

**DS011**

**Sistemas Numéricos**

**Prof. Clausius Duque Reis**

**`clausius.reis@ufpr.br`**

# Roteiro

- **O Sistema Decimal**
- **O Sistema Binário**
- **Conversão entre Binário e Decimal**
  - Inteiros
  - Frações
- **Notação Octal**
- **Notação Hexadecimal**
- **Exercícios**

# O Sistema Decimal

- **Sistema Baseado nos Dígitos Decimais**
  - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- **O que o número 83 significa?**
  - Significa 8 vezes 10 mais 3.
  - $83 = (8 \times 10) + 3$
- **O número 4728 significa**
  - 4 milhares, 7 centenas, 2 dezenas mais 8
  - $4728 = (4 \times 1000) + (7 \times 100) + (2 \times 10) + 8$

# O Sistema Decimal

- **O sistema decimal é dito possuir base 10**
  - Cada número é multiplicado por 10 elevado a uma potencia correspondente a posição do dígito.
  - $83 = (8 \times 10^1) + (3 \times 10^0)$
  - $4728 = (4 \times 10^3) + (7 \times 10^2) + (2 \times 10^1) + (8 \times 10^0)$
- **O mesmo principio vale para números decimais fracionários**
  - São utilizados potências negativas de 10
  - $0,256 = (2 \times 10^{-1}) + (5 \times 10^{-2}) + (6 \times 10^{-3})$
- **Um número composto por parte inteira e parte fracionária**
  - $472,256 = (4 \times 10^2) + (7 \times 10^1) + (2 \times 10^0) + (2 \times 10^{-1}) + (5 \times 10^{-2}) + (6 \times 10^{-3})$

# O Sistema Decimal

- Para a representação decimal do número
  - $X = \{ d_2d_1d_0, d_{-1}d_{-2}d_{-3} \}$
- O valor de  $X$  é
  - $X = \sum_i (d_i \times 10^i)$

# O Sistema Binário

- **Sistema Decimal**
  - 10 dígitos diferentes usados para representar números com uma base 10
- **Sistema Binário**
  - Apenas 2 dígitos: 1 e 0
  - Representados com a base 2
- **É comum incluir a base do número em subscrito para evitar confusão**
  - $83_{10}$  e  $4728_{10}$  números decimais
- **Os dígitos 1 e 0 em notação binária possuem o mesmo significado como quando em notação decimal**
  - $0_2 = 0_{10}$
  - $1_2 = 1_{10}$

# O Sistema Binário

- **Assim como na notação decimal,**
  - cada dígito de um número binário possui um valor dependendo de sua posição.
  - $10_2 = (1 \times 2^1) + (0 \times 2^0) = 2_{10}$
  - $11_2 = (1 \times 2^1) + (1 \times 2^0) = 3_{10}$
  - $100_2 = (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) = 4_{10}$

# Convertendo entre Binário e Decimal

- **Binário para Decimal**
  - **MUITO SIMPLES!!!**
  - Multiplique cada dígito pela potência de 2 apropriada e some os resultados.
  - Lembre dos exemplos anteriores!!!



# Convertendo entre Decimal e Binário

- **Decimal para Binário**

- **SIMPLES!!!**
- Inteiros e frações são manipulados separadamente.

- **Parte INTEIRA**

- Em notação binária um inteiro é representado por
  - $b_{m-1}b_{m-2}...b_2b_1b_0$        $b_i = 0$  ou  $1$
- Possui o valor
  - $(b_{m-1} \times 2_{m-1}) + (b_{m-2} \times 2_{m-2}) + ... + (b_1 \times 2_1) + b_0$

# Convertendo entre Decimal e Binário

- Converter o inteiro decimal  $N$  para a forma binária
  - Se dividirmos  $N$  por 2, na forma decimal, obtemos um quociente  $N_1$  e resto  $R_0$ 
    - $N = 2 \times N_1 + R_0$        $R_0 = 0 \text{ ou } 1$
  - Agora dividimos o quociente  $N_1$  por 2, obtemos um quociente  $N_2$  e um resto  $R_1$ 
    - $N_1 = 2 \times N_2 + R_1$        $R_1 = 0 \text{ ou } 1$
  - Então
    - $N = 2(2N_2 + R_1) + R_0 = (N_2 \times 2^2) + (R_1 \times 2^1) + R_0$
  - Se continuarmos...
    - $N_2 = 2N_3 + R_2$
  - Temos:
    - $N = (N_3 \times 2^3) + (R_2 \times 2^2) + (R_1 \times 2^1) + R_0$
  - Como  $N > N_1 > N_2 \dots$ , eventualmente será produzido :
    - Um quociente  $N_{m-1} = 1$  e um resto  $R_{m-2} = 0 \text{ ou } 1$ . Então:
    - $N = (1 \times 2^{m-1}) + (R_{m-2} \times 2^{m-2}) + \dots + (R_2 \times 2^2) + (R_1 \times 2^1) + R_0$

# Convertendo entre Decimal e Binário

- Entenderam?

