

DS011

Excesso-N e Ponto Flutuante

Prof. Clausius Duque Reis

`clausius.reis@ufpr.br`

O Sistema Binário... Relembrando

- **Assim como na notação decimal,**
 - cada dígito de um número binário possui um valor dependendo de sua posição.
 - $10_2 = (1 \times 2^1) + (0 \times 2^0) = 2_{10}$
 - $11_2 = (1 \times 2^1) + (1 \times 2^0) = 3_{10}$
 - $100_2 = (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) = 4_{10}$
- **Valores de frações são representadas com potências negativas da base**
 - $1001,101 = (1 \times 2^3) + (0 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3})$

O Sistema Binário

- Para a representação binária do número
 - $Y = \{ b_2b_1b_0, b_{-1}b_{-2}b_{-3} \}$
- O valor de Y é
 - $Y = \sum_i (b_i \times 2^i)$

O Sistema Binário

- Para a representação binária do número

- $Y = \{ b_2b_1b_0, b_{-1}b_{-2}b_{-3} \}$



- O valor de Y é

- $Y = \sum_i (b_i \times 2^i)$

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
 - $b_{m-1}b_{m-2}...b_2b_1b_0$ $b_i = 0 \text{ ou } 1$
- 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

- E a parte fracionária?

Convertendo entre Decimal e Binário

- Parte FRACIONÁRIA

- Em notação binária, um número com valor entre 0 e 1 é representado por

- $0, b_{-1} b_{-2} b_{-3} \dots$ $b_i = 0 \text{ ou } 1$

- E possui o valor

- $(b_{-1} \times 2^{-1}) + (b_{-2} \times 2^{-2}) + (b_{-3} \times 2^{-3}) \dots$

Convertendo entre Decimal e Binário

- O algoritmo de conversão envolve **repetidas multiplicações por 2**.
 - A cada passo a parte fracionaria do número é multiplicada por 2.
 - O dígito à esquerda da vírgula (0 ou 1) contribui para a representação binária.
- O processo **não é exato**.
 - Uma fração decimal com um número finito de dígitos pode gerar uma representação binária com um número **infinito de bits**.
 - O processo é cessado após uma **sequência predefinida de passos**, dependendo da **precisão desejada**.

Convertendo entre Decimal e Binário

- Entenderam?

Convertendo entre Decimal e Binário

- Entenderam?
- Mais fácil né?

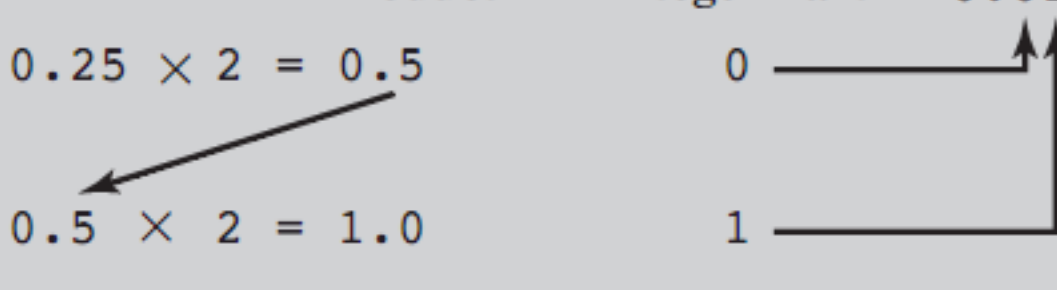
Convertendo entre Decimal e Binário

- Entenderam?
- Mais fácil né?
- **Converta o número $0,25_{10}$**

Convertendo entre Decimal e Binário

- Entenderam?
- Mais fácil né?
- **Converte o número $0,25_{10}$**

	Product	Integer Part	
$0.25 \times 2 = 0.5$		0	0.01_2
$0.5 \times 2 = 1.0$		1	



(b) $0.25_{10} = 0.01_2$ (exactly)

