



# Introdução à Computação (I.C)

Unidade 02

Prof. Daniel Caixeta



# Conteúdo programático

4

## Representação da Informação: Dos *bits* aos *bytes* [...]

- 4.1. Conceito de *bit* e *byte*.
- 4.2. Possibilidades de representação.

5

## Conversão numérica: Como o computador pensa e executa.

- 5.1. Sistema posicional.
- 5.2. As bases [...].
- 5.3. Conversão entre bases 2, 8 e 16.
- 5.4. Conversão de base *b* para base 10.
- 5.5. Conversão de base 10 para base *b*.
- 5.6. Números binários negativos.
- 5.7. Aritmética binária.

6

## A lógica binária: Circuitos lógicos e operadores.

- 6.1. Sistemas dicotômicos e a Álgebra de Boole.
- 6.2. Interruptores.
- 6.3. A lógica binária.
- 6.4. A soma em um computador.

## Referências





# 4. REPRESENTAÇÃO DA INFORMAÇÃO.

Dos *bits* aos *bytes* [...]

## 4.1. 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, e pode assumir apenas dois valores, 0 ou 1. (FARIAS, 2013).
- Já um *byte* é uma sequência de 8 *bits*.
- Portanto, o *byte* é a menor unidade de armazenamento utilizada pelos computadores. Isto quer dizer, que nunca conseguiremos salvar menos do que 8 *bits* em uma informação.

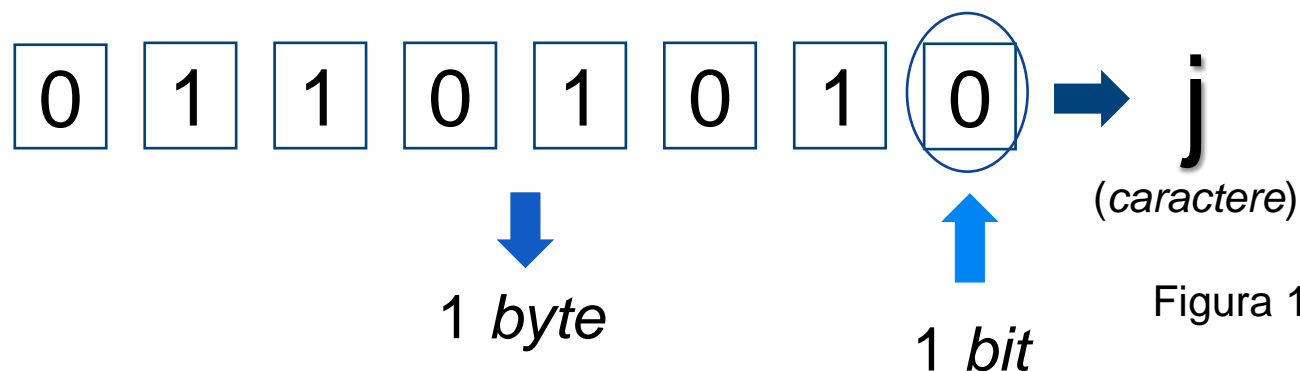


Figura 1. Representação de um *byte* e um *bit*.

- Todo dispositivo de armazenamento indica o número de *bytes* (8 *bits*) que ele pode conter.

Tabela 1. Principais unidades de medidas e sua base exponencial.

Unidade	Símbolo	Número de <i>bytes</i>	Base exponencial
<i>Kilobyte</i>	KB	1.024	$2^{10}$
<i>Megabyte</i>	MB	1.048.576	$2^{20}$
<i>Gigabyte</i>	GB	1.073.741.824	$2^{30}$
<i>Terabyte</i>	TB	1.099.511.627.776	$2^{40}$
<i>Petabyte</i>	PB	1.125.899.906.842.624	$2^{50}$
<i>Exabyte</i>	EB	1.152.921.504.606.846.976	$2^{60}$
<i>Zettabyte</i>	ZB	1.180.591.620.717.411.303.424	$2^{70}$
<i>Yottabyte</i>	YB	1.208.925.819.614.629.174.706.176	$2^{80}$

---

## 4.2. POSSIBILIDADES DE REPRESENTAÇÃO

- Um *bit* só pode assumir dois valores (0 ou 1), portanto, só será possível representar exatamente dois estados distintos. (FARIAS, 2013).





Tabela 2. Representação com um *bit*.

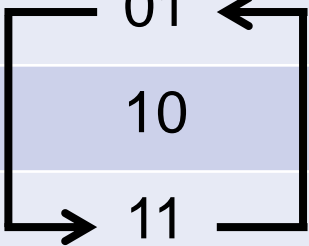
Bit	Porta	Lâmpada	Detector de movimento	Estado civil
0	Fechada	Desligada	Sem movimento	Solteiro
1	Aberta	Ligada	Com movimento	Casado

- Por exemplo, em um sistema com trava eletrônica, o valor 0 poderia indicar que a porta estava fechada, enquanto 1 indicaria que a porta estaria aberta.

- Para representar mais de dois valores distintos precisamos de uma sequência de *bits* maior. Na Tabela 2, é apresentado exemplos utilizando uma sequência de 2 *bits*, obtendo assim 4 possibilidades.

Tabela 3. Representação com dois *bit*.

Sequência de <i>bits</i>	Semáforo
00	Desligado 
01	Pare 
10	Atenção 
11	Siga 



- Segundo Farias (2013), o número de possibilidades diferentes que podemos representar depende do tamanho da sequência que estamos utilizando, mais precisamente:

$$P = 2^n, \text{ onde } n = \text{tamanho de } \textit{bits}.$$

- Exemplos:

$$2^1 = 2$$

$$2^2 = 4$$

$$2^3 = 8$$

$$2^4 = 16$$

$$2^5 = 32$$

$$2^6 = 64$$

$$2^7 = 128$$

$$2^8 = 256$$

(1 *byte*)

$$16 \text{ bits} = 65.535$$

$$32 \text{ bits} = 4.294.967.295$$

$$64 \text{ bits} = 18.446.744.073.709.551.615$$





## 5. CONVERSÃO NUMÉRICA.

Como o computador pensa e executa.