# Convertendo entre Decimal e Binário

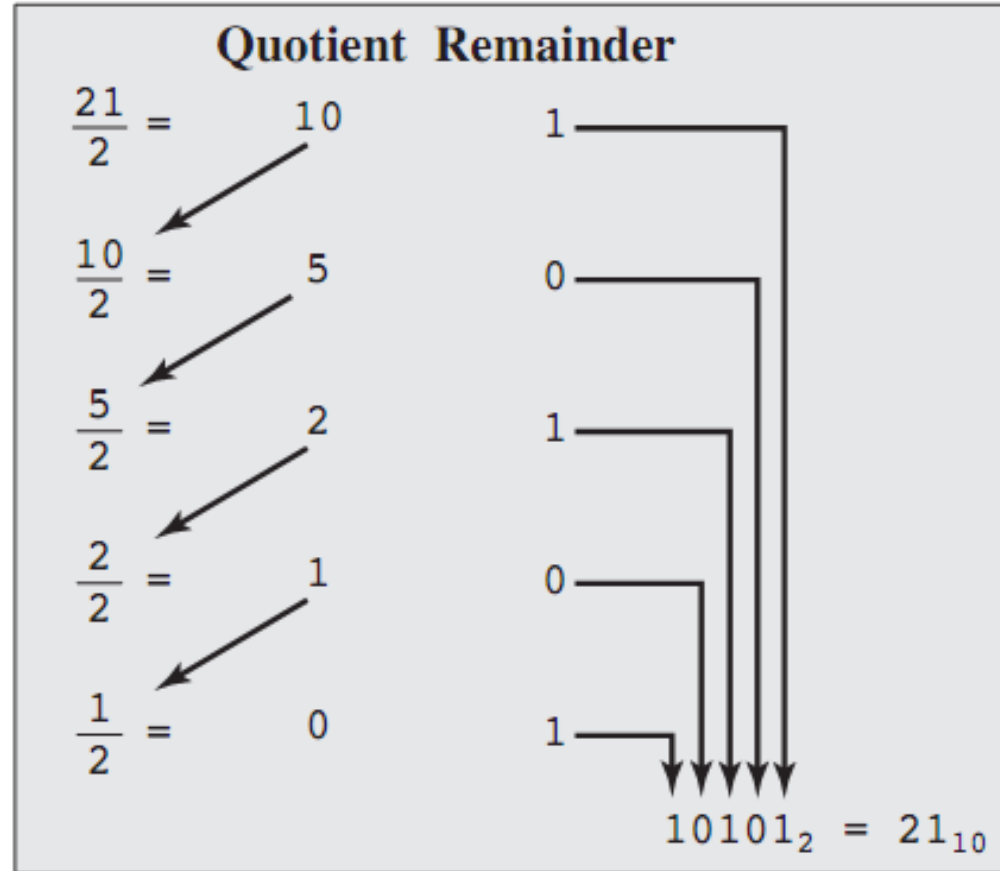
- Entenderam?
- Fácil né?