Convertendo entre Decimal e Binário

- **Converta o número $0,81_{10}$**

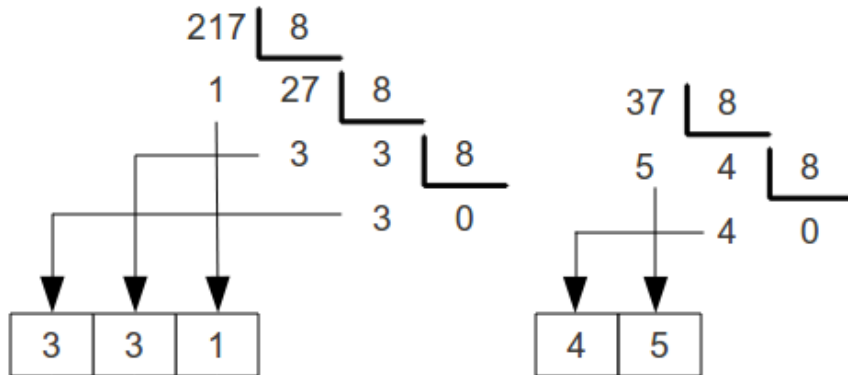
Convertendo entre Decimal e Binário

- Converta o número $0,81_{10}$

Product	Integer Part	
$0.81 \times 2 = 1.62$	1	0.110011 ₂
$0.62 \times 2 = 1.24$	1	
$0.24 \times 2 = 0.48$	0	
$0.48 \times 2 = 0.96$	0	
$0.96 \times 2 = 1.92$	1	
$0.92 \times 2 = 1.84$	1	

(a) $0.81_{10} = 0.110011_2$ (approximately)

Notação Octal



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

Binary 010 101 110 000 001 . 100 110
 Octal 2 5 6 0 1 . 3 6

$(2)_8$ $(7)_8$. $(5)_8$
 $(010)_2$ $(111)_2$. $(101)_2$

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)			

Notação Hexadecimal

- Considere a string binária

110111100001

1101 1110 0001 = DE1₁₆
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.

Notação Hexadecimal

- Binário – Hexadecimal (**FRAÇÃO**)
 - Caso seja necessário, acrescentar o dígito 0 (zero) nas extremidades!
 - 0.101_2 em hexadecimal?

0000 . 101**0**

$$\underbrace{0}_{0.101_2} \cdot \underbrace{A_{16}}_{0.A_{16}}$$

$0.101_2 = 0.A_{16}$

Exercícios

- Converta os números binários para decimal
 - A) 001.10 B) 10.010
 - C) 110.11 D) 1010.101
- Converta os números decimais para binário
 - A) 6.2 B) 54.70
 - B) 629.446 C) 1.053

Relembrando os negativos...

- **Sinal-Magnitude**

- A representação de sinal-magnitude é a mais familiar a nós que utilizamos o sistema numérico de base 10
- Bit à esquerda do número (mais significativo) para indicar se número é positivo ou negativo
- Os bits restantes do número representam a magnitude (ou o valor absoluto). Assim, em um byte com 8 bits, são utilizados 7 bits para representar o valor e um bit para representar o sinal
- Neste caso, o valor pode variar de 0000000 (0) a 1111111 (127), podendo representar números de -127_{10} a $+127_{10}$
- Uma **consequência** desta representação é que existem duas maneiras de representar o **zero**, 00000000 (0) e 10000000 (-0)

Relembrando os negativos...

- **Complemento a dois**

- Os problemas de múltiplas representações de 0 e a necessidade de tratamento com “vai-um” são contornadas pelo complemento a dois
- Em complemento a dois, há apenas um zero (00000000). Se nega um número (negativo ou positivo) invertendo-se todos os bits e, em seguida, adicionando 1 ao resultado
- A adição de de números em complemento para dois é o mesmo processo da adição de números sem-sinal
 - Mesmo hardware!!!
- Um método mais fácil de obter a negação de um número em complemento de dois é o que se segue:

	Exemplo 1	Exemplo 2
1. A partir da direita, encontre o primeiro '1'	0101001	0101100
2. Inverte todos os bits à esquerda deste um	1010111	1010100

Excesso-N

- A representação de Excesso-N usa um número pré-especificado N como um valor de polarização
- Um valor é representado pelo número sem sinal que é N unidades maior do que o valor pretendido
- Assim, 0 (zero) é representado por N , e $-N$ é representado pelo padrão de bits zeros (tudo zerado).
- Esta é uma representação que é agora utilizada principalmente em números de ponto flutuante

Excesso-127 em 8 bits

Valor binário	Interpretação de Excesso-127	Interpretação sem-sinal
00000000	-127	0
00000001	-126	1
...
01111111	0	127
10000000	1	128
...
11111111	+128	255

Decimal	Sem sinal	Sinal-e-magnitude	Complemento para um	Complemento de dois	Excesso-7
+16	—	—	—	—	—
+15	1111	—	—	—	—
+14	1110	—	—	—	—
+13	1101	—	—	—	—
+12	1100	—	—	—	—
+11	1011	—	—	—	—
+10	1010	—	—	—	—
+9	1001	—	—	—	—
+8	1000	—	—	—	1111
+7	0111	0111	0111	0111	1110
+6	0110	0110	0110	0110	1101
+5	0101	0101	0101	0101	1100
+4	0100	0100	0100	0100	1011
+3	0011	0011	0011	0011	1010
+2	0010	0010	0010	0010	1001
+1	0001	0001	0001	0001	1000
+0	—	0000	0000	—	—
0	0000	—	—	0000	0111
-0	—	1000	1111	—	—