---

## 5.1. SISTEMA POSICIONAL

- O método de numeração de quantidades que adotamos, utiliza um sistema de numeração posicional.
- Significa que a posição ocupada por cada algarismo em um número altera seu valor de uma potência decimal (base 10) para cada casa à esquerda. Vejamos o exemplo abaixo:

$$125_{10} = \underbrace{1 \times 10^2}_{100} + \underbrace{2 \times 10^1}_{20} + \underbrace{5 \times 10^0}_5$$

Centena      Dezena      Unidade

---

## 5.2. AS BASES [...]

- A base de um sistema é a quantidade de algarismos disponíveis em sua representação.
- A base 10 (decimal) é hoje a mais utilizada, mas não é a única. Por exemplo, temos:
  - ✓ A dúzia (base 12).
  - ✓ O minuto = 60 segundos (base 60).
  - ✓ 1 polegada = 2,54 cm.
- Em computadores usamos outras bases como a binária (base 2), octal (base 8) e hexadecimal (base 16).

- Portanto, temos:

- ✓ Base decimais ( $b_{10}$ ) → 0 1 2 3 4 5 6 7 8 9
- ✓ Base binária ( $b_2$ ) → 0 1
- ✓ Base octal ( $b_8$ ) → 0 1 2 3 4 5 6 7
- ✓ Base hexadecimal ( $b_{16}$ ) → 0 1 2 3 4 5 6 7 8 9 A B C D E F  
(10 algarismos + 6 símbolos)

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

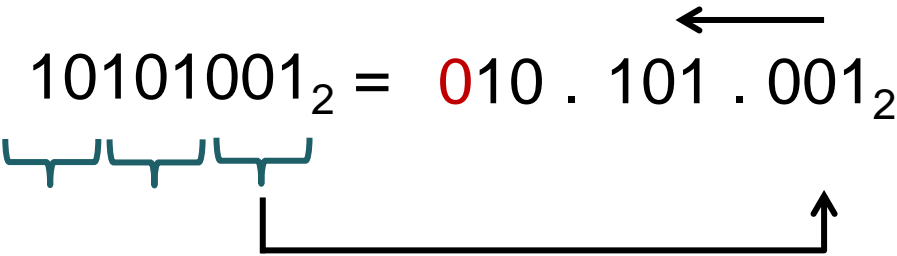


---

## 5.3. CONVERSÃO ENTRE BASES 2, 8 E 16.

- As conversões mais simples são as que envolvem bases que são potências entre si. (FARIAS, 2013).
- Leva-se em consideração que  $2^3 = 8$  (octal) e  $2^4 = 16$  (hexadecimal).
- Exemplifiquemos a conversão  $2^3$ , que funciona da seguinte forma:
  - 1º. Separa-se os algarismos de um número binário em grupos de três (começando sempre da direita para a esquerda).
  - 2º. Converta cada grupo de três algarismos por seu equivalente em octal.
- Vejamos:

$10101001_2 = 010 \cdot 101 \cdot 001_2$



- Olhando a tabela de conversão direta temos:

$010_2 = 2_8 \quad 101_2 = 5_8 \quad 001_2 = 1_8 \quad 251_8$

$10101001_2 = 251_8$

Tabela 4. 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

- Agora a conversão entre as bases 2 e 16.
- Como  $2^4 = 16$ , seguimos o mesmo processo anterior, bastando agora separarmos em grupos com quatro algarismos e converter cada grupo seguindo a Tabela 5. Por exemplo:

$11010101101_2 = \underbrace{0110}_2 . \underbrace{1010}_2 . \underbrace{1101}_2$

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

}

$11010101101_2 = 6AD_{16}$

Tabela 5. Conversão direta de binário para hexadecimal e vice-versa.

Bin.	Hexa.	Bin.	Hexa.
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

---

## 5.4. CONVERSÃO DE BASE $b$ PARA BASE 10

- Lembremos da expressão geral descrita na página 14.

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

- A melhor forma de fazer a conversão é usando essa expressão. Como exemplo usemos o valor  $101101_2$ .

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

- Outros exemplos:

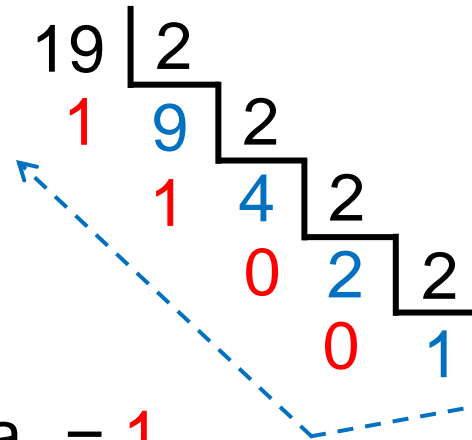
$$125_5 = 1 \times 5^3 + 0 \times 5^2 + 0 \times 5^1 + 0 \times 5^0 = 100_{10}$$

$$485_9 = 4 \times 9^2 + 8 \times 9^1 + 5 \times 9^0 = 324 + 72 + 5 = 401_{10}$$



## 5.5. CONVERSÃO DE BASE 10 PARA BASE b

- Exemplo: Converta o numero  $19_{10}$  para a base 2.



$$19_{10} = 10011_2$$

$$a_4 = 1 \quad a_3 = 0 \quad a_2 = 0 \quad a_1 = 1 \quad a_0 = 1$$

- Usando a conversão anterior como prova real, temos:

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

---

## 5.6. NÚMEROS BINÁRIOS NEGATIVOS

- Segundo Farias (2013), os computadores operam com números positivos (+) e negativos (-), sendo necessário encontrar uma representação para números com sinal negativo.
- Existem uma grande variedade de opções. Apresentemos aqui as três formas mais usuais:
  1. Sinal e amplitude/magnitude (S + M).
  2. Complemento de 1.
  3. Complemento de 2.

---

## 1. Sinal e Amplitude/Magnitude (S + M)

- Como o próprio nome indica, a representação sinal e amplitude utiliza um *bit* para representar o sinal.
- Por exemplo, no *bit* mais à esquerda incluem-se 0 para indicar um valor positivo, 1 para indicar um valor negativo.

$$+10_{10} = 0\ 1010_2$$

$$-10_{10} = 1\ 1010_2$$

---

## 2. Complemento de 1

- Na representação em complemento de 1 invertem-se todos os *bits* de um número para representar o seu complementar.
- Converte-se um valor positivo por um negativo, e vice-versa.
- Quando o *bit* mais à esquerda é 0, esse valor é positivo; se for 1, então é negativo. Por exemplo:

$$100_{10} = 01100100_2 \text{ (com 8 bits)}$$

$$-100_{10} = 10011011_2 \text{ (bits invertidos)}$$



O problema desta representação é que existem 2 padrões de *bits* para o 0, havendo assim desperdício de representação:

$$0_{10} = 00000000_2 = 11111111_2$$



### 3. Complemento de 2

- Para determinar o negativo de um número na forma de complemento de 2, basta inverter todos os *bits* e somar 1 unidade.