# Convertendo entre Decimal e Binário

- Entenderam?
- Fácil né?
- Converta o número  $11_{10}$  para binário

# Convertendo entre Decimal e Binário

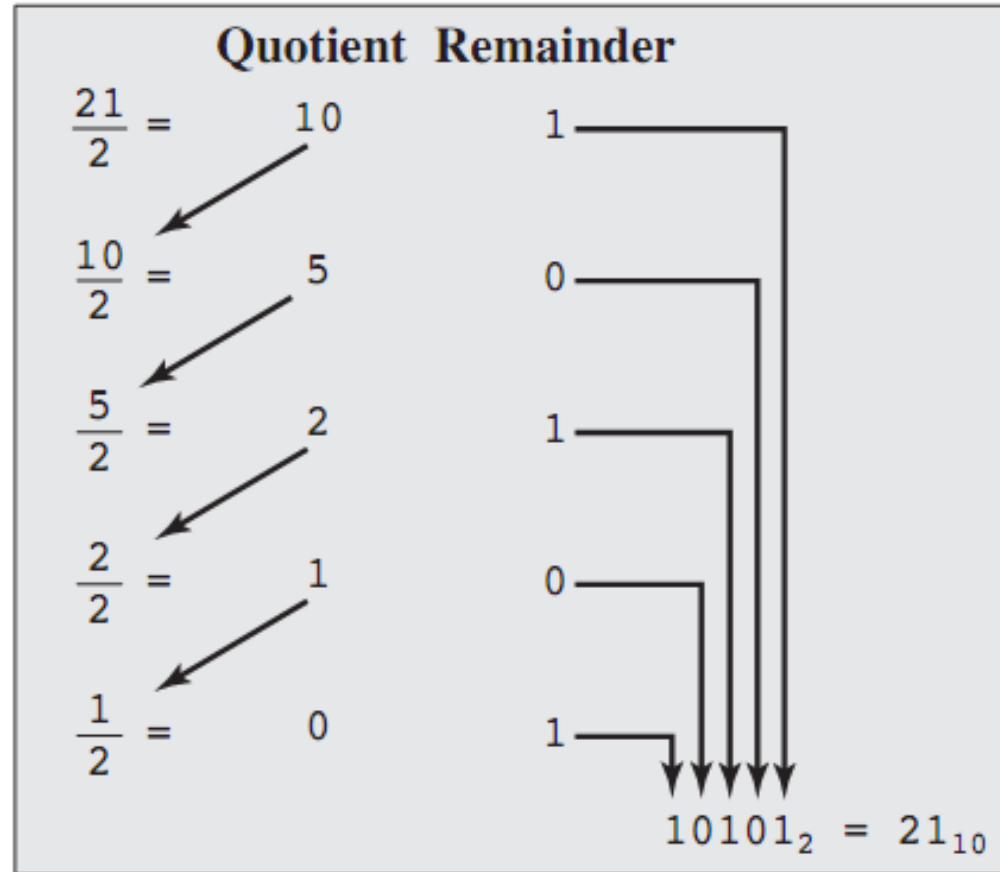
- Entenderam?
- Fácil né?
- Converta o número  $21_{10}$  para binário



(b)  $21_{10}$

# Convertendo entre Decimal e Binário

- Entenderam?
- Fácil né?
- Converta o número  $21_{10}$  para binário
- Agora converta um que você não ainda não treinou



(b)  $21_{10}$

# Notação Octal

- Sistema Octal
  - 8 dígitos diferentes usados para representar números com uma base 8.
  - 0, 1, 2, 3, 4, 5, 6, e 7
- Antigamente utilizado como uma alternativa mais compacta ao **binário**.
  - Programação em linguagem de máquina.
- Atualmente, o sistema hexadecimal é mais utilizado para esse fim.
- A aritmética é semelhante a dos sistemas decimal e binário.



# Notação Octal

- EXEMPLO:
  - $4701_8$  em base 10?

# Notação Octal

- EXEMPLO:
  - $4701_8$  em base 10?
    - $4 \times 8^3 + 7 \times 8^2 + 0 \times 8^1 + 1 \times 8^0 =$

# Notação Octal

- EXEMPLO:

- 4701<sub>8</sub> em base 10?

- $4 \times 8^3 + 7 \times 8^2 + 0 \times 8^1 + 1 \times 8^0 =$   
 $2048 + 448 + 0 + 1 =$

# Notação Octal

- EXEMPLO:

- $4701_8$  em base 10?

- $4 \times 8^3 + 7 \times 8^2 + 0 \times 8^1 + 1 \times 8^0 =$   
 $2048 + 448 + 0 + 1 =$   
 $2497_{(10)}$

# Notação Octal

- EXEMPLO:

- $4701_8$  em base 10?

- $4 \times 8^3 + 7 \times 8^2 + 0 \times 8^1 + 1 \times 8^0 =$   
 $2048 + 448 + 0 + 1 =$   
 $2497_{(10)}$

- $1234_8$  em base 10?

# Notação Octal

- EXEMPLO:

- $4701_8$  em base 10?

- $4 \times 8^3 + 7 \times 8^2 + 0 \times 8^1 + 1 \times 8^0 =$   
 $2048 + 448 + 0 + 1 =$   
 $2497_{(10)}$

- $1234_8$  em base 10?

- $1 \times 8^3 + 2 \times 8^2 + 3 \times 8^1 + 4 \times 8^0 =$

# Notação Octal

- EXEMPLO:

- $4701_8$  em base 10?