Decimal	Sem sinal	Sinal-e- magnitude	Complemento para um	Complemento de dois	Excesso-7
+9	1001	—	—	—	—
+8	1000	—	—	—	1111
+7	0111	0111	0111	0111	1110
+6	0110	0110	0110	0110	1101
+5	0101	0101	0101	0101	1100
+4	0100	0100	0100	0100	1011
+3	0011	0011	0011	0011	1010
+2	0010	0010	0010	0010	1001
+1	0001	0001	0001	0001	1000
+0	—	0000	0000	—	—
0	0000	—	—	0000	0111
-0	—	1000	1111	—	—
-1	—	1001	1110	1111	0110
-2	—	1010	1101	1110	0101
-3	—	1011	1100	1101	0100
-4	—	1100	1011	1100	0011
-5	—	1101	1010	1011	0010
-6	—	1110	1001	1010	0001
-7	—	1111	1000	1001	0000
-8	—	—	—	1000	—
-9	—	—	—	—	—
-10	—	—	—	—	—

Decimal	Sem sinal	Sinal-e- magnitude	Complemento para um	Complemento de dois	Excesso-7
+9	1001	—	—	—	—
+8	1000	—	—	—	1111
+7	0111	0111	0111	0111	1110
+6	0110	0110	0110	0110	1101
+5	0101	0101	0101	0101	1100
+4	0100	0100	0100	0100	1011
+3	0011	0011	0011	0011	1010
+2	0010	0010	0010	0010	1001
+1	0001	0001	0001	0001	1000
+0	—	0000	0000	—	—
0	0000	—	—	0000	0111
-0	—	1000	1111	—	—
-1	—	1001	1110	1111	0110
-2	—	1010	1101	1110	0101
-3	—	1011	1100	1101	0100
-4	—	1100	1011	1100	0011
-5	—	1101	1010	1011	0010
-6	—	1110	1001	1010	0001
-7	—	1111	1000	1001	0000
-8	—	—	—	1000	—
-9	—	—	—	—	—
-10	—	—	—	—	—

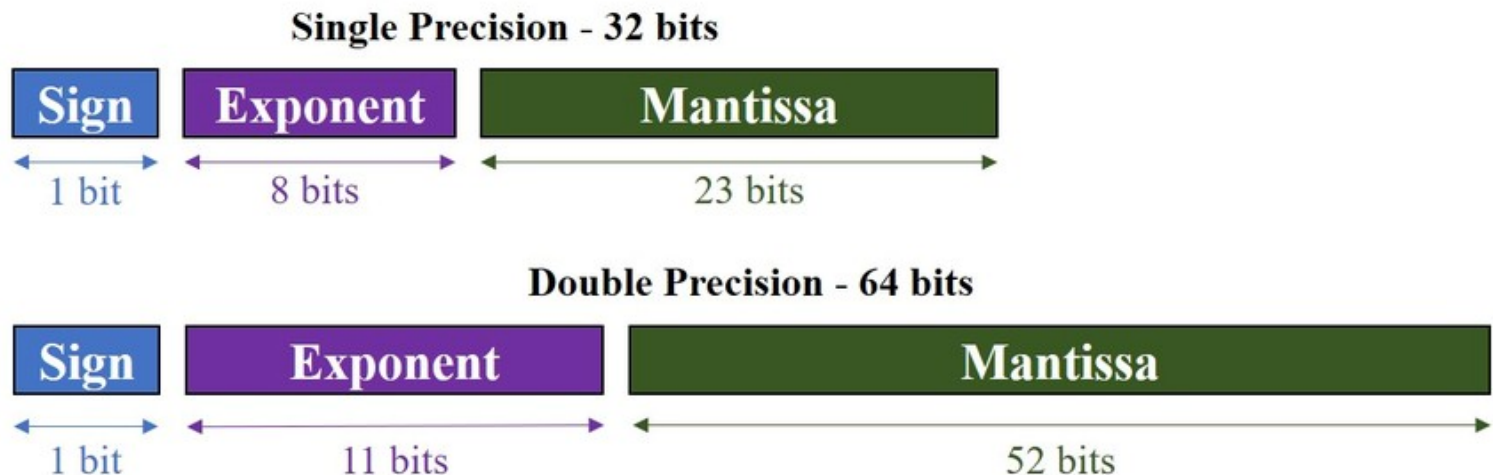
Decimal	Sem sinal	Sinal-e-magnitude	Complemento para um	Complemento de dois	Excesso-7
+9	1001	—	—	—	—
+8	1000	—	—	—	1111
+7	0111	0111	0111	0111	1110
+6	0110	0110	0110	0110	1101
+5	0101	0101	0101	0101	1100
+4	0100	0100	0100	0100	1011
+3	0011	0011	0011	0011	1010
+2	0010	0010	0010	0010	1001
+1	0001	0001	0001	0001	1000
+0	—	0000	0000	—	—
0	0000	—	—	0000	0111
-0	—	1000	1111	—	—
-1	—	1001	1110	1111	0110
-2	—	1010	1101	1110	0101
-3	—	1011	1100	1101	0100
-4	—	1100	1011	1100	0011
-5	—	1101	1010	1011	0010
-6	—	1110	1001	1010	0001
-7	—	1111	1000	1001	0000
-8	—	—	—	1000	—
-9	—	—	—	—	—
-10	—	—	—	—	—

