

# Aula 5

## Ponto Flutuante

Prof. Adalberto

# Alguns detalhes que ficaram Pendentes

- 1 Byte = 8 bits
- 1 Byte = 2 Nibbles; 1 Nibble = 4 bits
- MSB – Bit Mais Significativo
- LSB – Bit Menos Significativo
- Unidades Múltiplas
  - ✓ 1 Kilo (1K) =  $2^{10} = 1.024$
  - ✓ 1 Mega (1M) =  $2^{20} = 1.048.576$
  - ✓ 1 Giga (1G) =  $2^{30} = 1.073.741.824$

# Ponto Flutuante

- Números com fração
- Como representar em binário números fracionários?
  - ✓  $6,625_{10}$
  - ✓ Parte inteira:  $6_{10} = 110_2$
  - ✓ Parte fracionária:  $0,625_{10} = ?$

# Obtenção de Fração Binária

- Multiplicar a parte fracionária por 2 e separar a parte inteira do resultado até obter 0 na parte fracionária ou até chegar a um limite de bits
  - ✓  $0,625 \times 2 = 1,25$
  - ✓  $0,25 \times 2 = 0,5$
  - ✓  $0,5 \times 2 = 1,0$

# Obtenção de Fração Binária

- Multiplicar a parte fracionária por 2 e separar a parte inteira do resultado até obter 0 na parte fracionária ou até chegar a um limite de bits