- $4 \times 8^3 + 7 \times 8^2 + 0 \times 8^1 + 1 \times 8^0 =$   
 $2048 + 448 + 0 + 1 =$   
 $2497_{(10)}$

- $1234_8$  em base 10?

- $1 \times 8^3 + 2 \times 8^2 + 3 \times 8^1 + 4 \times 8^0 =$   
 $512 + 128 + 24 + 4 =$

# Notação Octal

- EXEMPLO:

- $4701_8$  em base 10?

- $4 \times 8^3 + 7 \times 8^2 + 0 \times 8^1 + 1 \times 8^0 =$   
 $2048 + 448 + 0 + 1 =$   
 $2497_{(10)}$

- $1234_8$  em base 10?

- $1 \times 8^3 + 2 \times 8^2 + 3 \times 8^1 + 4 \times 8^0 =$   
 $512 + 128 + 24 + 4 =$   
 $668_{10}$



# Notação Octal

- EXEMPLO:
  - $2781_8$  em base 10?

# Notação Octal

- EXEMPLO:
  - $2781_8$  em base 10?

NÃO EXISTE ALGARISMO 8 EM BASE 8!!!



# Notação Octal

- EXEMPLO:
  - $2781_8$  em base 10?



NÃO EXISTE ALGARISMO 8 EM BASE 8!!!

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

# Notação Octal

- **Decimal – Octal**
  - Parte inteira
    - Sucessivas divisões por 8
- **Octal – Decimal**
  - $4701_8 = 4 \times 8^3 + 7 \times 8^2 + 0 \times 8^1 + 1 \times 8^0 = 2497_{10}$

# Notação Octal

- **Binário – Octal**

- Dividir os bits em grupos de 3 (partindo do ponto decimal).
- Substituir cada grupo pelo dígito **octal** correspondente.

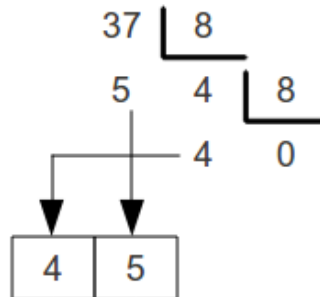
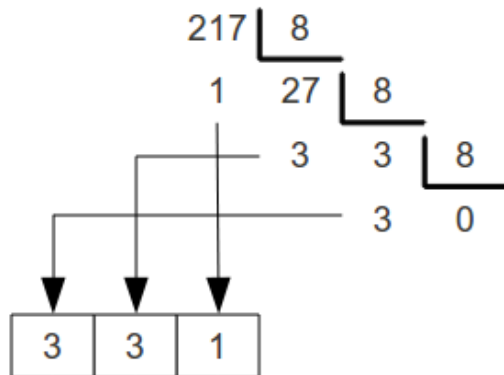
- **Octal – Binário**

- Substituir cada dígito **octal** pelo grupo de 3 bits correspondente.

# Notação Octal

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

# Notação Octal



Binary    011 100 110 100 010  
 Octal     3    4    6    4    2

Octal	7	2	3	0
Valor de Posição	$8^3$	$8^2$	$8^1$	$8^0$
Calculo	$7 \times 8^3 = 3584$	$2 \times 8^2 = 128$	$3 \times 8^1 = 24$	$0 \times 8^0 = 0$
Valor Final	$3584 + 128 + 24 + 0 = 3736$ (Decimal)			

## Hexadecimal



## Hexadecimal Hexa + Decimal

**Hexadecimal**  
**Hexa + Decimal**  
**6 + 10**

**Hexadecimal**  
**Hexa + Decimal**

**6 + 10**

**Base 16**

# Notação Hexadecimal

- Toda forma de dados nos computadores são representados por códigos binários
  - Natureza binária inerente dos computadores digitais
- Difícil manipulação para humanos.
- Notação mais compacta para profissionais da computação trabalharem com dados brutos
  - Notação decimal
    - Inerente para o ser humano
    - Processo de conversão para binário (e vice-versa) ... **TEDIOSO**

# Notação Hexadecimal

- Notação hexadecimal
  - 16 símbolos são usados (dígitos hexadecimais).
  - Cada possível combinação de quatro dígitos binários corresponde a um dígito hexadecimal.

