

# Aula 2 – Sistemas de Numeração e Códigos

Prof. Dr. Emerson Carlos Pedrino  
024376 – Circuitos Digitais  
DC/UFSCar  
[www.dc.ufscar.br/~emerson](http://www.dc.ufscar.br/~emerson)

# Conversões entre Sistemas de Numeração

## 1. BINÁRIO ➡ DECIMAL

110100

$$\begin{aligned}(110100)_2 &= 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = \\ &= 32 + 16 + 4 = (52)_{10}\end{aligned}$$

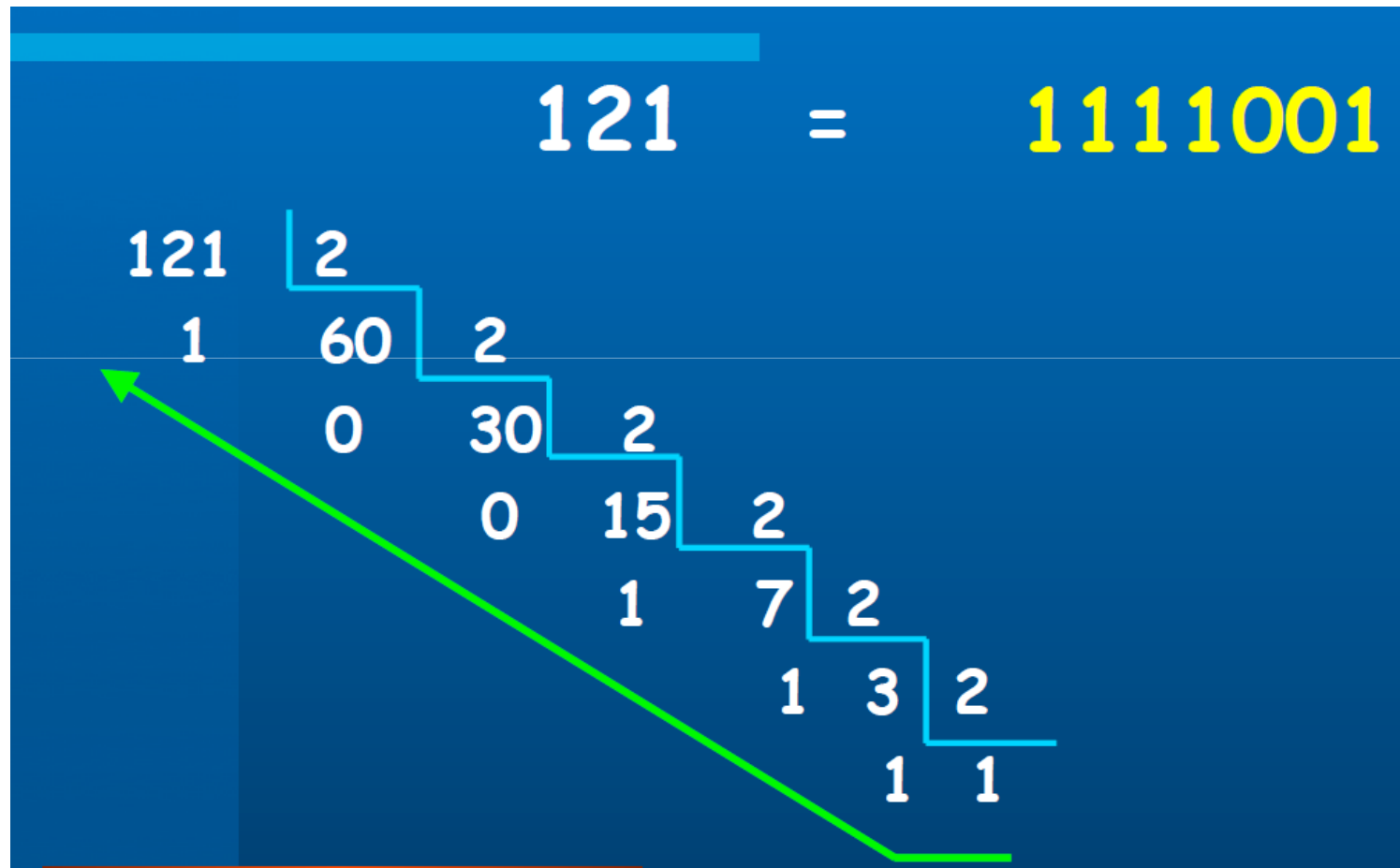
# Conversão entre Sistemas de Numeração

## 2. DECIMAL → BINÁRIO

Ex.:  $(49)_{10} \rightarrow (?)_2$

$$(49)_{10} = (110001)_2$$

# Decimal -> Binário



# Binário -> Decimal

## **BINÁRIO ➡ DECIMAL**

**Com dígitos após a vírgula (números não inteiros)**

**Ex. 11110,01**

$$\begin{aligned}(11110,01)_2 &= 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} \\ &= 16 + 8 + 4 + 2 + 0,25 = (30,25)_{10}\end{aligned}$$

# Decimal -> Binário (Parte Fracionária)

## DECIMAL ➡ BINÁRIO

Com dígitos após a vírgula (números não inteiros)

- Calcula-se a conversão da parte inteira e da parte fracionária separadamente;
- Para a parte fracionária, utiliza-se o método das **multiplicações sucessivas**:
  - Multiplica-se o nº fracionário por 2.
  - Desse resultado, a parte inteira será utilizada como dígito binário e a parte fracionária restante é novamente multiplicada por 2.
  - Repete-se o processo até que a parte fracionária do último produto seja igual a zero.

# Exemplo

## DECIMAL $\Rightarrow$ BINÁRIO

Com dígitos após a vírgula (números não inteiros)