✓  $0,625 \times 2 = 1,25$

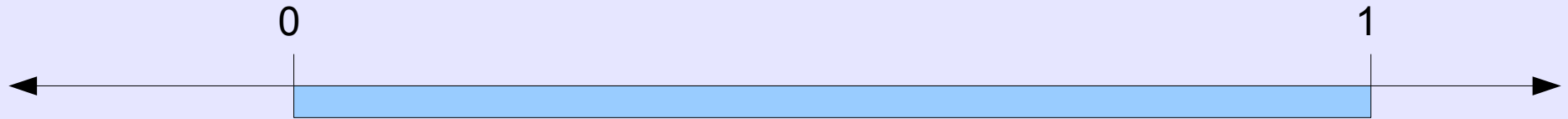
✓  $0,25 \times 2 = 0,5$

✓  $0,5 \times 2 = 1,0$

110,101<sub>2</sub>

# Conversão para Decimal

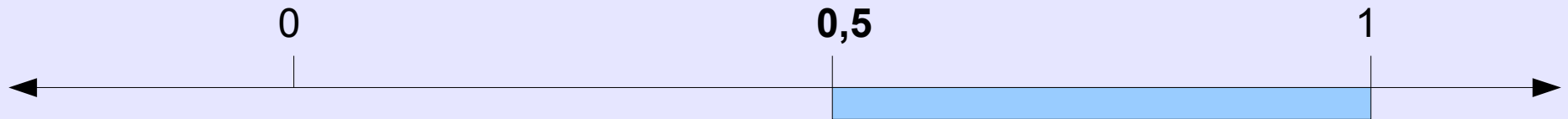
0,110101



# Conversão para Decimal

0,110101

$$1 \times 2^{-1} = 0,5$$

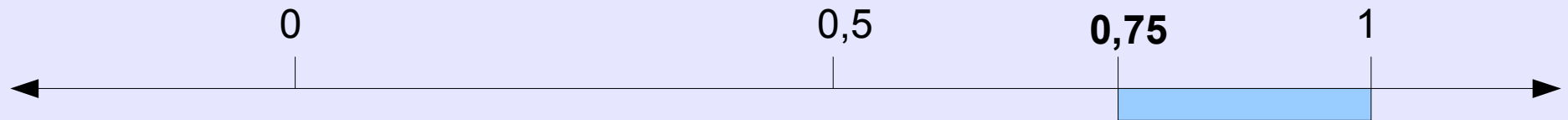


# Conversão para Decimal

0,110101

$$1 \times 2^{-1} = 0,5$$

$$1 \times 2^{-2} = 0,25$$



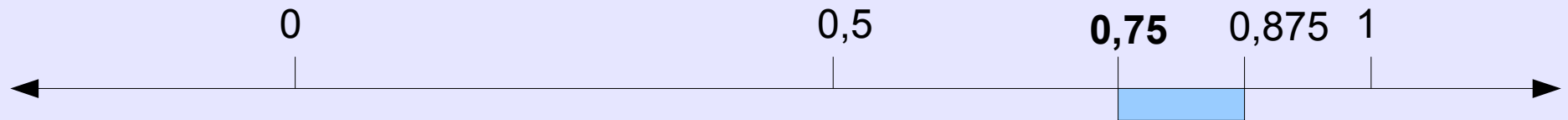
# Conversão para Decimal

0,110101

$$1 \times 2^{-1} = 0,5$$

$$1 \times 2^{-2} = 0,25$$

$$0 \times 2^{-3} = 0 \times 0,125$$



# Conversão para Decimal

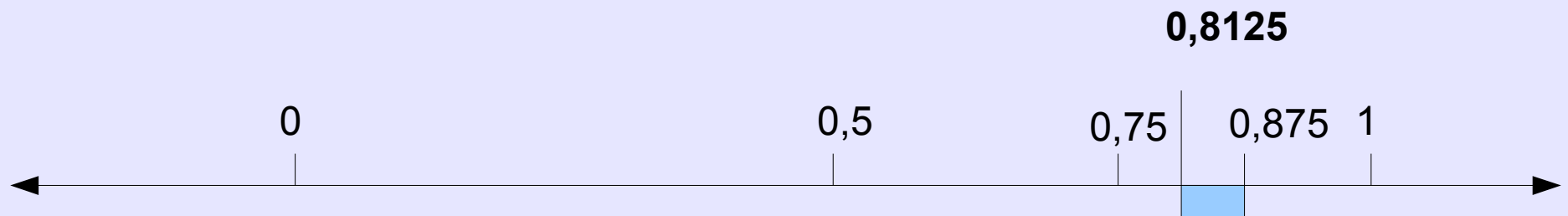
0,110101

$$1 \times 2^{-1} = 0,5$$

$$1 \times 2^{-2} = 0,25$$

$$0 \times 2^{-3} = 0$$

$$1 \times 2^{-4} = 0,0625$$



# Conversão para Decimal

0,110101

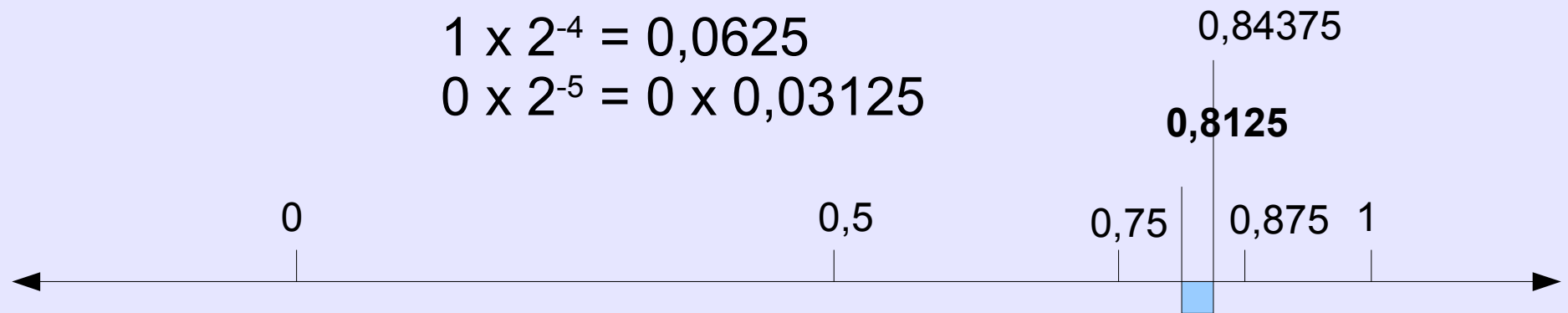
$$1 \times 2^{-1} = 0,5$$

$$1 \times 2^{-2} = 0,25$$

$$0 \times 2^{-3} = 0$$

$$1 \times 2^{-4} = 0,0625$$

$$0 \times 2^{-5} = 0 \times 0,03125$$



# Conversão para Decimal

0,110101

$$1 \times 2^{-1} = 0,5$$

$$1 \times 2^{-2} = 0,25$$

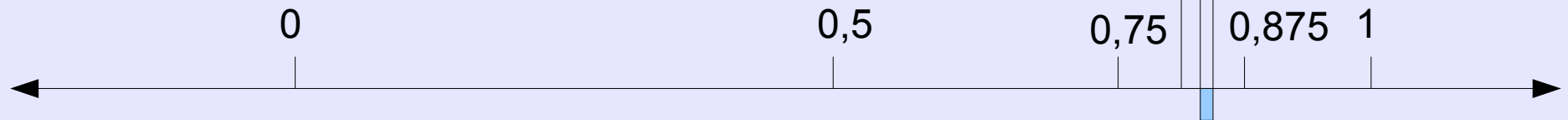
$$0 \times 2^{-3} = 0$$

$$1 \times 2^{-4} = 0,0625$$

$$0 \times 2^{-5} = 0$$

$$1 \times 2^{-6} = 0,015625$$

0,828125



# Algumas potências de dois

✓  $2^{-1} = 0,5$

✓  $2^{-2} = 0,25$

✓  $2^{-3} = 0,125$

✓  $2^{-4} = 0,0625$

✓  $2^{-5} = 0,03125$

✓  $2^{-6} = 0,015625$

✓  $2^{-7} = 0,0078125$