0000 = 0

0100 = 4

1000 = 8

1100 = C

0001 = 1

0101 = 5

1001 = 9

1101 = D

0010 = 2

0110 = 6

1010 = A

1110 = E

0011 = 3

0111 = 7

1011 = B

1111 = F

- Uma sequência de dígitos hexadecimais pode representar um inteiro na base 16:

- $2C_{16} = (2_{16} \times 16^1) + (C_{16} \times 16^0) = (2_{10} \times 16^1) + (12_{10} \times 16^0) = 44$

# Notação Hexadecimal

- Notação hexadecimal
  - Usada não apenas para representar inteiros

Decimal (base 10)	Binary (base 2)	Hexadecimal (base 16)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	0001 0000	10
17	0001 0001	11
18	0001 0010	12
100	0110 0100	64
255	1111 1111	FF
256	0001 0000 0000	100

# Notação Hexadecimal

- Notação hexadecimal
  - Usada não apenas para representar inteiros
  - Notação concisa para representar qualquer sequência de dígitos binários

Decimal (base 10)	Binary (base 2)	Hexadecimal (base 16)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	0001 0000	10
17	0001 0001	11
18	0001 0010	12
100	0110 0100	64
255	1111 1111	FF
256	0001 0000 0000	100

# Notação Hexadecimal

- Notação hexadecimal
  - Usada não apenas para representar inteiros
  - Notação concisa para representar qualquer sequência de dígitos binários
    - Textos, números ou qualquer outro tipo de dado

Decimal (base 10)	Binary (base 2)	Hexadecimal (base 16)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	0001 0000	10
17	0001 0001	11
18	0001 0010	12
100	0110 0100	64
255	1111 1111	FF
256	0001 0000 0000	100



# Notação Hexadecimal

- Razões para utilizar a notação hexadecimal

Decimal (base 10)	Binary (base 2)	Hexadecimal (base 16)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	0001 0000	10
17	0001 0001	11
18	0001 0010	12
100	0110 0100	64
255	1111 1111	FF
256	0001 0000 0000	100

# Notação Hexadecimal

- Razões para utilizar a notação hexadecimal
  - Mais compacta do que a notação binária

Decimal (base 10)	Binary (base 2)	Hexadecimal (base 16)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	0001 0000	10
17	0001 0001	11
18	0001 0010	12
100	0110 0100	64
255	1111 1111	FF
256	0001 0000 0000	100

# Notação Hexadecimal

- Razões para utilizar a notação hexadecimal
  - Mais compacta do que a notação binária
  - Na maioria dos computadores, dados binários são organizados em grupos de 4 (1 dígito hexadecimal)

Decimal (base 10)	Binary (base 2)	Hexadecimal (base 16)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	0001 0000	10
17	0001 0001	11
18	0001 0010	12
100	0110 0100	64
255	1111 1111	FF
256	0001 0000 0000	100

# Notação Hexadecimal

- Razões para utilizar a notação hexadecimal
  - Mais compacta do que a notação binária
  - Na maioria dos computadores, dados binários são organizados em grupos de 4 (1 dígito hexadecimal)
  - Conversão entre binário e hexadecimal é extremamente fácil

Decimal (base 10)	Binary (base 2)	Hexadecimal (base 16)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	0001 0000	10
17	0001 0001	11
18	0001 0010	12
100	0110 0100	64
255	1111 1111	FF
256	0001 0000 0000	100

# Notação Hexadecimal

- Considere a string binária

110111100001

Decimal (base 10)	Binary (base 2)	Hexadecimal (base 16)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	0001 0000	10
17	0001 0001	11
18	0001 0010	12
100	0110 0100	64
255	11111111	FF
256	0001 0000 0000	100

# Notação Hexadecimal

- Considere a string binária

110111100001

1101    1110    0001

Decimal (base 10)	Binary (base 2)	Hexadecimal (base 16)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	0001 0000	10
17	0001 0001	11
18	0001 0010	12
100	0110 0100	64
255	11111111	FF
256	0001 0000 0000	100

# Notação Hexadecimal

- Considere a string binária

110111100001

1101 1110 0001



Decimal (base 10)	Binary (base 2)	Hexadecimal (base 16)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	0001 0000	10
17	0001 0001	11
18	0001 0010	12
100	0110 0100	64
255	1111 1111	FF
256	0001 0000 0000	100

# Notação Hexadecimal

- Considere a string binária

110111100001

1101	1110	0001
D	E	1

Decimal (base 10)	Binary (base 2)	Hexadecimal (base 16)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	0001 0000	10
17	0001 0001	11
18	0001 0010	12
100	0110 0100	64
255	11111111	FF
256	0001 0000 0000	100



