

# Arquitetura e Organização de Computadores

Prof. Dr. Fernando Vernal Salina

# Sistema de Numeração

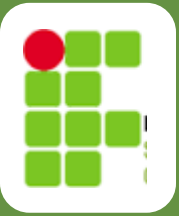
“O que sabemos é uma gota; o que ignoramos é um oceano”

Isaac Newton



# Sistema de Numeração

- O número é um conceito abstrato que representa a idéia de quantidade.
- Sistema de numeração é o conjunto de símbolos utilizados para a representação de quantidades e as regras que definem a forma de representação.
  - Não posicional
  - Posicional



# Sistemas de Numeração Não Posicional

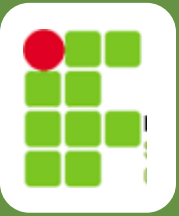
O valor de cada símbolo é determinado de acordo com a sua posição no número.

Exemplo: sistema de algarismos romanos.

Símbolos: I, V, X, L, C, D, M.

## Regras:

- Cada símbolo colocado à direita de um maior é adicionado a este.
- Cada símbolo colocado à esquerda de um maior tem o seu valor subtraído do maior.



# Sistemas de Numeração Posicional

O valor de cada símbolo é determinado de acordo com a sua posição no número.

Um sistema de numeração é determinado fundamentalmente pela BASE, que indica a quantidade de símbolos e o valor de cada símbolo.

Do ponto de vista numérico, o homem lida com o **Sistema Decimal**.



# Sistemas Decimal

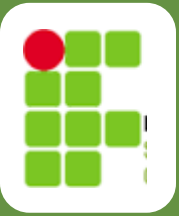
- Base: 10 (quantidade de símbolos).
- Elementos: 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9.

Embora o Sistema Decimal possua somente dez símbolos, qualquer número acima disso pode ser expresso usando o sistema de peso por posicionamento, conforme o exemplo a seguir:

$$3 \times 10^3 + 5 \times 10^2 + 4 \times 10^1 + 6 \times 10^0$$

$$3000 + 500 + 40 + 6 = 3546$$

Obs.: Dependendo do posicionamento, o dígito terá peso. Quanto mais próximo da extrema esquerda do número estiver o dígito, maior será a potência de dez que estará multiplicando o mesmo, ou seja, mais significativo será o dígito.



# Sistemas Binário

É o sistema de numeração mais utilizado em processamento de dados digitais, pois utiliza apenas dos algarismos ( 0 e 1 ), sendo portanto mais fácil de ser representado por circuitos eletrônicos (os dígitos binários podem ser representados pela presença ou não de tensão).

- Base: 2. (quantidade de símbolos)
- Elementos: 0 e 1.

Os dígitos binários chamam-se **BITS** (Binary Digit). Assim como no sistema decimal, dependendo do posicionamento, o algarismo ou bit terá um peso. O da extrema esquerda será o **bit mais significativo** e o da extrema direita será o **bit menos significativo**.

O Conjunto de 8 bits é denominado **Byte**.



# Sistema Octal

O Sistema Octal foi criado com o propósito de minimizar a representação de um número binário e facilitar a manipulação humana.

- Base: 8. (quantidade de símbolos)
- Elementos: 0, 1, 2, 3, 4, 5, 6 e 7.

O Sistema Octal (base 8) é formado por oito símbolos ou dígitos, para representação de qualquer dígito em octal, necessitamos de três dígitos binários.

Os números octais têm, portanto, um terço do comprimento de um número binário e fornecem a mesma informação.





# Sistemas Hexadecimal

- Base: 16. (quantidade de símbolos)
- Elementos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E e F.

O Sistema Hexadecimal ( base 16 ) fo criado com o mesmo propósito do Sistema Octal, o de minimizar a representação de um número binário.

