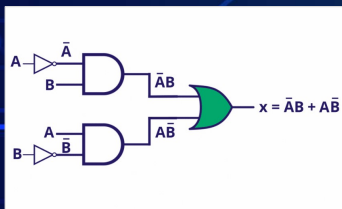
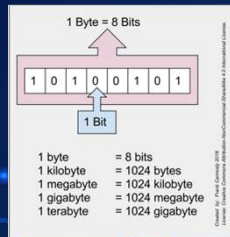


## IC – Introdução a Computação



# Sistema Binário

## Operações com binários

### Soma de Binários

A soma de números binários segue as mesmas regras da soma decimal, mas usando apenas os dígitos 0 e 1.

Mas antes vamos relembrar como fazemos com um número decimal:

$$\begin{array}{r} \textcircled{1} \longrightarrow \text{Vai 1 no popular} \\ 376 \\ + 461 \\ \hline 837 \end{array}$$

# Sistema Binário

## Operações com binários

### Soma de Binários

Para os binários faremos do mesmo jeito que fazemos no decimal, seguindo a regra abaixo:

Entradas		Saídas	
A	B	Soma	Vai 1
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

**Carry:** é um bit que é transferido para a próxima porta ou casa binária à esquerda.

# Sistema Binário

## Operações com binários

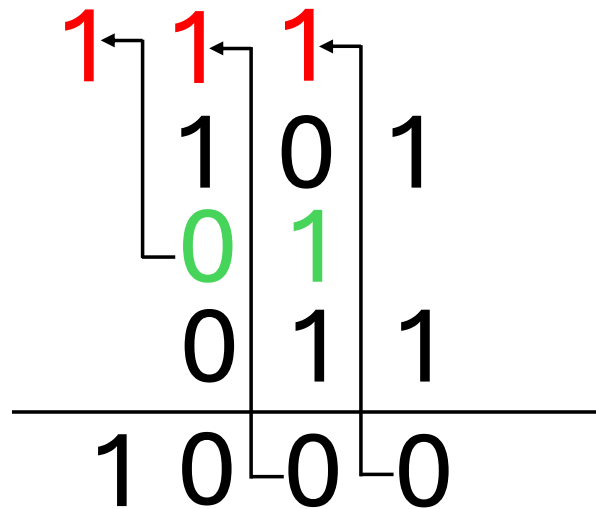
## Soma de Binários

Vamos fazer um exemplo:  $101 + 011$

Nossa cola:

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$

$1 + 1 = 0$  (carry =1)



Resultado: 1000

# Sistema Binário

## Operações com binários

### Multiplicação de Binários

- Mesmo método que o decimal: deslocamentos e adições.
- Número maior deve ser colocado acima do menor.

Vamos fazer um exemplo:  $101 + 011$

#### Regra

$$0 \times 0 = 0$$

$$0 \times 1 = 0$$

$$1 \times 0 = 0$$

$$1 \times 1 = 1$$

$$\begin{array}{r} 101 \\ \times 011 \\ \hline 101 \\ 101 \\ + 000 \\ \hline 01111 \end{array}$$

Produto/Multiplicação:



01111

# Sistema Binário

## Operações com binários

### Subtração de Binários

Vamos ao exemplo:  $110 - 11$

Entradas		Saídas	
A	B	Subtração	Vem -1
0	0	0	0
0	1	1	-1
1	0	1	0
1	1	0	0

#### Regra

$$0 - 0 = 0$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

$$0 - 1 = 1 \text{ (vem -1)}$$

$$\begin{array}{r}
 \textcolor{red}{-1} \textcolor{red}{-1} \\
 \begin{array}{r}
 110 \\
 -011 \\
 \hline
 011
 \end{array}
 \end{array}$$

Resultado: 011

# Sistema Binário

## Exercícios

### 1. Soma de números binários

Some os seguintes números binários e forneça a resposta em binário.

a)  $101 + 11 = ?$

a)  $101 + 11 = 1000$

b)  $1101 + 101 = ?$

b)  $1101 + 101 = 10010$

### 2. Subtração de números binários

Realize as seguintes subtrações em binário:

a)  $1010 - 11 = ?$

a)  $1010 - 11 = 111$

b)  $1100 - 101 = ?$

b)  $1100 - 101 = 111$

### 3. Multiplicação de números binários

Realize as seguintes multiplicações em binário:

a)  $101 \times 10 = ?$

a)  $101 \times 10 = 1010$

b)  $110 \times 11 = ?$

b)  $110 \times 11 = 10010$

Nossa cola Soma:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

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

Regra Subtração

$$0 - 0 = 0$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

$$0 - 1 = 1 \text{ (vem -1)}$$

Regra X

$$0 \times 0 = 0$$

$$0 \times 1 = 0$$

$$1 \times 0 = 0$$

$$1 \times 1 = 1$$





# Sistema Binário

## Bit

Qualquer dígito do Sistema Binário em geral é denominado bit, devido a contração das palavras binary digit, que vem inglês.