**Ex. 4,1875**

$$(4)_{10} = (100)_2$$

$$(0,1875)_{10} =$$

$$0,1875 \times 2 = 0,3750 = 0 + 0,3750$$

$$0,3750 \times 2 = 0,7500 = 0 + 0,7500$$

$$0,7500 \times 2 = 1,5000 = 1 + 0,5000$$

$$0,5000 \times 2 = 1,0000 = 1 + 0,0000$$

$$= (0011)_2$$

$$(4,1875)_{10} = (100,0011)_2$$

# Sistema Octal

## 3. SISTEMA OCTAL

- Composto por 8 símbolos ou numerais;
- Base 8 ➡ 0, 1, 2, 3, 4, 5, 6, 7.

$$o_{n-1} \dots o_3 o_2 o_1 o_0 = o_{n-1} 8^{n-1} + \dots + o_3 8^3 + o_2 8^2 + o_1 8^1 + o_0 8^0$$

$$\text{Ex.: } (372)_8 = 3 \cdot 8^2 + 7 \cdot 8^1 + 2 \cdot 8^0 = (250)_{10}$$




# Decimal -> Octal

$$(179)_{10} = ?$$

• Exemplo:

179		8
3	22	8
	6	2



Resposta:

$$(263)_8$$

# Octal -> Decimal

$$(274)_8 =$$

- Exemplo: **274**

$$2 \times 8^2 = 128$$

$$7 \times 8^1 = 56$$

$$4 \times 8^0 = 4$$

$$128 +$$

$$56 +$$

$$4 +$$

---

$$(188)_{10}$$

# Binário <-> Octal

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

- Como 8 é a terceira potência de 2, pode-se converter de octal em binário transformando cada dígito octal em seu equivalente com 3 dígitos binários.

# Octal -> Binário

• Exemplo  $\Rightarrow$  257

	2	5	7
	↓	↓	↓
010		101	111

• Resposta  $\Rightarrow$  010 101 111

# Binário -> Octal

- A conversão de binário para octal é o inverso dos procedimentos acima;
- Agrupe os bits de três em três, e converta cada grupo em seu equivalente octal;
- Se houver necessidade, adicione zeros à esquerda do número binário.