Se considerarmos quatro dígitos binários, ou seja, quatro bits, o maior número que se pode expressar com esses quatro bits é 1111, que é, em decimal 15. Como não existem símbolos dentro do sistema arábico, que possam representar os números decimais entre 10 e 15, sem repetir os símbolos anteriores, foram usados símbolos literais: A, B, C, D, E e F.



# Conversões Entre os Sistemas de Numeração

## Teorema Fundamental da Numeração

Relaciona uma quantidade expressa em um sistema de numeração qualquer com a mesma quantidade no sistema decimal

$$N = d_{n-1} \times b^{n-1} + \dots + d_1 \times b^1 + d_0 \times b^0 + d_{-1} \times b^{-1} + d_{-2} \times b^{-2} + \dots$$

Onde:

***d*** é o dígito,  
***n*** é a posição e  
***b*** é a base.



# Exemplos

$$128_{(\text{base}10)} = 1 \times 10^2 + 2 \times 10^1 + 8 \times 10^0$$

$$54347_{(\text{base}10)} = 5 \times 10^4 + 4 \times 10^3 + 3 \times 10^2 + 4 \times 10_1 + 7 \times 10_0$$

$$100_{(\text{base}2)} = 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 4$$

$$101_{(\text{base}2)} = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 5$$

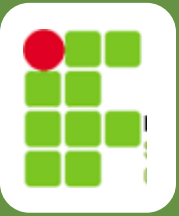
$$24_{(\text{base}8)} = 2 \times 8^1 + 4 \times 8^0 = 16 + 4 = 20$$

$$16_{(\text{base}8)} = 1 \times 8^1 + 6 \times 8^0 = 8 + 6 = 14$$



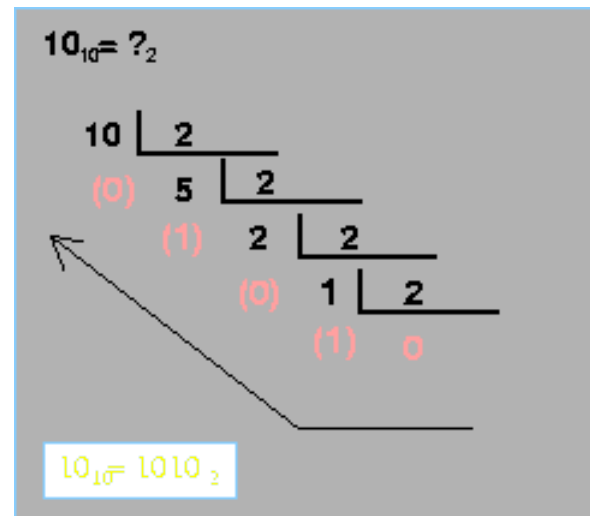
# Tabela de conversão de números

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



# Conversão Decimal-Binário

- Dividir sucessivamente por 2 o número decimal e os quocientes que vão sendo obtidos, até que o quociente de uma das divisões seja 0.
- O resultado é a seqüência de baixo para cima de todos os restos obtidos.





# Conversão Binário-Decimal

- Aplica-se Teorema Fundamental da Numeração

$$101011_2 = ?$$

$$1x2^5 + 0x2^4 + 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 = \\ 32 + 8 + 2 + 1 = 43_{10}$$

$$101011_2 = 43_{10}$$



# Conversão Decimal-Octal

- Divisões sucessivas por 8.
- Multiplicações sucessivas por 8 (parte fracionária).

The image shows a handwritten calculation for converting the decimal number 500 to octal. It uses a series of division steps, with the remainders (4), (6), (7), and 0 written in red and enclosed in parentheses. An arrow points from the final remainder 0 up to the first remainder (4), indicating the correct order for reading the digits. A small box on the right contains the final result:  $500_{10} = 764_8$ .

$$\begin{array}{r} 500_{10} = ?_8 \\ 500 \overline{) 8} \\ (4) \quad 62 \overline{) 8} \\ (6) \quad 7 \overline{) 8} \\ (7) \quad 0 \end{array}$$

$500_{10} = 764_8$

- O resultado é a sequência de baixo para cima de todos os restos obtidos.



