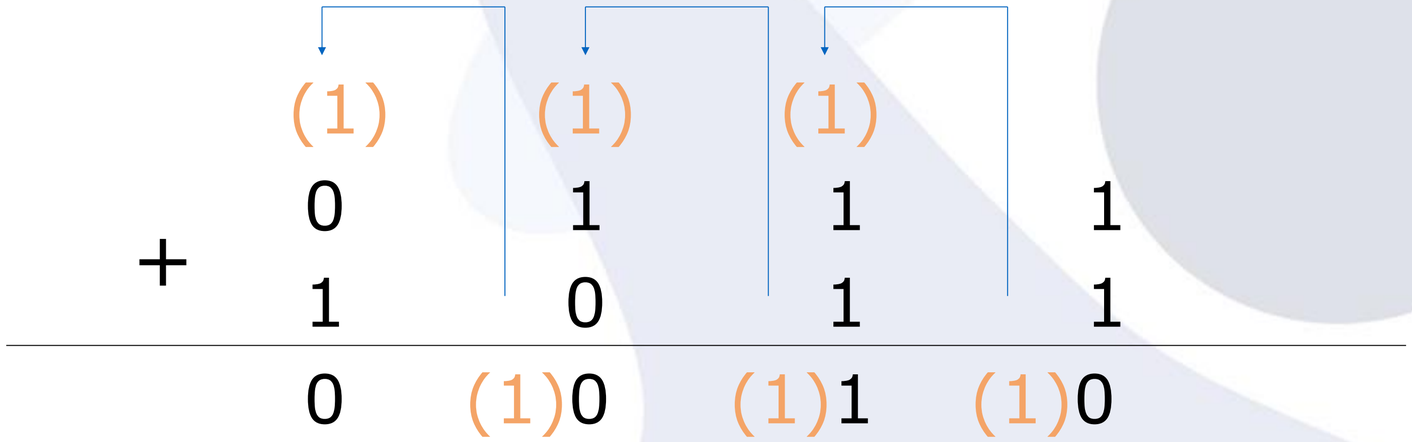
- Nega-se o segundo operando, e soma-se o resultado ao primeiro.

Ex. 7 – 5 = 7 + (-5)

$7 = 0111_2$

$5 = 0101_2$

$-5 = 1011_2$



Overflow

- Ocorre quando o resultado da operação não pode ser representado, com uma palavra de ***n*** bits.