✓  $2^{-8} = 0,00390625$

✓  $2^{-9} = 0,00195312$

✓  $2^0 = 1$

✓  $2^1 = 2$

✓  $2^2 = 4$

✓  $2^3 = 8$

✓  $2^4 = 16$

✓  $2^5 = 32$

✓  $2^6 = 64$

✓  $2^7 = 128$

✓  $2^8 = 256$

# Prática

- Converta os valores abaixo para a base 2:
  - ✓ 0,8125
  - ✓ 15,9375
  - ✓ 5,796875
- Converta os valores abaixo para a base 10:
  - ✓  $111,010011001_2$
  - ✓  $0,11011_2$
  - ✓  $11,001001001_2$

# Notação Científica

- Em algumas áreas do conhecimento usa-se valores muito grandes ou muito pequenos
- Números decimais podem ser representados em notação científica
  - ✓ 976.000.000.000.000 pode ser representado como  $9,76 \times 10^{14}$  e
  - ✓ 0,000000000000000976 pode ser representado como  $9,76 \times 10^{-14}$

# Notação Científica

- Esta mesma abordagem pode ser adotada por números binários.
- Podemos representar um número qualquer na forma

$$\pm M \times B^{\pm E}$$

- ✓ Sinal – Mais ou Menos
- ✓ Mantissa
- ✓ Expoente
- ✓ A Base é implícita, então não precisa ser armazenada

# Números Normalizados

- Número Normalizado em binário tem o seguinte formato:

$$\pm 1.bbb\dots b \times 2^{\pm E}$$

- Peguemos um exemplo

- ✓  $5,796875 = 101,110011_2$

- ✓  $101,110011 = 1,01110011_2 \times 2^2$

# Prática

- Represente os números abaixo em binários normalizados
  - ✓ 0,8125
  - ✓ 15,9375
  - ✓ 5,796875
  - ✓  $111,010011001_2$
  - ✓  $0,11011_2$
  - ✓  $11,001001001_2$

# Ponto Flutuante

- Outro Exemplo:

- ✓  $1.712.128 = 1101000100000000000000$

- ✓  $1101000100000000000000_2 = 1,1010001_2 \times 2^{20}$

$$1.1010001 \times 2^{10100} = 1.6328125 \times 2^{20}$$

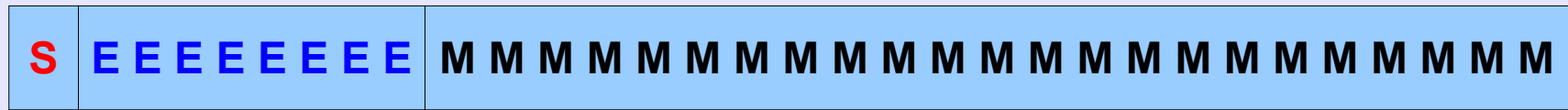
$$-1.1010001 \times 2^{10100} = -1.6328125 \times 2^{20}$$

$$1.1010001 \times 2^{-10100} = 1.6328125 \times 2^{-20}$$

$$-1.1010001 \times 2^{-10100} = -1.6328125 \times 2^{-20}$$

# Formato do Ponto Flutuante

- O bits ficam distribuídos da seguinte forma:



$$1.1010001 \times 2^{10100} = 0 \ 10010011 \ 101000100000000000000000$$

$$-1.1010001 \times 2^{10100} = 1 \ 10010011 \ 101000100000000000000000$$

$$1.1010001 \times 2^{-10100} = 0 \ 01101011 \ 101000100000000000000000$$

$$-1.1010001 \times 2^{-10100} = 1 \ 01101011 \ 101000100000000000000000$$

# Expoente do Ponto Flutuante

- Para calcular o campo expoente deve-se somar o valor do expoente (expoente real) a um excesso
- O excesso do expoente é dado por:
  - ✓  $2^{k-1}-1$ , onde  $k$  é o número de bits do campo expoente
  - ✓ Exemplo, se o expoente é de 8bits, o excesso será  $2^{8-1}-1 = 127$
  - ✓ O excesso será sempre  $011111\dots1_2$

# Expoente do Ponto Flutuante

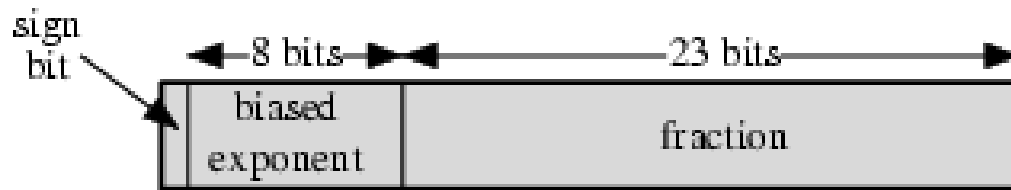
- Então o expoente real  $20$  ( $00010100_2$ ) será
  - ✓  $20 + 127 = 147$  ( $10010011_2$ )
- Se o expoente real é negativo  $-20$  ( $11101010_2$ )
  - ✓  $-20 + 127 = 107 = (01101011_2)$
- Se o expoente real é zero então o campo expoente será o próprio excesso
  - ✓  $0 + 127 = 127 = (01111111_2)$

# Mantissa

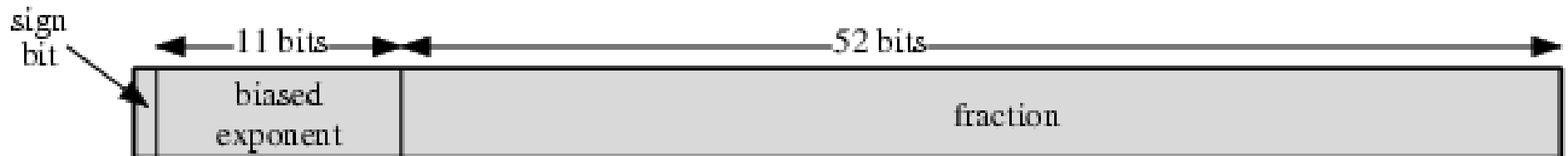
- O dígito “1” antes da vírgula na mantissa não é representado no campo mantissa do número, então a mantissa
  - ✓ 1,1011 será representada apenas com 1011
- O valor zero é tratado de forma especial. Todos os bits de todos os campos são zeros