# Conversão Octal-Decimal

- Aplica-se Teorema Fundamental da Numeração

$$764_8 = ?_{10}$$

$$7 \times 8^2 + 6 \times 8^1 + 4 \times 8^0 = 500_{10}$$

$$764_8 = 500_{10}$$





# Conversão Decimal-Hexa

- Divisões sucessivas por 16.
- Multiplicações sucessivas por 16 (parte fracionária).

$1000_{10} = ?_{16}$

1000		16	
(8)	62		16
(14)	3		16
(3)			0

→

$1000_{10} = 3E8_{16}$



# Conversão Hexa-Decimal

- Aplica-se Teorema Fundamental da Numeração

$$3E8_{16} = ?_{10}$$

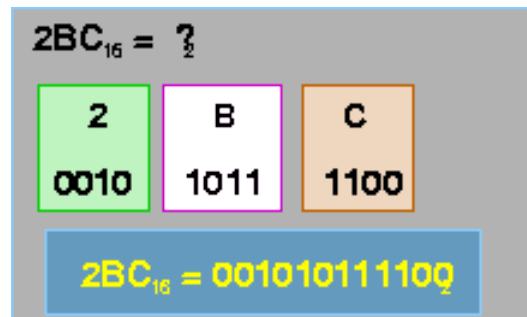
$$3 \times 16^2 + 14 \times 16^1 + 8 \times 16^0 = 1000_{10}$$

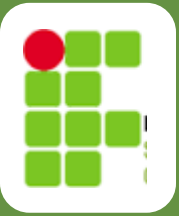
$$3E8_{16} = 1000_{10}$$



# Conversão Hexa-Binário

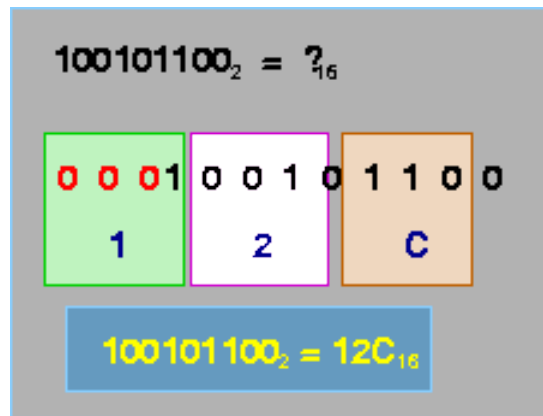
- Agrupamento de 4 bits.
- Usar a tabela (Tabela).





# Conversão Binário-Hexa

- Usar a tabela (Tabela)





# Conversão Octal-Binário

- Agrupamento de 3 bits.
- Usar a tabela (Tabela)

$123_8 = ?_2$

1	2	3
001	010	011

$123_8 = 001010011_2$



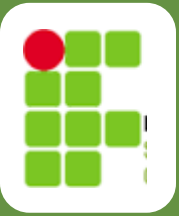
# Conversão Binário-Octal

- Usar a tabela (Tabela)

$10101100_2 = ?_8$

0	1	0	1	0	1	1	0	0
2			5			4		

$10101100_2 = 254_8$



# Conversão Octal-Hexa

- Dois passos:
  - ✓ Converter octal para binário.
  - ✓ Converter binário para hexa.



# Conversão Hexa-Octal

- Dois passos:
  - ✓ Converter hexa para binário.
  - ✓ Converter binário para octal





# Exercícios

- Converter os números abaixo para binário
  - $(128)_{10}$
  - FAH
  - $(17)_8$
- Converta os números abaixo para hexadecimal
  - $(1011101)_2$
  - $(126)_{10}$
  - $(127)_8$



# Exercícios

- Converta os números abaixo para octal
  - FAH
  - $(10111)_2$
  - $(128)_{10}$

# Conceitos de Álgebra Booleana



# Álgebra Booleana

- Em 1854 o matemático inglês George Boole apresentou um **sistema matemático de análise lógica** conhecido como **álgebra de Boole**.
- Somente em 1938, um engenheiro americano utilizou as teorias da álgebra de Boole para a **solução de problemas de circuitos de telefonia com relés**, tendo publicado um artigo que praticamente introduziu na área tecnológica o campo da eletrônica digital.