Ex. $4_{10} = 0100_2$
$5_{10} = 0101_2$

$5 + 4 = 9$

```
  0100
+ 0101
------
  1001
```

- 7

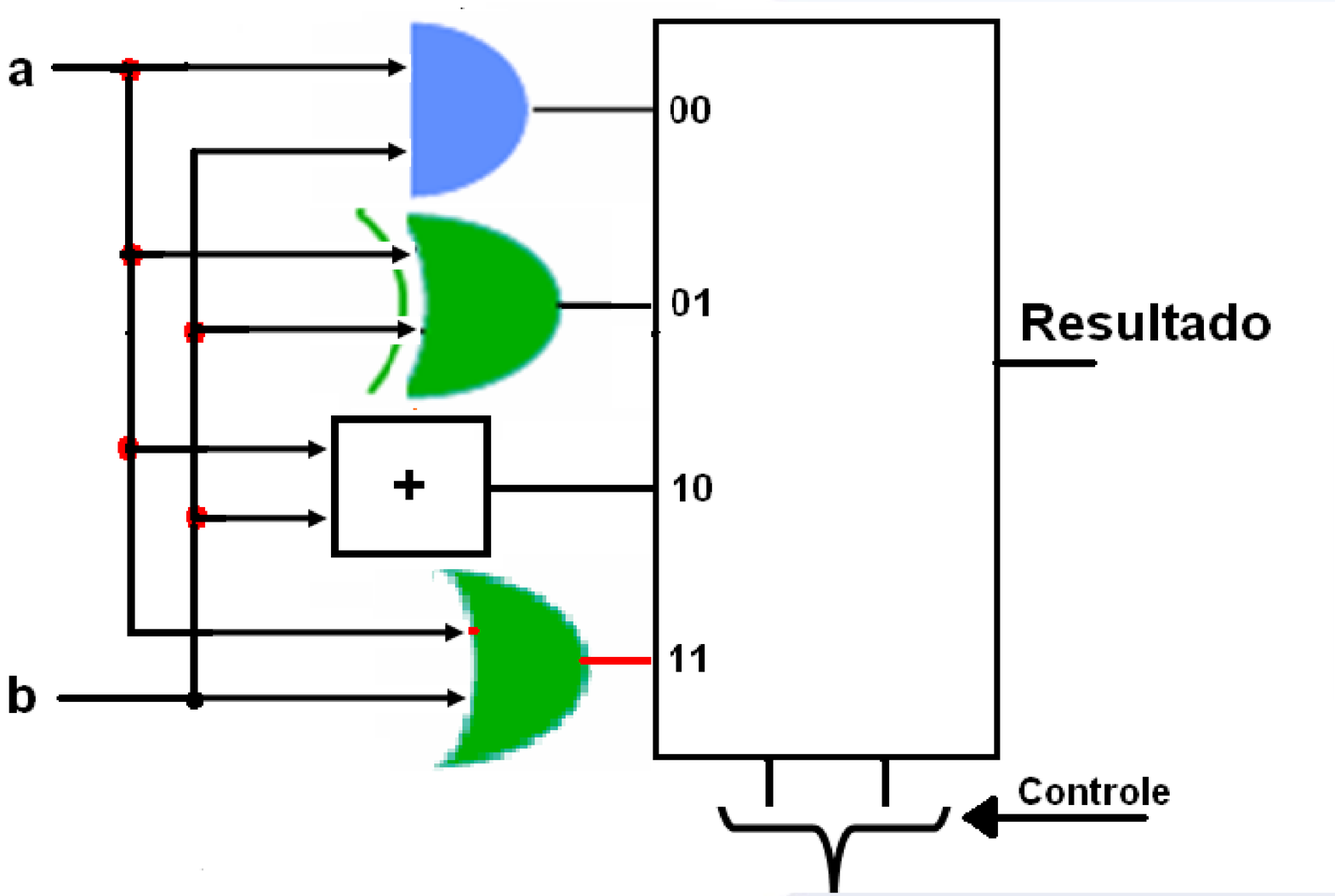
```
  00100
+ 00101
-------
  01001
```

9

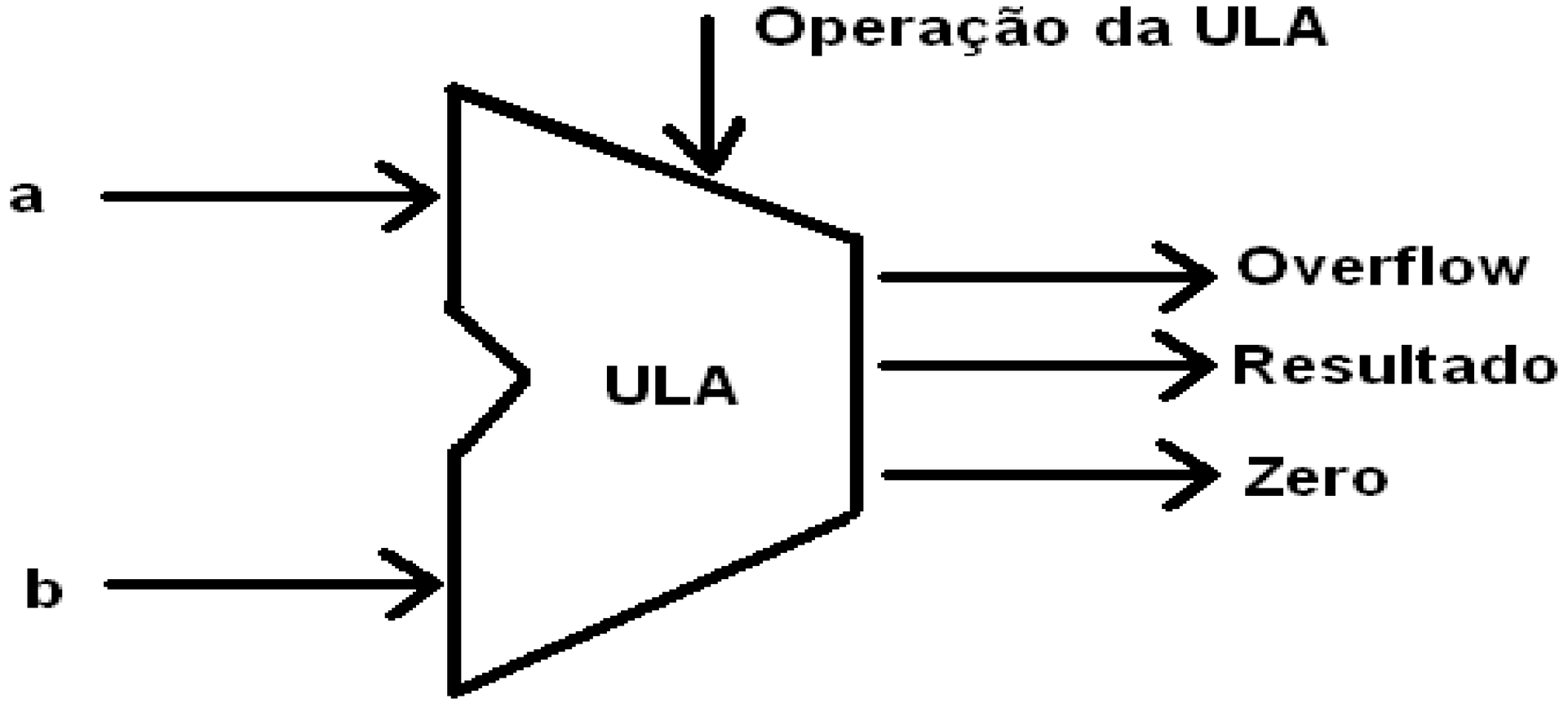
Unidade lógica aritmética (ULA)

- É o dispositivo que realiza operações aritméticas (soma, subtração,...) e lógicas (and, or,...) .
- São os músculos do computador.
- Com os conhecimentos adquiridos até então, podemos criar uma ULA de 1 bit.

ULA



ULA



Exercícios de fixação

1) Realiza as operações em binário, e indique quando houver overflow:

   a) 0110 + 0010 =

   b) 1100 + 0110 =

   c) 1010 + 1001 =

   d) 0001 + 0110 =

Obs. Considere números de 4 bits na notação complemento de dois.

OBRIGADO(A)

LEOPOLDO FRANÇA

Professor