1. Representação binária

$101_{10} = 01100101_2$  (com 8 *bits*)



Aritmética binária

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 0 \text{ (sobe 1)}$$

2. Invertendo todos os *bits*

$01100101_2 \rightarrow 10011010_2$

3. Somando 1 unidade

$10011010_2 + 1$



$10011010_2$

+ 1

$10011011_2$

=  $-101_{10}$



---

## EXERCÍCIO

1. Determine o número binário negativo de  $120_{10}$  em 8 *bits* usando a representação de complemento 1.

### Solução:

1. Converte dec. para bin.

$$120_{10} = 01111000_2$$

2. Inverte os bits binários

$$10000111_2 = -120_{10}$$

2. Qual o número representado por  $11100100_2$  (com 8 *bits*)? Este número é negativo ou positivo?

Solução:

1. Verifique se o *bit* mais a esquerda é 0 (para positivo) ou 1 para negativo).

$11100100_2$  Negativo

2. Inverte todos os bits e soma 1 unidade

$$\begin{array}{rcl} 00011011_2 + 1 & \rightarrow & \begin{array}{r} \phantom{000}11 \\ 00011011_2 \\ +1 \\ \hline 00011100_2 = 28_{10} \end{array} \end{array}$$



Aritmética binária

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

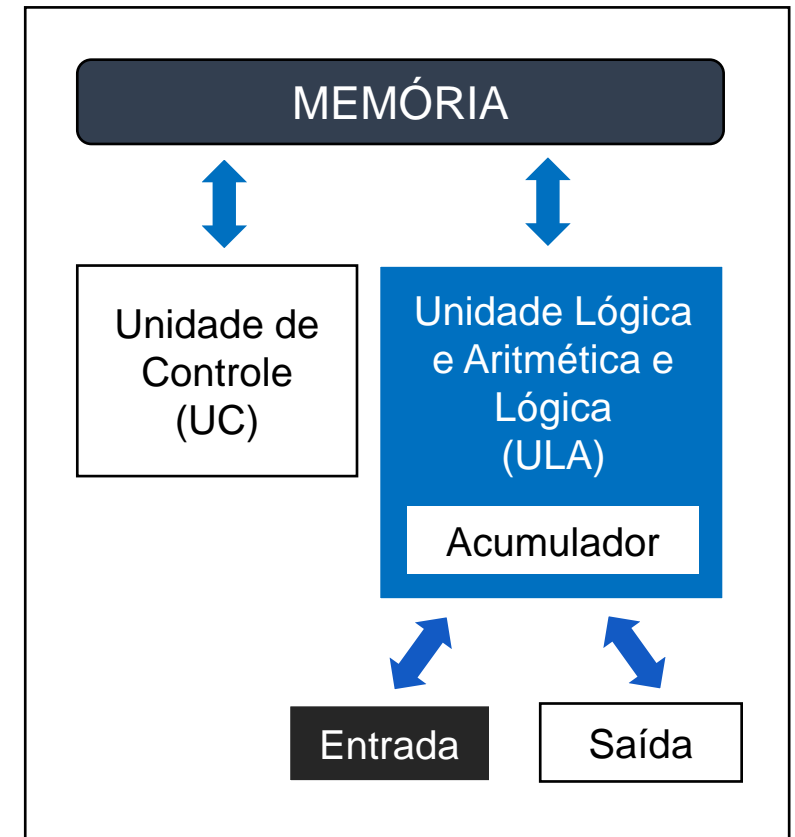
$$1 + 1 = 0 \text{ (sobe 1)}$$

$$11100100_2 = -28_{10}$$

## 5.7. ARITMÉTICA BINÁRIA

- Como o computador manipula dados (números) através de uma representação binária, veremos a partir de agora como a aritmética do sistema binário, a mesma usada pela ULA (Unidade Lógica e Aritmética) dos processadores, processa esses dados.

Figura 2. Arquitetura proposta por John von Neumann (1946).



# Soma e Subtração Binária

Tabela 6. Tabuada de soma binária.

Operação	Valor	Obs.
$0 + 0 =$	0	
$0 + 1 =$	1	
$1 + 0 =$	1	
$1 + 1 =$	0	“vai um”(*)
$1 + 1 + 1 =$	1	“vai um” (*)

Tabela 7. Tabuada de subtração binária.

Operação	Valor	Obs.
$0 - 0 =$	0	
$0 - 1 =$	1	“vem um”(**)
$1 - 0 =$	1	
$1 - 1 =$	0	

(\*) – Vai um para a ordem superior ou seja, sobe 1, como na aritmética tradicional.

(\*\*) – Vem um do próximo




• Exemplo 01: Efetue  $011100_2 + 011010_2$

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

$$\begin{array}{r} \textcolor{blue}{1} \textcolor{blue}{1} \quad \leftarrow \text{“vai um”} \\ 011100_2 \\ + 011010_2 \\ \hline 110110_2 \end{array}$$

• Exemplo 02: Efetue  $11100_2 - 01010_2$

 Como não é possível 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 modo é o mesmo da subtração em decimal.

Subtrai-se a direita para a esquerda.

$$\begin{array}{r} \quad \textcolor{blue}{0} \textcolor{blue}{2} \quad \leftarrow \text{“vem um”} \\ 11\cancel{1}00_2 \\ - 01010_2 \\ \hline 10010_2 \end{array}$$

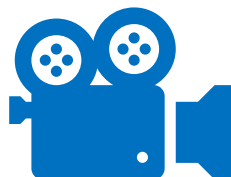
---

## APRENDA MAIS

- Dica bacana [...].



Vídeo sobre vídeo sobre a Representação da informação.