Decimal	Sem sinal	Sinal-e-magnitude	Complemento para um	Complemento de dois	Excesso-7
+9	1001	—	—	—	—
+8	1000	—	—	—	1111
+7	0111	0111	0111	0111	1110
+6	0110	0110	0110	0110	1101
+5	0101	0101	0101	0101	1100
+4	0100	0100	0100	0100	1011
+3	0011	0011	0011	0011	1010
+2	0010	0010	0010	0010	1001
+1	0001	0001	0001	0001	1000
+0	—	0000	0000	—	—
0	0000	—	—	0000	0111
-0	—	1000	1111	—	—
-1	—	1001	1110	1111	0110
-2	—	1010	1101	1110	0101
-3	—	1011	1100	1101	0100
-4	—	1100	1011	1100	0011
-5	—	1101	1010	1011	0010
-6	—	1110	1001	1010	0001
-7	—	1111	1000	1001	0000
-8	—	—	—	1000	—
-9	—	—	—	—	—
-10	—	—	—	—	—

IEEE-754

- O padrão IEEE de ponto flutuante (**IEEE-754**), define o campo de **expoente** de um número de 32 bits de precisão simples como um campo de 8 bits na representação de Excesso-127
- O **expoente** de um número de 64 bits de precisão dupla é um de campo de 11 bits com representação de Excesso-1023.

IEEE 754 Floating Point Standard



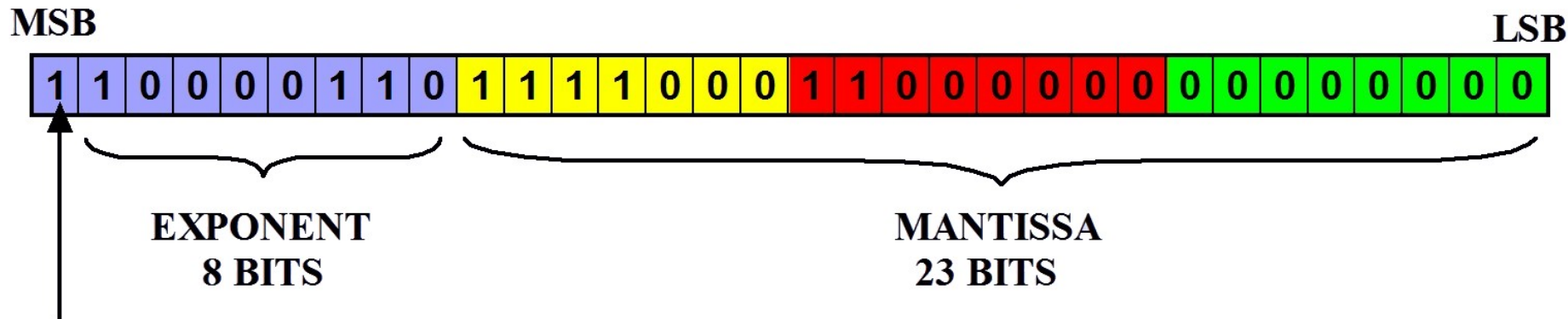
IEEE-745

IEEE 754 Converter, 2024-02

	Sign	Exponent	Mantissa
Value:	-1	2^7	$1 + 0.943359375$
Encoded as:	1	134	7913472
Binary:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
Decimal Representation	<input type="text" value="-248.75"/>		
Value actually stored in float:	<input type="text" value="-248.75"/>		
Error due to conversion:	<input type="text" value="0"/>		
Binary Representation	<input type="text" value="11000011011110001100000000000000"/>		
Hexadecimal Representation	<input type="text" value="c378c000"/>		