- Um bit é a menor quantidade de informação que pode ser processada e armazenada na memória do computador.
- É o sinal elétrico que representa a menor unidade de informação que o computador pode entender.
- Assume somente dois valores (0 ou 1).

Os binários, quando agrupados em determinado número de dígitos, constituindo-se numa informação completa, recebe designações especiais a saber:

- BIT = um dígito binário 0 ou 1
- NIBBLE = um grupo de 4 bits
- BYTE = um grupo de 8 bits
- WORD = um grupo de 2 bytes ou 4 nibbles, ou múltiplos desses.

# Sistema Binário

## Byte

- Com a definição de bit, fica muito fácil entender o que é um byte (Binary Term).
- O byte é composto por 8 (oito) bits, e é o necessário para representar qualquer caractere (letras, números, sinais de pontuação).
- Então, para representar a letra "a", por exemplo, é necessário 1 byte, ou seja, 8 bits.
- Se um byte tem oito bits, existem 256 combinações possíveis de bytes.
- Dois bytes, ou 16 bits, podem ter 65.535 combinações diferentes.

8 Bits = 1 byte = 1 character.

0	1	1	1	0	1	1	0
---	---	---	---	---	---	---	---

Por que preciso saber disso???

Tudo na Computação é medido com bits e bytes!!

# Sistema Binário

Uma tabela com valores convertidos.

DECIMAL	BINÁRIO	OCTAL	HEXADECIMAL
0	00000	0	0
1	00001	1	1
2	00010	2	2
3	00011	3	3
4	00100	4	4
5	00101	5	5
6	00110	6	6
7	00111	7	7
8	01000	10	8
9	01001	11	9
10	01010	12	A
11	01011	13	B
12	01100	14	C
13	01101	15	D
14	01110	16	E
15	01111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17
24	11000	30	18
25	11001	31	19

26	11010	32	1A
27	11011	33	1B
28	11100	34	1C
29	11101	35	1D
30	11110	36	1E
31	11111	37	1F
32	100000	40	20
50	110010	62	32
60	111100	74	3C
64	1000000	100	40
100	1100100	144	64
255	11111111	377	FF
1000	1111101000	1750	3E8
1024	1000000000	2000	400

# Sistema Binário

## Códigos Binários

- Para representar os binários nem sempre usaremos sua equivalência direta com os números decimais, octais ou hexadecimais.
- Sempre poderemos considerar por razões de praticidade, aumento de velocidade das operações lógicas e simplificação dos circuitos digitais que executam operações lógicas, sobretudo no que diz respeito a circuitos aritméticos de computadores, utilizar da conveniência de expressar determinadas informações de uma forma codificada, sem que a mesma guarde qualquer relação com o valor numérico, propriamente dito, que se encontra simbolizado por um determinado número em binário.
- Existem diversos códigos que podem ser utilizados.
  - Código binário natural (8 4 2 1 )
  - BCD (Binary Coded Decimal)
  - Código de Gray
  - Código Excesso de 3 (XS3)
  - Código ASCII
  - Códigos de paridade, detecção e correção de erros, etc.

# Sistema Binário

## Códigos Binários

- Em geral nos utilizaremos para nosso curso os códigos BCD (8421) e Gray
- BCD (Binary Coded Decimal)
- Código de Gray

O código BCD (8421) representa os decimais de 0 a 9 sob a forma de 4 bits, basta convertermos dígito a dígito de decimal para binário.

Exemplo:

572  $\Rightarrow$  0101 0111 0010<sub>BCD</sub>

572  $\Rightarrow$  (1000111100)<sub>2</sub>

Atenção, BCD não é um novo sistema numérico! Basicamente é só um sistema em que cada dígito está escrito em binário.

Além disso BCD é diferente do número binário puro.