# Álgebra Booleana

- Os sistemas digitais são formados **por circuitos lógicos denominados de portas lógicas** que, utilizados de forma conveniente, podem implementar todas as expressões geradas pela álgebra de Boole.
- Existem três portas básicas (**E, OU e NÃO**) que podem ser **conectadas de várias maneiras, formando sistemas que vão de simples relógios digitais aos computadores de grande porte.**

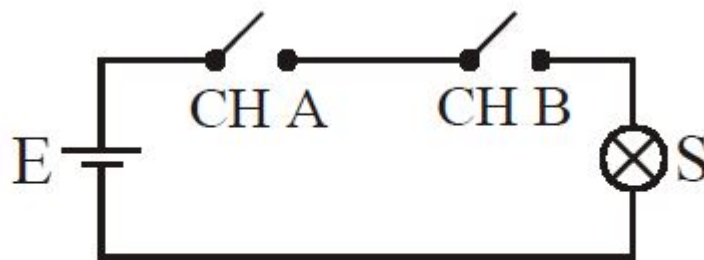


# Função E ou AND

- A função **E é aquela que executa a multiplicação de duas ou mais variáveis booleanas**. Sua representação algébrica para duas variáveis é  $S=A.B$ , onde se lê: S=A e B.
- Para compreender a função E da álgebra Booleana, deve-se analisar o circuito da Fig. 2.1, para o qual se adota as seguintes convenções:
- chave aberta=0, chave fechada=1,
- lâmpada apagada=0 e lâmpada acesa=1.



# Função E ou AND



*Figura 2.1 – Circuito representativo da função **E**.*

A análise da Fig. 2.1 revela que a lâmpada somente acenderá se ambas as chaves estiverem fechadas e, seguindo a convenção, tem-se: CH A=1, CH B=1, resulta em S=1.



# Função E ou AND

- Pode-se, desta forma, escrever todas as possíveis combinações de operação das chaves na chamada **Tabela da Verdade, que é definida como um mapa onde se depositam todas as possíveis situações com seus respectivos resultados**. O número de combinações possíveis é igual a  $2^N$ , onde  $N$  é o número de variáveis de entrada.

*Tabela da verdade da função E.*

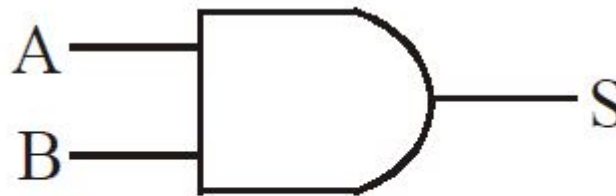
A	B	S
0	0	0
0	1	0
1	0	0
1	1	1





# Função E ou AND

- A porta lógica E é um circuito que executa a função E da álgebra de Boole, sendo representada, na prática, através do símbolo visto na Fig. 2.2.



*Figura 2.2 – Porta lógica E.*

- “A saída da porta E será 1, somente se todas as entradas forem 1”.