[https://www.youtube.com/watch?v=y\\_eClEibHI](https://www.youtube.com/watch?v=y_eClEibHI)

---

## A subtração nos computadores

- Na eletrônica digital a construção de circuitos simples custa menos e operam mais rápido do que circuitos mais complexos.
- Portanto, os números utilizados na aritmética em Complemento de 2 (pág. 23), permitem a implementação e uso de circuitos mais simples, baratos e rápidos.
- Uma característica é que tanto os números com sinal quanto os números sem sinal podem ser somados pelo mesmo circuito.
- Por exemplo, suponha que você deseje somar os números sem sinal  $132_{10}$  e  $14_{10}$ .

$$\begin{array}{r}
 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0_2 \\
 +\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0_2 \\
 \hline
 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0_2
 \end{array}$$

Entrada: A

Entrada: B

Saída: C

+ 132<sub>10</sub>+ 14<sub>10</sub>+ 146<sub>10</sub>

?

- O microprocessador tem um circuito na ULA que pode somar números binários sem sinal.
- Surge então a pergunta: como a ULA sabe que os padrões de *bits* nas entradas representam número sem sinal e não em complemento de 2?
- A resposta é: **NÃO SABE!** A ULA sempre soma como se as entradas fossem números binários sem sinal. Sempre produzirá o resultado correto, mesmo se as entradas forem números em complemento de 2. É um circuito/instrução corretiva.

- A ULA realiza operações de soma tratando os *bits* como números binários sem sinal, cabendo à nossa interpretação definir se os valores são com ou sem sinal.
- Essa abordagem permite utilizar o mesmo circuito para ambos os casos, otimizando o *design* do *hardware*.
- Além disso, a aritmética de complemento de 2 simplifica a implementação da subtração, eliminando a necessidade de um circuito específico para essa operação. Assim, a CPU pode utilizar o mesmo circuito-somador tanto para soma quanto para subtração, reduzindo a complexidade e os custos do microprocessador.



# Multiplicação binária

Tabela 8. Tabuada multiplicação binária.

Operação	Valor
$0 \times 0 =$	0
$0 \times 1 =$	0
$1 \times 0 =$	0
$1 \times 1 =$	1



O processo é idêntico à multiplicação entre números decimais.

- Exemplo: Efetue  $101_2 \times 110_2$

$$\begin{array}{r} 101_2 \\ \times 110_2 \\ \hline 000 \\ + 101 \\ 101 \\ \hline 11110_2 \end{array}$$

$$\begin{array}{r} 5_{10} \\ \times 6_{10} \\ \hline 30_{10} \end{array} = 11110_2$$



## 6. A LÓGICA BINÁRIA.

Circuitos lógicos e operadores.

---

## 6.1. SISTEMAS DICOTÔMICOS E A ÁLGEBRA DE BOOLE

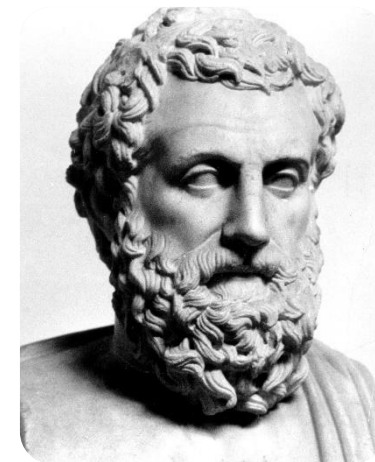
- Segundo Daghlia (2008), o mundo em que vivemos apresenta situações dualísticas em sua grande maioria, ou seja, com dois estados que mutuamente se excluem.

Tabela 9. Situações dualísticas.

Valor 1	Valor 2
1	0
Sim	Não
Dia	Noite
Preto	Branco
Ligado	Desligado

- Existem situações como morno, tépido, diferentes tons de cores que não apresentam estritamente como dicotômicas, i.e., com dois estados excluídos bem definidos.

- Segundo Daghlian (2008) a Lógica começou a se desenvolver no século IV a.C., com Aristóteles.
- Neste período os filósofos gregos passaram a usar em suas discussões sentenças lógicas enunciadas nas formas afirmativas e negativas, resultando assim em grande simplificação da realidade no dia a dia [...]. E quase 2.000 anos depois, por volta de 1666, Leibniz usou em vários trabalhos o que chamou de *Calculus ratiotinator*. Essas ideias nunca foram teorizadas, porém seus escritos trazem a ideia do que seria a Lógica Matemática.



Aristóteles  
(384 – 322 a.C.)



Gottfried W. Leibniz  
(1646 – 1716)



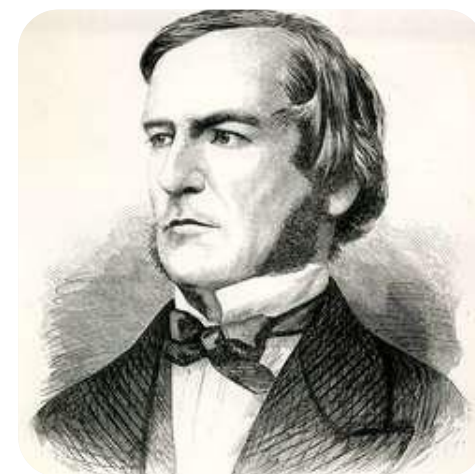


William Rowan Hamilton  
(1805 – 1865)

- Após décadas/séculos de discussões entre filósofos e matemáticos, com críticas por ambas as partes, ao ponto do filósofo William Rowan Hamilton dizer:

A Matemática congela e embota a mente; um excessivo estudo da Matemática incapacita a mente para as energias que a filosofia e a vida requerem.

- Em 1854, George Boole introduz o formalismo que até hoje usamos para o tratamento sistemático da lógica, chamada de Álgebra Booleana.



George Boole  
(1815 – 1864)



- A álgebra booleana pode ser definida como um conjunto de operadores e de axiomas, que são assumidos verdadeiros sem a necessidade de prova. (Güntzel & Nascimento, 2001).
- Em 1938, C. E. Shannon aplicou esta álgebra para mostrar que as propriedades de circuitos elétricos de chaveamento podem ser representadas por uma álgebra booleana com dois valores. (*ibidem*).
- Para Güntzel & Nascimento (2001), diferentemente da álgebra ordinária dos números reais, onde as variáveis podem assumir valores no intervalo  $(-\infty ; +\infty)$ , as variáveis booleanas só podem assumir um número finito de valores.