# Notação Hexadecimal

- Considere a string binária

110111100001

1101    1110    0001    =   DE1<sub>16</sub>  
D        E        1

Decimal (base 10)	Binary (base 2)	Hexadecimal (base 16)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	0001 0000	10
17	0001 0001	11
18	0001 0010	12
100	0110 0100	64
255	1111 1111	FF
256	0001 0000 0000	100

# Notação Hexadecimal

- Considere a string binária

110111100001

1101    1110    0001    =    DE1<sub>16</sub>  
D        E        1

- $110111100001_2 = DE1_{16}$

Decimal (base 10)	Binary (base 2)	Hexadecimal (base 16)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	0001 0000	10
17	0001 0001	11
18	0001 0010	12
100	0110 0100	64
255	1111 1111	FF
256	0001 0000 0000	100

# Notação Hexadecimal

- Hexadecimal - Decimal
  - $2C_{16} = (2_{16} \times 16^1) + (C_{16} \times 16^0) = (2_{10} \times 16^1) + (12_{10} \times 16^0) = 44_{10}$
- Decimal – Hexadecimal
  - Divisões sucessivas por 16 (parte inteira).
  - Multiplicações sucessivas por 16 (parte fracionária)
- Hexadecimal – Binário (Extremamente fácil)
  - Substituir cada dígito hexadecimal pelo grupo de 4 bits correspondente
- Binário – Hexadecimal (Extremamente fácil)
  - Dividir os bits em grupos de 4 (partindo do ponto decimal).
  - Substituir cada grupo pelo dígito hexadecimal correspondente.

# Exercícios

- Converta os números binários para decimal
  - A) 001100                      B) 11100
  - C) 101                              D) 001100100010
- Converta os números decimais para binário
  - A) 64                              B) 34
  - B) 679                              C) 1063

**E para nosso próximo truque...**

# **ARITMÉTICA BINÁRIA**

# Aritmética Binária

- **Soma decimal**

A	B	+
0	0	0
0	1	1
1	0	1
1	1	2

# Aritmética Binária

- **Soma decimal**

A	B	+
0	0	0
0	1	1
1	0	1
1	1	2

- **Soma binária**

A	B	+
0	0	0
0	1	1
1	0	1
1	1	10

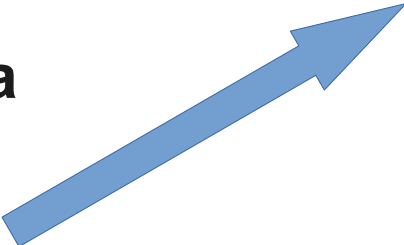
# Aritmética Binária

- **Soma decimal**

A	B	+
0	0	0
0	1	1
1	0	1
1	1	2

- **Soma binária**

A	B	+
0	0	0
0	1	1
1	0	1
1	1	10



A	B	+
0	0	0
0	1	1
1	0	1
1	1	0



# Aritmética Binária

- **Soma decimal**

A	B	+
0	0	0
0	1	1
1	0	1
1	1	2

- **Soma binária**

A	B	+
0	0	0
0	1	1
1	0	1
1	1	10

A	B	+
0	0	0
0	1	1
1	0	1
1	1	0

→ Vai 1

# Aritmética Binária

- **Soma binária**

- $5_{10} + 3_{10} = 8_{10}$

- $0101_2 + 0011_2 = 1000_2$

- $5_{10} + 2_{10} = 7_{10}$

- $0101_2 + 0010_2 = 0111_2$

- $11_{10} + 5_{10} = 16_{10}$

- $1011_2 + 0101_2 = 10000_2$  (Necessário mais um bit)