# Binário -> Octal

Exemplo  $\Rightarrow$  10101111

010	101	111
↓	↓	↓
2	5	7

● Resposta  $\Rightarrow$  257

# Contagem em Octal

0	→ 14	→ 30	→ 104
1	15	31	105
2	16	....	....
3	17	....	....
4	20	....	....
5	21	75	776
6	22	76	777
7	23	77	1000
10	24	100	1001
11	25	101	....
12	26	102	....
13	27	103	....

# Sistema Hexadecimal

- Composto por 16 símbolos ou numerais;
- Base 16 ➡ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

$$h_{n-1} \dots h_3 h_2 h_1 h_0 = h_{n-1} 16^{n-1} + \dots + h_3 16^3 + h_2 16^2 + h_1 16^1 + h_0 16^0$$

$$\text{Ex.: } (1A7)_{16} = 1 \cdot 16^2 + 10 \cdot 16^1 + 7 \cdot 16^0 = (423)_{10}$$



# Decimal -> Hexadecimal

- Exemplo:

2479

16

15

154

16

10

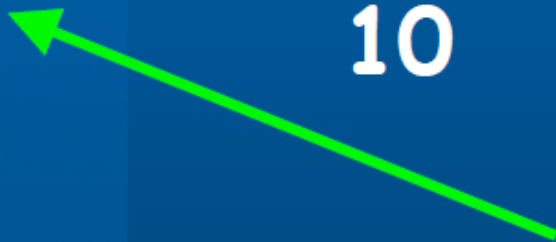
9

9  $\Rightarrow$  9

10  $\Rightarrow$  A

15  $\Rightarrow$  F

9AF



# Hexadecimal -> Decimal

- Exemplo: 9AF

$$9 \times 16^2 = 2304$$

$$A \times 16^1 = 160$$

$$F \times 16^0 = 15$$

$$2304 +$$

$$160 +$$

$$15 +$$

---

$$2479$$

# Hexadecimal <-> Binário

- Para converter um número hexadecimal em um número binário, converta cada dígito hexadecimal em seu equivalente de 4 bits;
- A vantagem do sistema hexadecimal, é poder agrupar cada conjunto de 4 dígitos binários em apenas 1 dígito hexa;
- Muito utilizado em endereçamento de memória.

# Números Hexadecimais

Dec	Binário	Hexa
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7

Dec	Binário	Hexa
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

# Hexadecimal -> Binário

9AF

9

A

F



1001

1010

1111

● Resposta  $\Rightarrow$  100110101111

# Binário -> Hexadecimal

Exemplo  $\Rightarrow$  **1110101101**

**0011**    1010    1101



3

A

D

● Resposta  $\Rightarrow$  **3AD**

# Contagem em Hexadecimal

0	→ C	→ 18	→ 99	→ FA	→ 9FD
1	D	19	9A	FB	9FE
2	E	1A	9B	FC	9FF
3	F	1B	9C	FD	A00
4	10	1C	9D	FE	A01
5	11	1D	9E	FF	.....
6	12	1E	9F	100	.....
7	13	1F	A0	101	FFE
8	14	20	A1	102	FFF
9	15	...	...	....	1000
A	16	...	...	....	1001
B	17	98	F9	9FC	.....