0 ou 1

- Em particular, na álgebra booleana, cada variável pode assumir um dentre dois valores possíveis. Por exemplo:

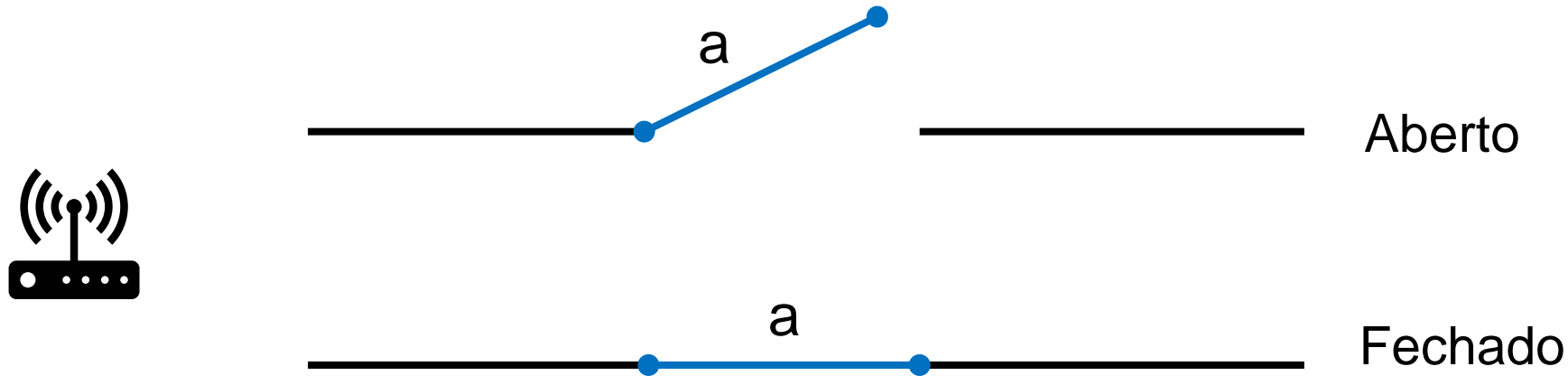
Tabela 10. Exemplo de operadores.

Operador	Valores
F ou V	Falso ou Verdadeiro
C ou E	Certo ou Errado
H ou L	<i>High ou Low</i>
0 ou 1	-

- Portanto são sistemas dicotômicos. Computacionalmente dizendo 0 e 1, a qual é também utilizada na eletrônica digital de circuitos.

## 6.2. INTERRUPTORES

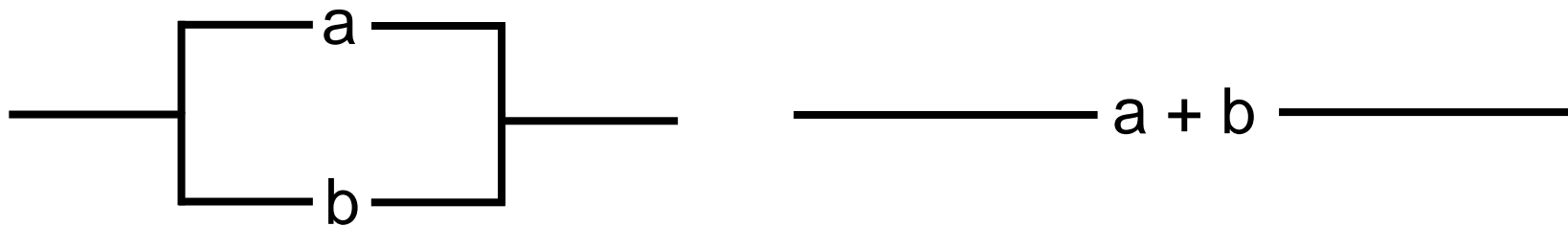
- Chamamos interruptor ao dispositivo ligado a um ponto de um circuito elétrico, o que pode assumir um dos dois estados, e.g., Fechado (1) e Aberto (0). (DAGHLIAN, 2008).
- Quando fechado, o interruptor permite que a corrente passe através do ponto, enquanto aberto nenhuma corrente passa.



- Por conveniência, representaremos os interruptores da seguinte forma (DAGHLIAN, 2008):

\_\_\_\_\_ a \_\_\_\_\_

- Sejam a e b dois interruptores ligados em paralelo, só passará corrente se pelo menos um dos interruptores estiver fechado. Então denotaremos a ligação de dois interruptores em paralelo por  $a + b$ . (*ibidem*). Então:



- Mas se dois interruptores estão ligados em série, só passará corrente se ambos estiverem fechados, i.e.,  $a = b = 1$ . Portanto denotaremos a ligação de dois interruptores  $a$  e  $b$  por  $a \cdot b$  ou simplesmente  $ab$ . (DAGHLIAN, 2008):



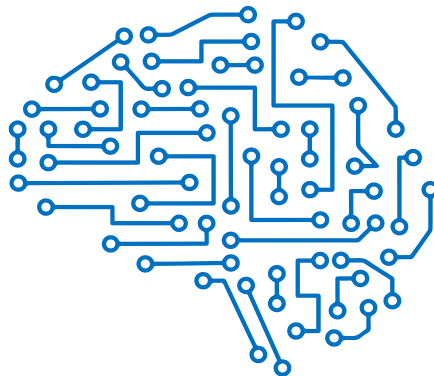
- Então temos:

Paralelo	Série
$0 + 0 = 0$	$0 \cdot 0 = 0$
$0 + 1 = 1$	$0 \cdot 1 = 0$
$1 + 0 = 1$	$1 \cdot 0 = 0$
$1 + 1 = 1$	$1 \cdot 1 = 1$