# Aritmética Binária

- **Subtração decimal**

A	B	-
0	0	0
0	1	-1
1	0	1
1	1	0

# Aritmética Binária

- **Subtração decimal**

A	B	-
0	0	0
0	1	-1
1	0	1
1	1	0

- **Subtração binária**

A	B	-
0	0	0
0	1	1
1	0	1
1	1	0

# Aritmética Binária

- **Subtração decimal**

A	B	-
0	0	0
0	1	-1
1	0	1
1	1	0

- **Subtração binária**

A	B	-
0	0	0
0	1	1
1	0	1
1	1	0



A	B	-
0	0	0
0	1	1
1	0	1
1	1	0

# Aritmética Binária

- Subtração decimal

A	B	-
0	0	0
0	1	-1
1	0	1
1	1	0

- Subtração binária

A	B	-
0	0	0
0	1	1
1	0	1
1	1	0

A	B	-
0	0	0
0	1	1
1	0	1
1	1	0

→ Vem 1

# Aritmética Binária

- **Subtração binária**

$$- 13_{10} - 4_{10} = 9_{10}$$

# Aritmética Binária

- **Subtração binária**

$$- 13_{10} - 4_{10} = 9_{10}$$

$$\begin{array}{r} 1101 \\ - 0100 \\ \hline 1001 \end{array}$$

Processo simples

$$- 1101_2 - 0100_2 = 1001_2$$



# Aritmética Binária

- **Subtração binária**

$$- 5_{10} - 2_{10} = 3_{10}$$

# Aritmética Binária

- **Subtração binária**

$$- 5_{10} - 2_{10} = 3_{10}$$

$$\begin{array}{r} \overset{-1}{0}101 \\ - 0010 \\ \hline 1001 \end{array}$$

Processo com 1 empréstimo

$$- 0101_2 - 0010_2 = 0011_2$$

# Aritmética Binária

- **Subtração binária**

$$- 9_{10} - 3_{10} = 6_{10}$$

# Aritmética Binária

- **Subtração binária**

$$- 9_{10} - 3_{10} = 6_{10}$$

$$\begin{array}{r} \textcolor{red}{-1} \textcolor{red}{-1} \\ 1001 \\ - \\ 0011 \\ \hline 0110 \end{array}$$

Processo com 2 empréstimos

$$- 1001_2 - 0011_2 = 0110_2$$

# Aritmética Binária

- **Multiplicação binária**

–  $11_{10} * 3_{10} = 33_{10}$

$$\begin{array}{r} 1011 \\ 0011 \\ \hline \end{array}$$
  
$$\hline$$

# Aritmética Binária

- **Multiplicação binária**

–  $11_{10} * 3_{10} = 33_{10}$

$$\begin{array}{r} 1011 \\ 0011 \\ \hline 1011 \\ \\ \hline \end{array}$$

# Aritmética Binária

- **Multiplicação binária**

–  $11_{10} * 3_{10} = 33_{10}$

$$\begin{array}{r} 1011 \\ 0011 \\ \hline 1011 \\ 1011+ \\ \hline \end{array}$$

# Aritmética Binária

- **Multiplicação binária**

–  $11_{10} * 3_{10} = 33_{10}$

$$\begin{array}{r} 1011 \\ 0011 \\ \hline 1011 \\ 1011+ \\ 0000+ \\ \hline \end{array}$$



# Aritmética Binária

- **Multiplicação binária**

–  $11_{10} * 3_{10} = 33_{10}$

$$\begin{array}{r} 1011 \\ 0011 \\ \hline 1011 \\ 1011+ \\ 0000+ \\ 0000+ \\ \hline \end{array}$$

# Aritmética Binária

- **Multiplicação binária**

–  $11_{10} * 3_{10} = 33_{10}$

$$\begin{array}{r} 1011 \\ 0011 \\ \hline 1011 \\ 1011+ \\ 0000+ \\ 0000+ \\ \hline 0100001_2 \end{array}$$

# Aritmética Binária

- Multiplicação binária

–  $11_{10} * 3_{10} = 33_{10}$


$$\begin{array}{r} 1011 \\ 0011 \\ \hline 1011 \\ 1011+ \\ 0000+ \\ 0000+ \\ \hline 0100001_2 = 33_{10} \end{array}$$

# Aritmética Binária

## Divisão binária

$$- 10_{10} / 2_{10} = 5_{10}$$

Dividendo		Divisor
Resto		Quociente

10		2		1010		0010
0		5		0000		0101

# Aritmética Binária

## Divisão binária

$$- 10_{10} / 2_{10} = 5_{10}$$

Dividendo

Divisor

Resto

Quociente

$$1010 \overline{) 0010}$$

# Aritmética Binária

# Divisão binária

$$- 10_{10} / 2_{10} = 5_{10}$$

# Dividendo

## Divisor

## Resto

## Quociente

1010 | 0010

1 0 ←

0 ← 10 < 01 então coloco 0

# Aritmética Binária

## Divisão binária

$$- 10_{10} / 2_{10} = 5_{10}$$

Dividendo

Divisor

Resto

Quociente

$$\begin{array}{r} 1010 \quad | \quad 00\boxed{10} \\ \underline{10} \quad 01 \end{array} \quad \leftarrow 10 = 10 \text{ então coloco } 1$$