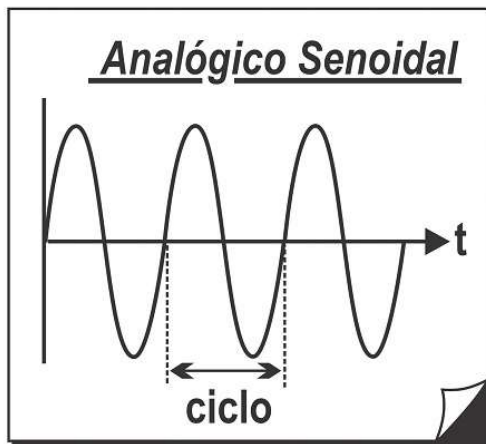
No nosso exemplo para BCD são necessários 12 bits enquanto no binário apenas 10 bits.

# Eletrônica Digital e Analógica

Qual a diferença?

A eletrônica é um campo bem vasto e muito diversificado. Ela basicamente se divide em duas categorias principais: Analógica e Digital

Na eletrônica Analógica temos os sinais contínuos que podem assumir qualquer valor de um intervalo específico. Representamos esses sinais com ondas senoidais ou ondas contínuas.



Fonte: <https://oficinabrasil.com.br/noticia/tecnicas/sinais-eletricos-transportam-informacoes-geradas-pelos-sensores-para-serem-processadas>

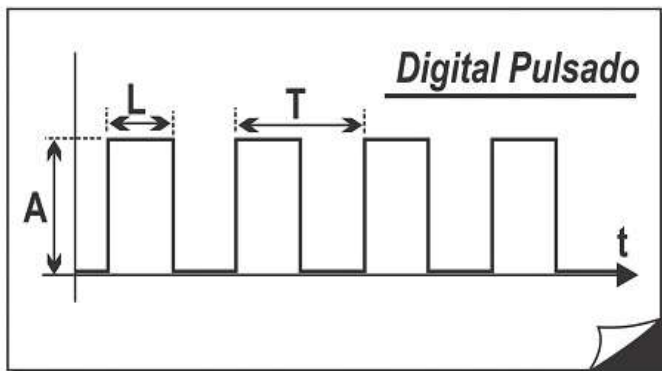
Seus principais componentes são:

Resistores, capacitores, indutores, diodos e transistores.

# Eletrônica Digital e Analógica

Qual a diferença?

Na eletrônica digital os sinais são chamados de sinais discretos e só podem assumir os valores distintos 0 e 1. Esses sinais são representados por pulsos de tensão, onde o 0 significa baixa tensão e o 1 alta tensão.



Fonte: <https://oficinabrasil.com.br/noticia/tecnicas/sinais-eletricos-transportam-informacoes-geradas-pelos-sensores-para-serem-processadas>

Seus principais componentes são:

Portas lógicas, flip-flops, contadores, registradores, microcontroladores e processadores.

Vantagens

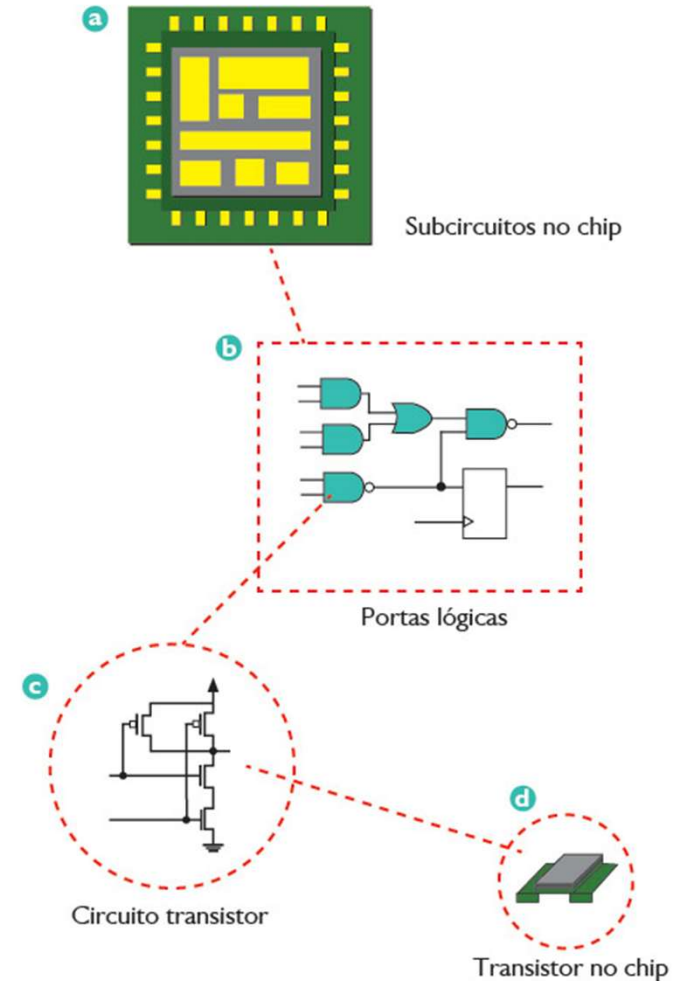
- Facilidade no projeto
- Armazenamento fácil
- Imunidade a ruídos eletromagnéticos
- Precisão
- Não tem ruído
- Não tem variação de medição
- Não tem distorção