# Código BCD

**DECIMAL ➡ BINÁRIO ou BINÁRIO ➡ DECIMAL**

- Muito complicado na prática
- Hardware complexo
- Binário Puro

**Alternativa:**

- Uso de um Código
- Decimal codificado em Binário



# Código BCD

## 1. CÓDIGO BCD (*Binary-Coded Decimal*)

- Cada dígito decimal é representado por um “código” equivalente em binário;
- Não é um sistema de numeração;
- É diferente de conversão em binário puro;
- Quantos bits?
  - 4 bits
  - 16 códigos possíveis → só se usa 10 para o sistema decimal

# Código BCD 8421

Decimal	Binário	Decimal	Binário
0	0000	8	1000
1	0001	9	1001
2	0010	10	1010
3	0011	11	1011
4	0100	12	1100
5	0101	13	1101
6	0110	14	1110
7	0111	15	1111

**Não Utilizado**

# BCD 8421 x Número Binário

Decimal	Binário	BCD (8421)
0	0	0000
1	1	0001
2	10	0010
3	11	0011
4	100	0100
5	101	0101
6	110	0110
7	111	0111
8	1000	1000
9	1001	1001
10	1010	0001 0000
11	1011	0001 0001
12	1100	0001 0010
...	...	...
98	1100010	1001 1000
99	1100011	1001 1001
100	1100100	0001 0000 0000
101	1100101	0001 0000 0001
...	...	...
578	1001000010	0101 0111 1000
...	...	...

# Decimal -> BCD

- Exemplo  $\Rightarrow$  137

1	3	7
↓	↓	↓
0001	0011	0111

- Resposta  $\Rightarrow$  (000100110111)<sub>BCD</sub>  $\Rightarrow$  12 bits
- Em Binário  $\Rightarrow$  (10001001)<sub>2</sub>  $\Rightarrow$  8 bits

# Código Gray

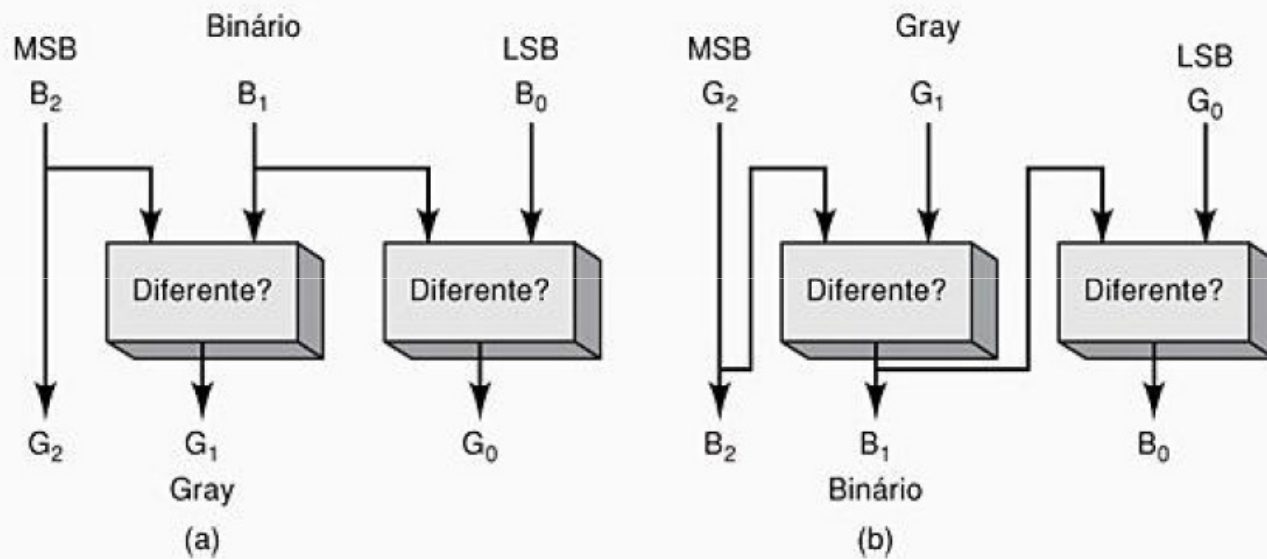
- Garante que apenas um *bit* muda entre dois números sucessivos em uma seqüência.
- Usado para reduzir a probabilidade de um circuito digital interpretar mal uma entrada que esteja mudando.