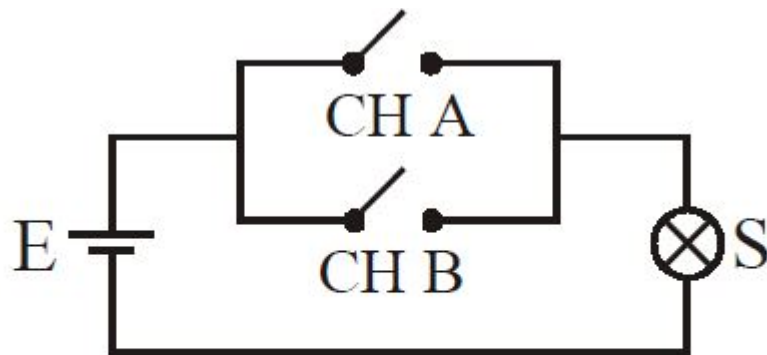
# Função OU ou OR

- A função **OU** é aquela que assume valor 1 quando uma ou mais **variáveis de entrada** forem iguais a 1 e assume 0 se, e somente se, **todas as variáveis de entrada** forem iguais a zero. Sua representação algébrica para duas variáveis de entrada é  $S=A+B$ , onde se lê: **S=A ou B**.
- Para entender melhor a função **OU** da álgebra booleana, analisa-se **todas as situações possíveis** de operação das chaves do circuito da Fig. 2.3.
- A convenção é a mesma adotada anteriormente:
  - chave aberta=0, chave fechada=1,
  - lâmpada apagada=0 e lâmpada acesa=1.



# Função OU ou OR

- O circuito acima mostra que a lâmpada acende quando qualquer uma das chaves estiver fechada e permanece apagada se ambas estiverem abertas, ou seja,  $CH A=0$ ,  $CH B=0$ , resulta em  $S=0$ .

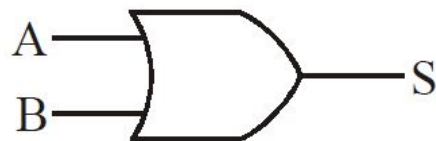


*Figura 2.3 – Circuito que representa a função **OU**.*



# Função OU ou OR

- A Fig. 2.4 ilustra a porta lógica que executa a função **OU** da álgebra de Boole, juntamente com a sua tabela da verdade.



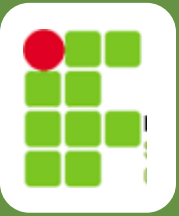
*Porta lógica **OU***

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

*Tabela da verdade da função **OU***

*Figura 2.4 – Porta lógica e tabela da verdade da função **OU**.*

- “A saída de uma porta **OU** será 1 se uma ou mais entradas forem 1”.



# Função NÃO ou NOT

- A função **NÃO** é aquela que inverte ou complementa o estado da variável de entrada, ou seja, se a variável estiver em 0, a saída vai para 1, e se estiver em 1 a saída vai para 0.
- É representada algebricamente da seguinte forma:, onde se lê: A barra ou NÃO A.



# Função NÃO ou NOT

- A análise do circuito da Fig. 2.5 ajuda a compreender melhor a função **NÃO** da álgebra Booleana. Será utilizada a mesma convenção dos casos anteriores.



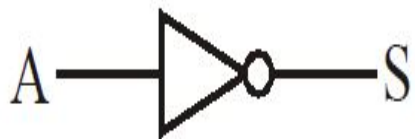
*Figura 2.5 – Circuito representativo da função NÃO.*

- Observando o circuito da Fig. 2.5, pode-se concluir que a lâmpada estará acesa somente se a chave estiver aberta
- (CH A=0, S=1), quando a chave fecha, a corrente desvia por ela e a Lâmpada apaga (CH A=1, S=0).



# Função NÃO ou NOT

- O inversor é o bloco lógico que executa a função **NÃO**. Sua **representação simbólica é vista na Figura juntamente com sua tabela da verdade**



*Porta lógica NÃO ou inversora*




A	S
0	1
1	0

*Tabela da verdade da função NÃO*

- “A saída de uma porta NÃO assume o nível lógico 1
- somente quando sua entrada é 0 e vice-versa”.



# Blocos Lógicos Básicos

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	<b>Função E:</b> 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	<b>Função E:</b> 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	<b>Função NÃO:</b> Inverte a variável aplicada à sua entrada.	$S=\overline{A}$									
A	S																		
0	1																		
1	0																		