# Portas Lógicas

Agora vamos começar a estudar os circuitos lógicos, os mais básicos. Eles se chamam portas lógicas, que na verdade são fundamentais para implementação de nossos circuitos lógicos.

Nós iremos agora trabalhar com os estados 0 e 1. Valores lógicos.

Os circuitos são formados por sub circuitos (a), esses sub circuitos são formados por um conjunto de portas lógicas conectadas (b), que, por sua vez, são constituídas internamente por um conjunto de transistores (d) interligados (c) em cada uma delas.





# Portas Lógicas

## Tabela Verdade

Todo circuito lógico tem uma relação entre as suas entradas e saídas. Essa relação depende do nível lógico na sua entrada

Utilizaremos a relação de  $2^n$  para uma tabela verdade de  $n$  variáveis de entrada.

Para uma tabela de 2 entradas teremos 4 saídas.

Entradas		Saída
A	B	X
0	0	1
0	1	0
1	0	1
1	1	0

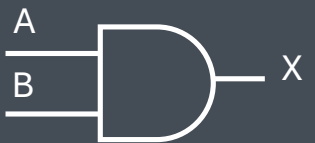
Exemplo de Tabela Verdade



# Portas Lógicas

## Portas Lógicas

E (AND)		
A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1

A função lógica “AND” de duas entradas realiza a seguinte operação de dependência.

$$X = f(A,B) = A.B = B.A \text{ (produto lógico)}$$

### RESUMO:

Quando as entradas da porta AND, A e B, forem iguais a 0, a saída X será igual a 0. Com isso, o X assumirá o valor 1, apenas se todas as entradas forem 1.

Para uma função AND com mais de duas variáveis na entrada.

$$X = A.B.C = B.A.C = C.A.B = (A.B).C = A.(B.C)$$

Comutatividade

Associatividade

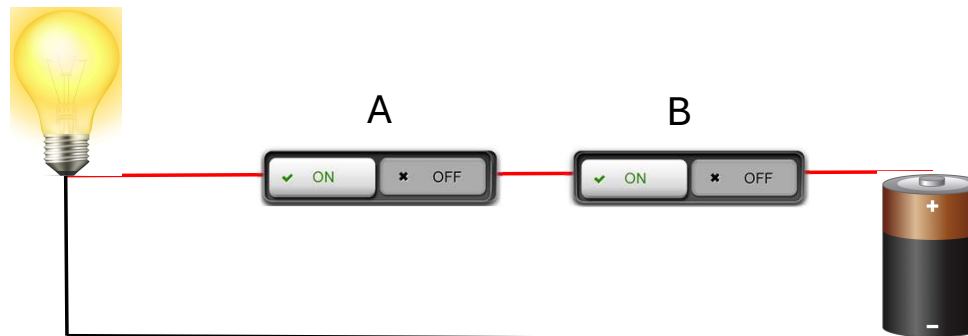
(propriedades aritméticas...)

# Portas Lógicas

## Portas Lógicas

E (AND)

A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1



# Portas Lógicas

## Portas Lógicas

OU (OR)		
A	B	OR
0	0	0
0	1	1
1	0	1
1	1	1


A função lógica OR de duas variáveis realiza a seguinte operação de dependência:

$$X = f(A,B) = A+B \text{ (soma lógica)}$$

### RESUMO:

Quando as entradas da porta OR, A e B, forem iguais a 1, a saída X será igual a 1. Com isso, o X assumirá o valor 0, apenas se todas as entradas forem 0.