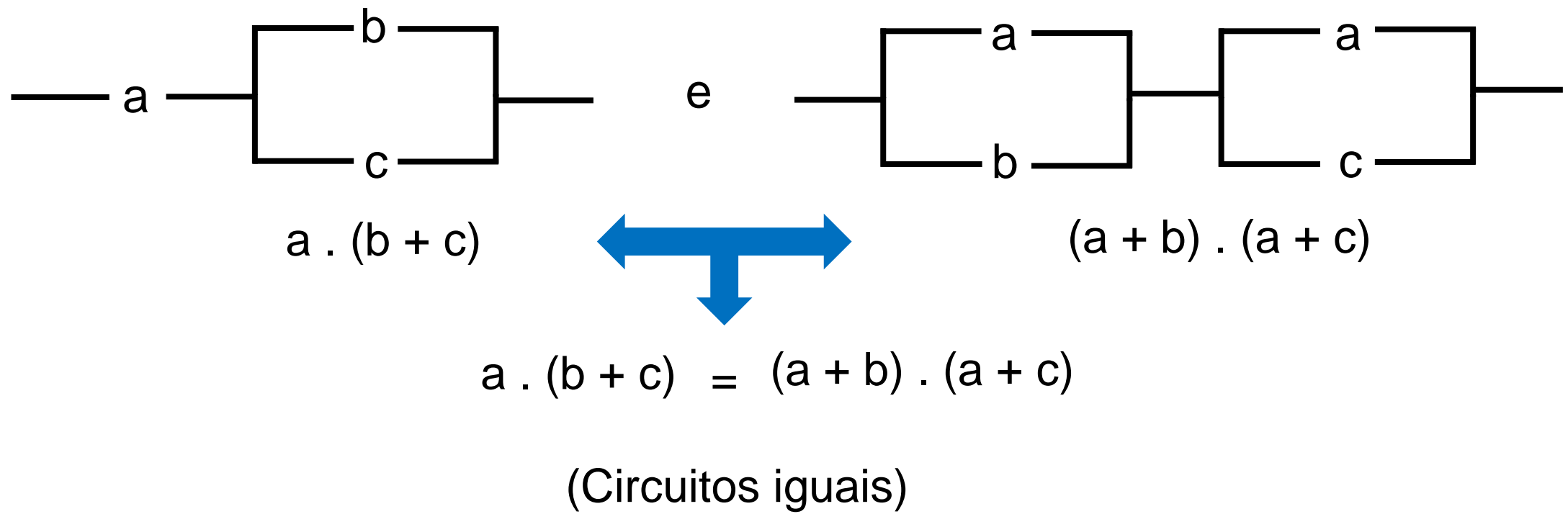
Tabela 11. Possíveis ligações em interruptores.



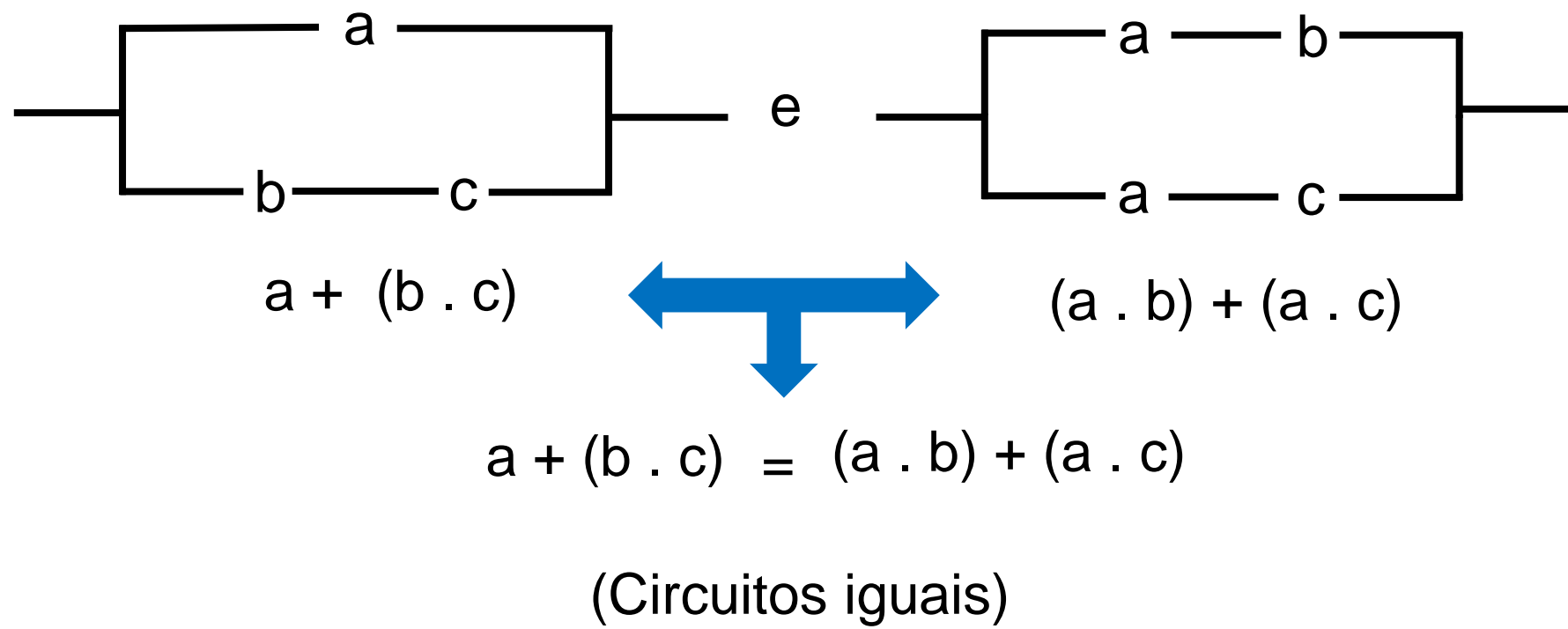
- Observações importantes segundo Daghlian (2008):
  - Conhecendo o estado de um interruptor a, podemos compreender que qualquer outro interruptor tenha o mesmo estado de a, i.e., aberto quando a está aberto e fechado quando a está fechado.
  - Chamamos de complemento quando um interruptor aberto está fechado e vice-versa. Isso se chama de inversão ou negação.
  - Por exemplo:  $a \neq a' \longrightarrow 1 \neq 0$
  - É possível criar inúmeras operações/expressões lineares na lógica digital compreendendo suas ligações nos circuitos apresentados.



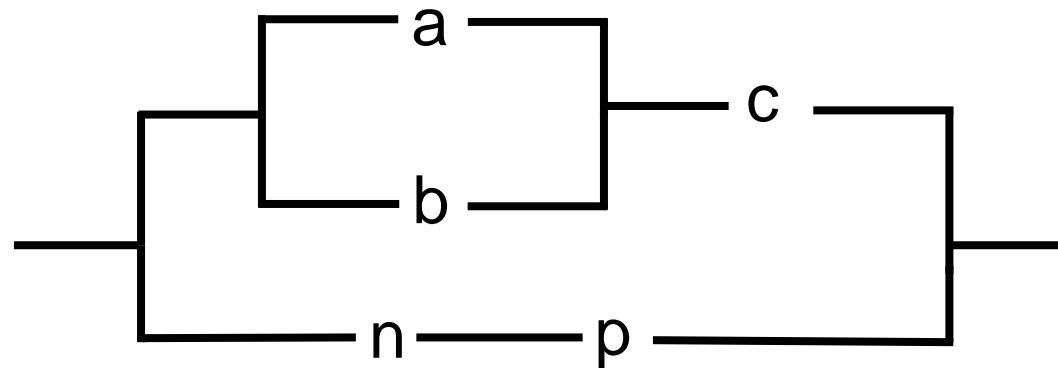
- Apresentamos abaixo algumas equações baseadas em circuitos lógicos.