1

-1



SIGN BIT
1= NEGATIVE
0=POSITIVE

EXAMPLE: -248.75
HEXADECIMAL: C3 78 C0 00

IEEE-745

IEEE 754 Converter, 2024-02

	Sign	Exponent	Mantissa
Value:	-1	2^7	$1 + 0.943359375$
Encoded as:	1	134	7913472
Binary:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
Decimal Representation	<input type="text" value="-248.75"/>		
Value actually stored in float:	<input type="text" value="-248.75"/>		
Error due to conversion:	<input type="text" value="0"/>		
Binary Representation	<input type="text" value="11000011011110001100000000000000"/>		
Hexadecimal Representation	<input type="text" value="c378c000"/>		

Calculadora IEEE-745 online

<https://www.h-schmidt.net/FloatConverter/IEEE754.html>

Exercício

Dado que um computador trabalha em excesso-127, os números abaixo representam quais valores em decimal?

- 11001100
- 10101010
- 01100110
- 00110011

Exercício

Dado que um computador trabalha em excesso-127, os números abaixo representam quais valores em decimal?

- 11001100
- 10101010
- 01100110
- 00110011

Solução: Converta de binário para decimal e subtraia o excesso, no caso 127.

Exercício

Dado que um computador trabalha em excesso-127, os números abaixo representam quais valores em decimal?

- $11001100 \rightarrow (1 \times 2^7) + (1 \times 2^6) + (1 \times 2^3) + (1 \times 2^2) =$
 $128 + 64 + 8 + 4 = 204$
 $204 - 127 = 77$
- 10101010
- 01100110
- 00110011

Solução: Converta de binário para decimal e subtraia o excesso, no caso 127.

Exercício

Agora faça o inverso, converta os números decimais abaixo para excesso-127.

- 33
- 87
- 15
- 65

Exercício

Agora faça o inverso, converta os números decimais abaixo para excesso-127.

- 33
- 87
- 15
- 65

Solução: Some o excesso, no caso 127, e depois converta de decimal para binário.

Exercício

Agora faça o inverso, converta os números decimais abaixo para excesso-127.

- $33 \rightarrow 33 + 127 = 160$
 $160_{10} = 10100000_2 \quad (127 + 33)$
- 87
- 15
- 65

Solução: Some o excesso, no caso 127, e depois converta de decimal para binário.

DS011

Binário BCD e Codificação de Caracteres

Prof. Clausius Duque Reis
clausius.reis@ufpr.br

Códigos Binários

- **Sistema numérico decimal**
 - Conveniente para os seres humanos.
- **Sistema numérico binário**
 - Conveniente para computadores.
 - (BEM) menos conveniente para os seres humanos.

Códigos Binários

- **Exemplo:**

- 1010011_2 em decimal ???
- Processo de conversão simples, porém tedioso → consome muito tempo.

Códigos Binários

- **Exemplo:**
 - 1010011_2 em decimal ???
 - Processo de conversão simples, porém tedioso → consome muito tempo.
- **BCD – Forma especial de código binário MAIS compatível com o sistema decimal.**

Código BCD 8421

- **BCD – Binary Coded Decimal**

Código BCD 8421

- **BCD – Binary Coded Decimal**
- **Binário Codificado em Decimal.**
 - Não é um sistema de numeração!

Código BCD 8421

- **BCD – Binary Coded Decimal**
- **Binário Codificado em Decimal.**
 - Não é um sistema de numeração!
- **Representa os dígitos decimais de 0 a 9 com um código binário de 4 dígitos.**

Código BCD 8421

- **BCD – Binary Coded Decimal**
- **Binário Codificado em Decimal.**
 - Não é um sistema de numeração!
- **Representa os dígitos decimais de 0 a 9 com um código binário de 4 dígitos.**
- **Usa o sistema de pesos posicionais 8421 do código binário puro**
 - $dB \times 2^3 + dB \times 2^2 + dB \times 2^1 + dB \times 2^0$
 - $dB \times 8 + dB \times 4 + dB \times 2 + dB \times 1$

Código BCD 8421

- **BCD – Binary Coded Decimal**
- **Exemplo: Decimal \rightarrow BCD**
 - 834_{10} em BCD =

Código BCD 8421

- **BCD – Binary Coded Decimal**
- **Exemplo: Decimal \rightarrow BCD**
 - 834_{10} em BCD = 1000 0011 0100

Código BCD 8421

- **BCD – Binary Coded Decimal**
- **Exemplo: Decimal \rightarrow BCD**
 - 834_{10} em BCD = 1000 0011 0100
 - $0,764_{10}$ em BCD =

Código BCD 8421

- **BCD – Binary Coded Decimal**
- **Exemplo: Decimal \rightarrow BCD**
 - 834_{10} em BCD = 1000 0011 0100
 - $0,764_{10}$ em BCD = 0,0111 0110 0100
- **Exemplos: BCD \rightarrow Decimal**
 - 0110 0010 1000,1001 0101 0100 =