Para uma função OR de mais de duas variáveis de entrada

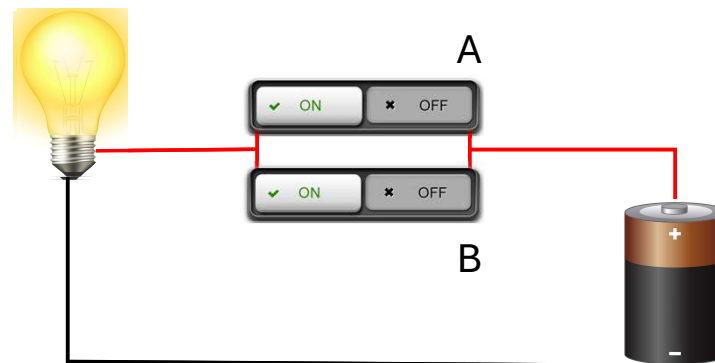
$$X = \underbrace{A+B+C = C+B+A = B+C+A}_{\text{Comutatividade}} = \underbrace{A+(B+C) = (A+B)+C}_{\text{Associatividade}}$$

# Portas Lógicas

## Portas Lógicas

OU (OR)

A	B	OR
0	0	0
0	1	1
1	0	1
1	1	1



# Portas Lógicas

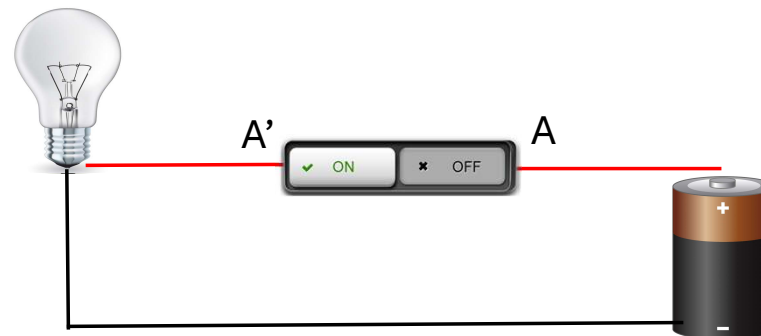
## Portas Lógicas

NÃO (NOT)

A	NOT
0	1
1	0



Inverte a variável aplicada em sua entrada.



# Portas Lógicas

## Portas Lógicas

### NÃO E (NAND)

A	B	AND
0	0	1
0	1	1
1	0	1
1	1	0

Como sugere o nome da função é uma combinação das funções AND e INVERSOR, onde é realizada a função E invertida.

$$X = f(A,B) = \overline{A \cdot B}$$



### NÃO OU (NOR)

A	B	OR
0	0	1
0	1	0
1	0	0
1	1	0






Como sugere o nome da função é uma combinação das funções OR e INVERSOR, onde é realizada a função OU invertida.

$$X = f(A,B) = \overline{A + B}$$



# Portas Lógicas

## Portas Lógicas

BLOCOS LÓGICOS BÁSICOS																			
PORTA	Símbolo Usual	Tabela da Verdade	Função Lógica	Expressão															
E AND		<table><tr><th>A</th><th>B</th><th>S</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	S	0	0	0	0	1	0	1	0	0	1	1	1	Função E: Assume 1 quando todas as variáveis forem 1 e 0 nos outros casos.	$S=A.B$
A	B	S																	
0	0	0																	
0	1	0																	
1	0	0																	
1	1	1																	
OU OR		<table><tr><th>A</th><th>B</th><th>S</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	S	0	0	0	0	1	1	1	0	1	1	1	1	Função E: Assume 0 quando todas as variáveis forem 0 e 1 nos outros casos.	$S=A+B$
A	B	S																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	1																	
NÃO NOT		<table><tr><th>A</th><th>S</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	S	0	1	1	0	Função NÃO: Inverte a variável aplicada à sua entrada.	$S=\overline{A}$									
A	S																		
0	1																		
1	0																		
NE NAND		<table><tr><th>A</th><th>B</th><th>S</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	S	0	0	1	0	1	1	1	0	1	1	1	0	Função NE: Inverso da função E.	$S=\overline{(A.B)}$
A	B	S																	
0	0	1																	
0	1	1																	
1	0	1																	
1	1	0																	
NOU NOR		<table><tr><th>A</th><th>B</th><th>S</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	S	0	0	1	0	1	0	1	0	0	1	1	0	Função NOU: Inverso da função OU.	$S=\overline{(A+B)}$
A	B	S																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	0																	



# Portas Lógicas



## Exercícios

Esboce o circuito das expressões abaixo:

a)  $S = (A+B).C.(B+D)$

b)  $S = (A + B' + C). (A + C + D')'$

Representar portas NOR e NAND com mais de duas entradas (símbolo, função e tabela da verdade).

# Portas Lógicas



Próxima Aula.

Começaremos os circuitos lógicos complexos e a construção dos circuitos de soma, subtração e multiplicação binária.

# Dúvidas?



**ATÉ A  
PRÓXIMA  
AULA!**