Outro exemplo:



- Exercício 1. Determine a ligação do seguinte circuito:



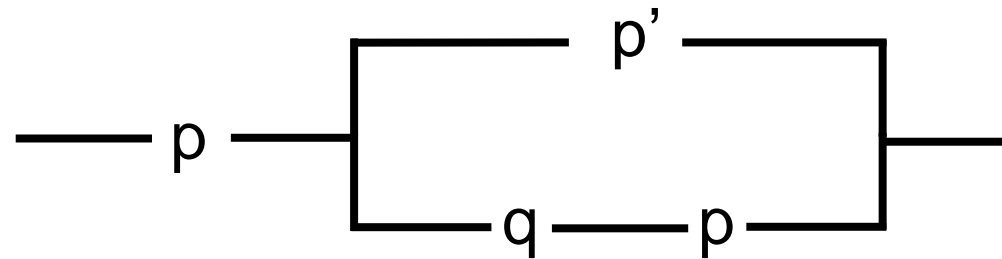
Solução:  $(a + b) \cdot c + (n \cdot p)$

- Exercício 2. Desenhe os circuitos cujas as ligações são:

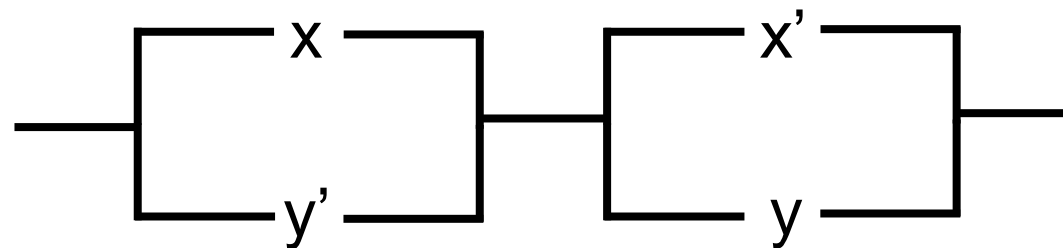
Solução a

a)  $p \cdot (p' + q \cdot p)$

b)  $(x + y') \cdot (x' + y)$



Solução b



---

## 6.3. A LÓGICA BINÁRIA

- George Boole publicou a Álgebra Booleana em 1854, como sendo um sistema completo que permitia a construção de modelos matemáticos para o processamento computacional.
- O interessante é que a partir de três operadores básicos (NOT, AND e OR), podemos construir circuitos lógicos capazes de realizar diversas operações em um computador.
- Como o número de valores que cada variável pode assumir é finito (e pequeno), o número de estados que uma função booleana pode assumir também será finito, o que significa que podemos descrevê-las completamente utilizando tabelas.

- Essa tabela recebe o nome de Tabela Verdade, e nela são listadas todas as combinações de valores que as variáveis de entrada podem assumir e os correspondentes valores da função (saídas).

### 1. Operador NOT (Negação Binária)

No operador unário NOT, inverte o valor do operando, produzindo seu complemento, i.e., o resultado será 1 se o operando for 0 e 0 se o operando for 1. A Tabela 12 ilustra esse comportamento, onde *A* representa o *bit* de entrada e *S* corresponde ao *bit* de saída.

Tabela 12. Tabela verdade operador NOT.

A	S ou A'
0	1
1	0

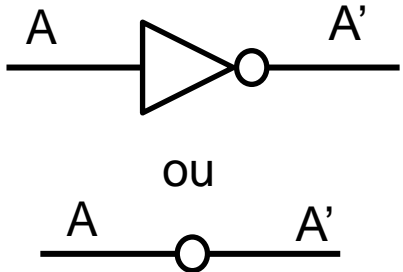


Figura 3. Representação gráfica do operador lógico NOT, com seus valores de entrada e saída.

## 2. Operador AND (Conjunção Binária)

O operador binário AND devolve um *bit* 1 sempre que ambos operandos sejam 1, conforme podemos confirmar na Tabela 13, onde A e B são *bits* de entrada, e S é o *bit*-resposta, ou *bit* de saída.

Tabela 13. Tabela verdade operador AND.

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

$(A \cdot B)$

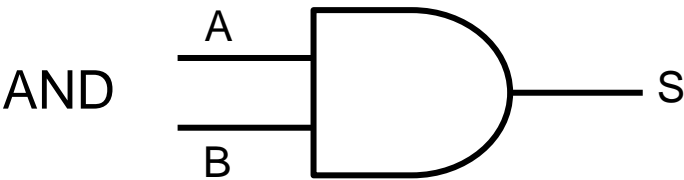


Figura 4. Representação gráfica do operador lógico AND, com seus valores de entrada e saída.



### 3. Operador OR (Disjunção Binária)

O operador binário OR devolve um *bit* 1 sempre que pelo menos um dos operandos seja 1, conforme podemos confirmar na Tabela 14, onde A e B são os *bits* de entrada, e S é o *bit*-resposta, ou *bit* de saída.

Tabela 14. Tabela verdade operador OR.

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

$(A + B)$

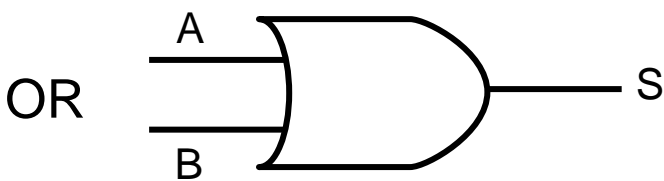


Figura 5. Representação gráfica do operador lógico OR, com seus valores de entrada e saída.

---

## 6.4. A SOMA EM UM COMPUTADOR

- Neste módulo, aprendemos sobre a representação numérica, dando ênfase ao sistema binário.
- Aprendemos como funciona a aritmética binária (soma, subtração, multiplicação, etc.), representação negativa dos números, entre outros.
- Mas como um computador soma?
- Primeiro, 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, resistores, capacitores entre outros que são abordados em cursos avançados da Eletrônica Analógica/Digital.
- O importante é sabermos que existem portas lógicas que seguem a lógica binária já apresentada e que estas portas podem ser combinadas, formando os circuitos digitais.
- A Figura 6 apresenta um circuito digital somador de 2 *bits*.