Código BCD 8421

- **BCD – Binary Coded Decimal**
- **Exemplo: Decimal \rightarrow BCD**
 - 834_{10} em BCD = 1000 0011 0100
 - $0,764_{10}$ em BCD = 0,0111 0110 0100
- **Exemplos: BCD \rightarrow Decimal**
 - 0110 0010 1000,1001 0101 0100 = 628,954

- **Vantagens BCD**

- Simples manipulação e conversão

- **Desvantagens**

- Menos eficiente que o código binário puro. Utiliza maior número de bits.
- Maior complexidade dos circuitos, maior consumo de energia, ...
- As operações aritméticas consomem mais tempo.

Código BCD 8421

- **Tabela de conversão**

- Decimal
- BCD 8421
- Binário puro

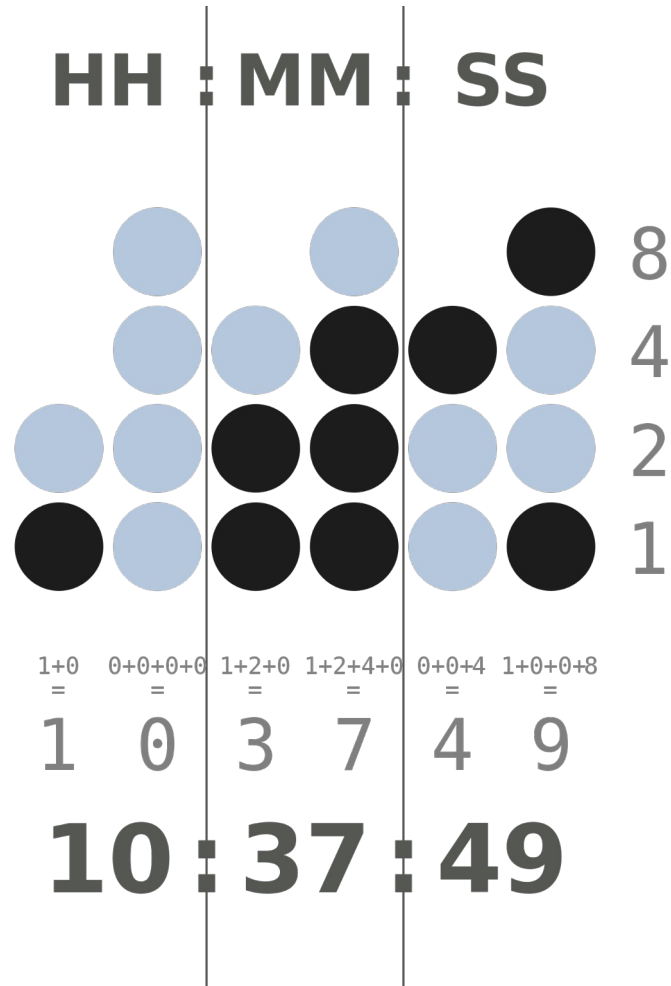
DECIMAL	BCD 8421	BINÁRIO
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1000
9	1001	1001
10	0001 0000	1010
11	0001 0001	1011
12	0001 0010	1100
13	0001 0011	1101
14	0001 0100	1110
15	0001 0101	1111

- **BCD – Binary Coded Decimal**
 - A BIOS em muitos computadores armazena a data e a hora utilizando BCD devido a utilização do chip MC6818. Essa forma é facilmente convertida em ASCII para os displays.

- **BCD – Binary Coded Decimal**
 - A BIOS em muitos computadores armazena a data e a hora utilizando BCD devido a utilização do chip MC6818. Essa forma é facilmente convertida em ASCII para os displays.
 - O Atari (família 8-bit) utilizava BCD para implementar os algoritmos de ponto flutuante.