# Aritmética Binária

## Divisão binária

$$- 10_{10} / 2_{10} = 5_{10}$$

Dividendo

Divisor

Resto

Quociente

$$\begin{array}{r|l} 1010 & 0010 \\ - 10 & 01 \\ \hline 0010 & \end{array}$$

Efetuo uma subtração  
binária



# Aritmética Binária

## Divisão binária

$$- 10_{10} / 2_{10} = 5_{10}$$

Dividendo

Divisor

Resto

Quociente

$$\begin{array}{r} 1010 \quad | \quad 0010 \\ - 0010 \\ \hline 0010 \\ - 0010 \\ \hline 0 \end{array}$$

10 < 01 então coloco 0

# Aritmética Binária

## Divisão binária

$$- 10_{10} / 2_{10} = 5_{10}$$

Dividendo

Divisor

Resto

Quociente

$$\begin{array}{r} 1010 \quad | \quad 0010 \\ - 0010 \\ \hline 0101 \\ - 0010 \\ \hline 0010 \\ - 0010 \\ \hline 0000 \end{array}$$

10 = 10 então coloco 1

# Aritmética Binária

## Divisão binária

$$- 10_{10} / 2_{10} = 5_{10}$$

Dividendo

Divisor

Resto

Quociente

$$\begin{array}{r|l} 1010 & 0010 \\ - 10 & 0101 \\ \hline 0010 & \\ - 10 & \\ \hline 00 & \end{array}$$

Efetuo uma subtração  
binária

## Shift Left Logical (SLL) e Shift Right Logical (SRL)

- Representado por:
  - $\ll$  (SLL) e  $\gg$  (SRL)

## Shift Left Logical (SLL) e Shift Right Logical (SRL)

- Representado por:
  - $\ll$  (SLL) e  $\gg$  (SRL)
- Exemplos no quadro:
  - $00001010_2 \ll$
- Deslocamento à esquerda (SLL) **multiplica** por **2 (base)**
- Deslocamento à direita (SRL) **divide** por **2 (base)**

## Shift Left Logical (SLL) e Shift Right Logical (SRL)

- Representado por:
  - $\ll$  (SLL) e  $\gg$  (SRL)
- Exemplos no quadro:
  - $00001010_2 \ll$       10 vira 20
  - $00010110_2 \ll$

# Aritmética Binária

## Shift Left Logical (SLL) e Shift Right Logical (SRL)

- Representado por:
  - $\ll$  (SLL) e  $\gg$  (SRL)
- Exemplos no quadro:
  - $00001010_2 \ll$  10 vira 20
  - $00010110_2 \ll$  22 vira 44
  - $00000110_2 \gg$

# Aritmética Binária

## Shift Left Logical (SLL) e Shift Right Logical (SRL)

- Representado por:
  - $\ll$  (SLL) e  $\gg$  (SRL)
- Exemplos no quadro:
  - $00001010_2 \ll$  10 vira 20
  - $00010110_2 \ll$  22 vira 44
  - $00000110_2 \gg$  6 vira 3
  - $00001101_2 \gg$



# Aritmética Binária

## Shift Left Logical (SLL) e Shift Right Logical (SRL)

- Representado por:
  - $\ll$  (SLL) e  $\gg$  (SRL)
- Exemplos no quadro:
  - $00001010_2 \ll$  10 vira 20
  - $00010110_2 \ll$  22 vira 44
  - $00000110_2 \gg$  6 vira 3
  - $00001101_2 \gg$  13 vira 6

# Aritmética Binária

## Shift Left Logical (SLL) e Shift Right Logical (SRL)

- Representado por:
  - $\ll$  (SLL) e  $\gg$  (SRL)
- Exemplos no quadro:
  - $00001010_2 \ll$  10 vira 20
  - $00010110_2 \ll$  22 vira 44
  - $00000110_2 \gg$  6 vira 3
  - $00001101_2 \gg$  13 vira 6
- Deslocamento à esquerda (SLL) multiplica por 2 (base)

## Shift Left Logical (SLL) e Shift Right Logical (SRL)

- Representado por:
  - $\ll$  (SLL) e  $\gg$  (SRL)
- Exemplos no quadro:
  - $00001010_2 \ll$  10 vira 20
  - $00010110_2 \ll$  22 vira 44
  - $00000110_2 \gg$  6 vira 3
  - $00001101_2 \gg$  13 vira 6
- Deslocamento à esquerda (SLL) multiplica por 2 (base)
- Deslocamento à direita (SRL) divide por 2 (base)