# Código Gray (Tocci et al.)

**TABELA 2.2** Equivalentes entre binários de três bits e código Gray

B <sub>2</sub>	B <sub>1</sub>	B <sub>0</sub>	G <sub>2</sub>	G <sub>1</sub>	G <sub>0</sub>
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

# Código Gray



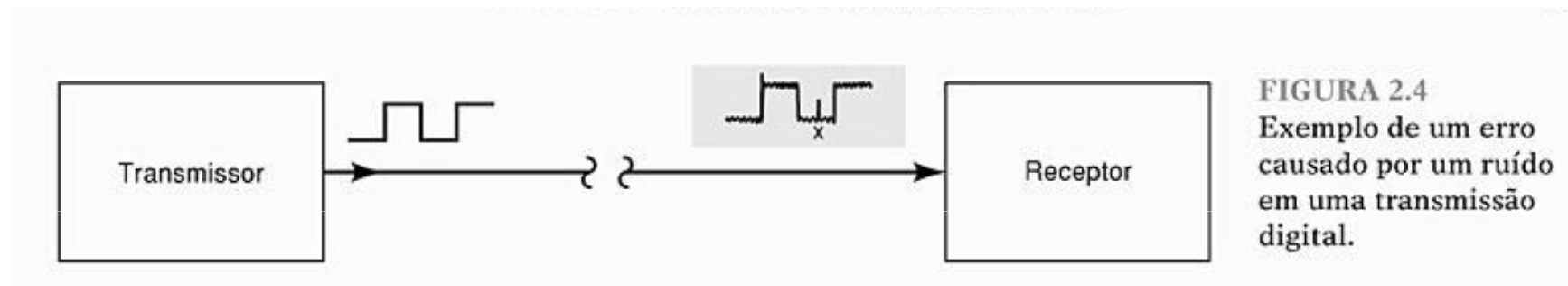
**FIGURA 2.2**  
Convertendo  
(a) binário em Gray e  
(b) Gray em binário.

# Códigos Alfanuméricos

- Código ASCII (American Standard Code for Information Interchange) – 7 *bits* -> 128 representações codificadas.
- Usado para transferência de informações entre um computador e dispositivos externos.



# Detecção de Erros pelo Método de Paridade (Tocci et al.)



# *Bit* de Paridade

- Exemplo de Paridade Par:
  - Caractere ASCII 'C':

1 1 0 0 0 0 1 1  
↑ bit de paridade anexado<sup>1</sup>

- Exemplo de Paridade Ímpar:
  - Caractere ASCII 'A':

1 1 0 0 0 0 0 1

# Exercícios

- 1. Quantos *bits* de dados um CD-ROM típico de 650 MB pode armazenar?
- 2.

## APLICAÇÃO 2.2

Para programar vários microcontroladores, as instruções binárias são armazenadas em um arquivo de um computador pessoal de uma forma especial conhecida como formato Intel-Hex. A informação hexadecimal é codificada em caracteres ASCII de modo a ser exibida facilmente na tela do PC, impressa e transmitida um caractere de cada vez por um porto serial COM de um PC padrão. A seguir, você pode ver uma linha de um arquivo em formato Intel-Hex:

```
:10002000F7CFFFCF1FEF2FEF2A95F1F71A95D9F7EA
```

O primeiro caractere enviado é o código ASCII para dois pontos, seguido por um 1. Cada um deles possui um bit de paridade par anexado como o bit mais significativo. Um instrumento de teste verifica o padrão binário ao passar pelo cabo até o microcontrolador.

- (a) Qual deve ser a aparência do padrão binário (inclusive a paridade)? (MSB – LSB)
- (b) O valor 10, seguindo os dois pontos, representa o número de bytes hexadecimal total que devem ser carregados na memória do micro. Qual é o número decimal de bytes que está sendo carregado?
- (c) O número 0020 é um valor hexa de 4 dígitos que representa o endereço em que o primeiro byte será armazenado. Qual é o maior endereço possível? Quantos bits seriam necessários para representar esse endereço?
- (d) O valor do primeiro byte de dados é F7. Qual é o valor (em binário) do nibble menos significativo desse byte?