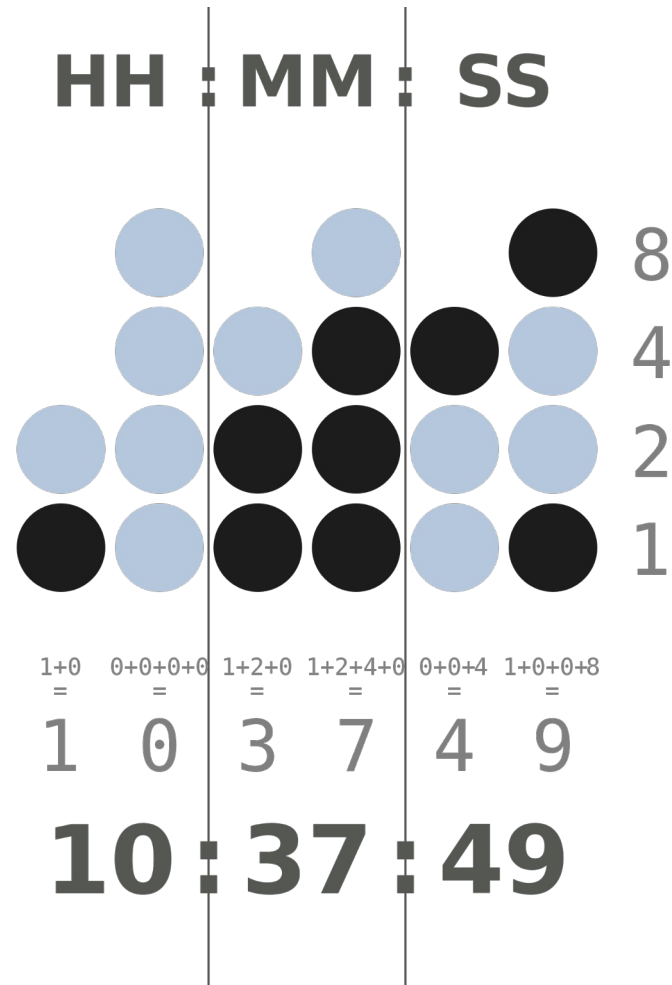
- **BCD – Binary Coded Decimal**
 - A BIOS em muitos computadores armazena a data e a hora utilizando BCD devido a utilização do chip MC6818. Essa forma é facilmente convertida em ASCII para os displays.
 - O Atari (família 8-bit) utilizava BCD para implementar os algoritmos de ponto flutuante.
 - Alguns modelos do Playstation 3 também armazenam a data e a hora utilizando BCD.

Código BCD 8421



Código BCD 8421

**Pausa para
Demonstração**



Código BCD 8421

Conversão:

- **Decimal → BCD**
 - Simples e direta.

Conversão:

- **Decimal → BCD**
 - Simples e direta.
- **Binário (puro) → BCD**
 - Converte primeiro para DECIMAL.
 - Na sequência de Decimal para BCD.

Código BCD 8421

Conversão:

- **Decimal \rightarrow BCD**
 - Simples e direta.
- **Binário (puro) \rightarrow BCD**
 - Converte primeiro para DECIMAL.
 - Na sequência de Decimal para BCD.
- **EXEMPLO:**
 - $1011,01_2 =$
 - $11,25_{10} =$

Código BCD 8421

Conversão:

- **Decimal → BCD**
 - Simples e direta.
- **Binário (puro) → BCD**
 - Converte primeiro para DECIMAL.
 - Na sequência de Decimal para BCD.
- **EXEMPLO:**
 - $1011,01_2 = (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) + (0 \times 2^{-1}) + (1 \times 2^{-2}) =$
 - $11,25_{10} =$

Código BCD 8421

Conversão:

- **Decimal → BCD**
 - Simples e direta.
- **Binário (puro) → BCD**
 - Converte primeiro para DECIMAL.
 - Na sequência de Decimal para BCD.
- **EXEMPLO:**
 - $1011,01_2 = (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) + (0 \times 2^{-1}) + (1 \times 2^{-2}) = 8 + 0 + 2 + 1 + 0 + 0,25 = 11,25_{10}$
 - $11,25_{10} =$

Código BCD 8421

Conversão:

- **Decimal → BCD**
 - Simples e direta.
- **Binário (puro) → BCD**
 - Converte primeiro para DECIMAL.
 - Na sequência de Decimal para BCD.
- **EXEMPLO:**
 - $1011,01_2 = (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) + (0 \times 2^{-1}) + (1 \times 2^{-2}) = 8 + 0 + 2 + 1 + 0 + 0,25 = 11,25_{10}$
 - $11,25_{10} = 0001\ 0001.0010\ 0101_2$

Conversão:

- **Converte de BCD para Binário puro**
 - 1) Converte BCD para decimal
 - 2) Decimal é convertido para binário
- **Exemplo:**
 - **1001 0110,0110 0010 0101**

Código BCD 8421

Conversão:

- **Converte de BCD para Binário puro**

- 1) Converte BCD para decimal
- 2) Decimal é convertido para binário

- **Exemplo:**

- **1001 0110,0110 0010 0101 = 96,625**

Inteiro	Resto	Posição	Fração	Inteiro	Posição
$96 \div 2 = 48$	0	-> LSB	$0,625 \times 2 = 1,25 = 0,25$	1	<- MSB
$48 \div 2 = 24$	0		$0,250 \times 2 = 0,50 = 0,50$	0	
$24 \div 2 = 12$	0		$0,500 \times 2 = 1,00 = 0$	0	<- LSB
$12 \div 2 = 06$	0				
$06 \div 2 = 03$	0				
$03 \div 2 = 01$	1				
$01 \div 2 = 00$	1	<- MSB			
$96_{10} = 1100000_2$			$0,625_{10} = 0.101_2$		