# Padrão IEEE-754

- Números de precisão simples (32bits) e precisão dupla (64bits)

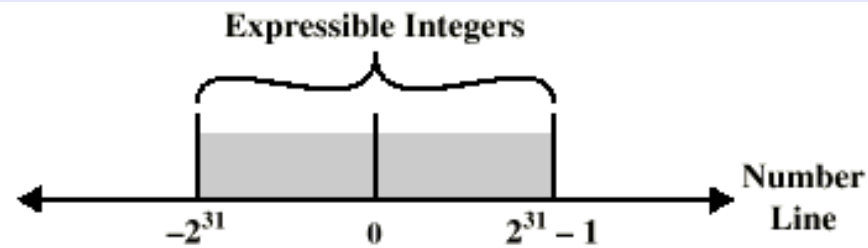


(a) Single format

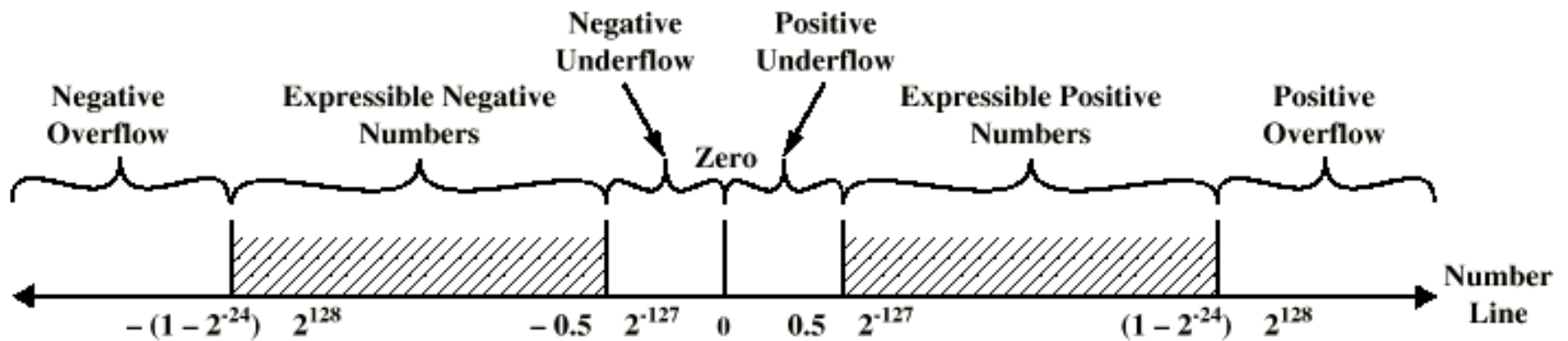


(b) Double format

# Números Expressivos



(a) Two's Complement Integers



(b) Floating-Point Numbers

# Prática

- Represente os números abaixo no formato IEEE-754 com precisão simples e dupla
  - ✓ 0,8125
  - ✓ 15,9375
  - ✓ 5,796875
  - ✓  $111,010011001_2$
  - ✓  $0,11011_2$
  - ✓  $11,001001001_2$

# Pro Lar

- Converta os seguintes números para o formato IEEE-754 de precisão simples
  - ✓ 9
  - ✓  $5/32$
  - ✓  $-5/32$
  - ✓ 6,125

## ...mais

- Converta os números abaixo em formato IEEE-754 de hexadecimal para decimal
  - ✓ 0x42E48000
  - ✓ 0x3F880000
  - ✓ 0x00800000
  - ✓ 0xC7F00000