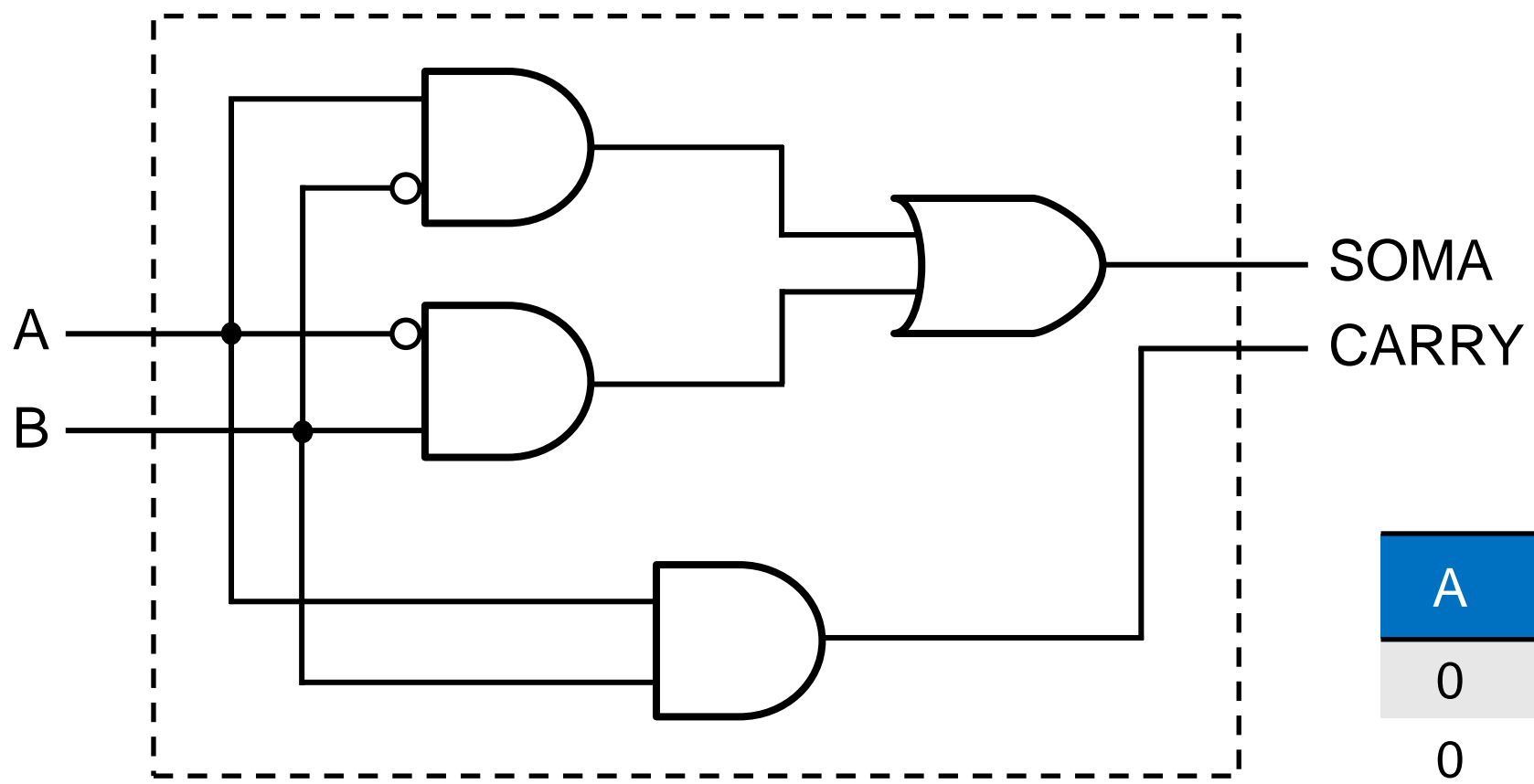


Figura 6. Circuito digital somador de dois *bits* formado pelas portas lógicas básicas (AND, OR, NOT).

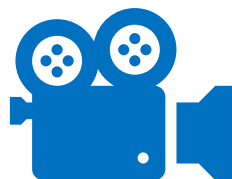
A	B	SOMA	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Tabela 15. Tabela de valores da operação de Soma de 2 *bits*.

- Para entendermos o passo a passo do circuito digital proposto, torna-se necessário criarmos uma tabela verdade para que em caso de dúvidas sobre os valores, revisarmos a operação binária (SOMA) dos dois *bits*, e constatar se a saída corresponde ao esperado, considerando que a saída CARRY é a operação “vai um” da aritmética binária, neste caso a soma.
- E uma dica bem legal. Vejam o vídeo abaixo:



Circuito digital somador de 2 *bits*:



<https://www.youtube.com/watch?v=E5yDNF2clQw>

---

## REFERÊNCIAS

DAGHLIAN, Jacob. Lógica e Álgebra de Boole. 4ª ed. 12ª reimpr. São Paulo : Atlas, 2008.

FARIAS, Gilberto. Introdução à computação. UFPB, 2013.

GÜNTZEL, José Luís; NASCIMENTO, Francisco de Assis. Introdução aos Sistemas Digitais (v.2001/1). Disponível in: <https://www.inf.ufsc.br/~j.guntzel/isd/isd.html>.

TANENBAUM, Andrew S. Sistemas Operacionais Modernos. 3ª ed. Pearson, 2010.

TANENBAUM, Andrew S. AUSTIN, Todd. Organização Estruturada de Computadores. 6ª ed. Pearson, 2013.

MONTEIRO, Mário A. Introdução à Organização dos Computadores. 5ª ed. LTC. 2014.

TANENBAUM, Andrew S. VAN STEEN, Maarten. Sistemas Distribuídos. Princípios e Paradigmas. 2ª ed. Pearson, 2007.

TEDESCO, Kennedy. Bits, bytes e unidades de medida. Disponível in: <http://twixar.me/Fgjm> . Acessado em: 10.mar.2021.

RIBEIRO, Carlos; DELGADO, José. Arquitetura de Computadores. 2ª ed. Rio de Janeiro: LTC, 2009.



# Obrigado!

Disciplina: Introdução à Computação (I.C)