# Exercícios

- 3.

## APLICAÇÃO 2.3

Um pequeno computador de controle de processos usa código hexadecimal para representar seus endereços de memória de 16 bits.

- (a) Quantos dígitos hexadecimais são necessários?
- (b) Qual é a faixa de endereços em hexadecimal?
- (c) Quantas posições de memória existem?

# Exercícios

- 4.

## APLICAÇÃO 2.4

Números são fornecidos em BCD para um sistema baseado em microcontrolador, mas armazenados em binário puro. Como programador, você precisa decidir se precisa de um ou dois bytes de posições de armazenamento.

- (a) Quantos bytes serão necessários se o sistema requerer uma entrada decimal de 2 dígitos?
- (b) E se forem necessários 3 dígitos?

# Exercícios

- 5.

## APLICAÇÃO 2.5

Quando é necessário transmitir caracteres ASCII entre dois sistemas independentes (como entre um computador e um modem), é preciso encontrar uma forma de avisar o receptor quando um novo caractere está entrando. Além disso, muitas vezes é preciso detectar erros na transmissão. O estado normal de repouso da linha de transmissão é o lógico 1. Quando o transmissor envia um caractere ASCII, é preciso ser “enquadrado” (*framed*) para que o receptor saiba onde os dados começam e terminam. O primeiro bit deve ser sempre um bit inicial (*start bit* — o nível lógico 0). A seguir, o código ASCII é enviado primeiro o LSB e por último o MSB. Depois do MSB, um bit de paridade é anexado para verificar possíveis erros de transmissão. Finalmente, a transmissão é encerrada pelo envio de um bit de parada (*stop bit* — o nível lógico 1). Na Figura 2.5 você pode ver uma transmissão assíncrona típica de um código ASCII de sete bits para o símbolo # (Hexa 23) com paridade par.

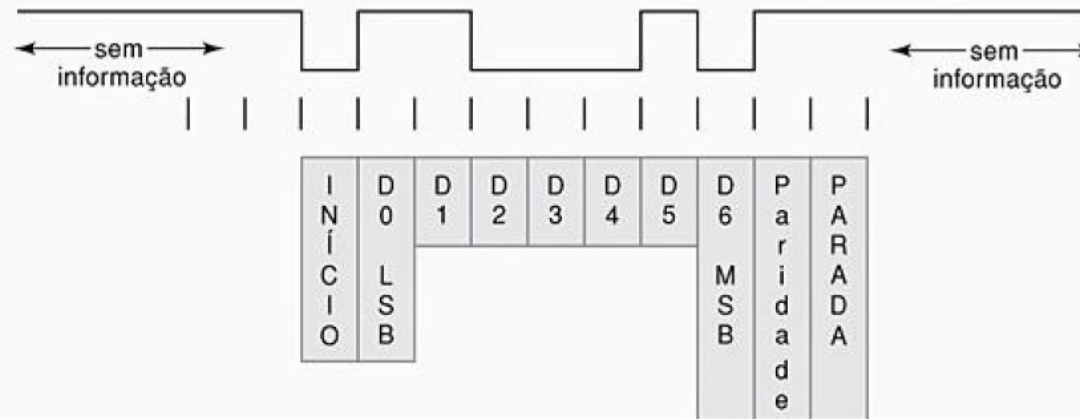


FIGURA 2.5  
Dados seriais assíncronos  
com paridade par.

# Exercícios\* 😊

- Problemas do Cap. 2: Sugestão: Todos. 😊

# Referências

- Tocci, R. J. et al. Sistemas Digitais (princípios e aplicações), 10a Edição. Pearson, 2007.
- Vieira, M. A. C. SEL-0414-Sistemas Digitais, EESC-USP.