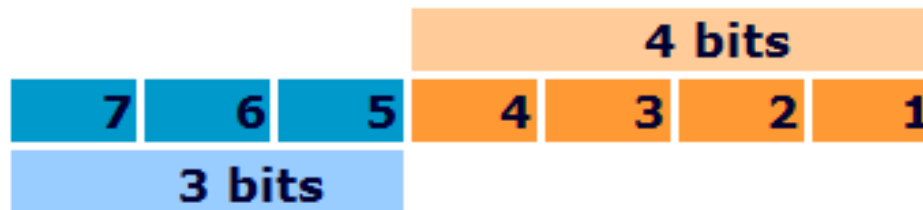
$$96,625_{10} = 96_{10} + 0,625_{10} = 1100000_2 + 0.101_2 = 1100000.101_2$$

Código ASCII

- “**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange”
 - Forma especial de código binário.
 - Largamente utilizado.
 - 7 bits: pode-se representar um total de $2^7 = 128$ caracteres diferentes
 - Números decimais de **0 até 9**
 - **Letras maiúsculas e minúsculas do alfabeto**
 - Outros **caracteres especiais** usados para pontuação e controle de dados.

Código ASCII - Conversão

- Composto por 2 grupos:
 - Um de 4 bits e outro de 3 bits.
- O grupo de 4 bits está a direita e o bit 1 é o LSB.
 - LSB: Bit Menos Significativo.
 - MSB: Bit Mais Significativo.



Código ASCII - Conversão

NULL	Null	DLE	Data Link Escape
SOH	Start of Heading	DC1	Device Control 1
STX	Start of Text	DC2	Device Control 2
ETX	End of Text	DC3	Device Control 3
EOT	End of Transmission	DC4	Device Control 4
ENQ	Enquiry	NAK	Negative Acknowledge
ACK	Acknowledge	SYN	Synchronous Idle
BEL	Bell (audible signal)	ETB	End Transmission Block
BS	Backspace	CAN	Cancel
HT	Horizontal Tabulação (punched card skip)	EM	End of Medium
		SUB	Substitute
LF	Line Feed	ESC	Escape
VT	Vertical Tabulation	FS	File Separator
FF	Form Feed	GS	Group Separator
CR	Carriage Return	RS	Record Separator
SO	Shift Out	US	Unit Separator
SI	Shift In	DEL	Delete
SP	Space (blank)		

	coluna								
	bits	0	1	2	3	4	5	6	7
linha	7654321	000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SP	0	@	P	`	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	\$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(8	H	X	h	x
9	1001	HT	EM)	9	I	Y	i	y
10	1010	LF	SUB	*	:	J	Z	j	z
11	1011	VT	ESC	+	;	K	[k	{
12	1100	FF	FS	,	<	L	\	l	
13	1101	CR	GS	-	=	M]	m	}
14	1110	SO	RS	.	>	N	^	n	~
15	1111	SI	US	/	?	O	_	o	DEL

Código ASCII - Conversão

- Exemplo: Código ASCII para a letra **L** é **1001100**
- Localizado na **coluna 4, linha 12**
- O grupo de 3 bits é 100 e o grupo de 4 bits é 1100.
 - Código ASCII: 100 1100.

Código ASCII - Conversão

- No código ASCII de 7 bits, o oitavo bit é geralmente usado como um **bit de paridade**
 - Para determinar se o dado (caractere) foi transmitido corretamente.
 - Determinado pelo tipo de paridade desejado.
 - Paridade par → a soma de todos os 1's, incluindo o bit de paridade, é um número par.
 - EXEMPLO:
 - Caractere G – código ASCII é 1000111
 - 4 bits UM – O bit de paridade é 0 → 01000111

Exercícios

1) Converta de Decimal para BCD

- a) 1234
- b) 235,876

2) Converta de BCD para Decimal

- a) 0010 0011 0101 1000 0000
- b) 0001 0010 1001. 0011 0110 0000 0010

3) Converta de Binário para BCD

- a) 01110110110
- b) 011000111.011101

Exercícios

1) Converta de BCD para Binário

a) 1001 0100 0011 0101 0110

b) 0111 1000 1001 0001.0000 0101 1000

2) Converta a string para ASCII (7 bits)

a) Hello Computer!

b) Linux is better

c) Lord Darth Vader

3) Inclua o bit de paridade na resposta do exercício anterior, utilizando 8 bits