

CIRCUITOS DIGITAIS

SISTEMAS NUMÉRICOS

Prof. Marcelo Grandi Mandelli

`mgmandelli@unb.br`

Representação de Números Decimais

- ❑ Sistema decimal (**base 10**) → duas regras básicas:
 - Usamos **10** símbolos: 0,1,2,3,4,5,6,7,8 e 9.
 - O valor de cada símbolo depende de sua posição.
 - Exemplo:

$$9845 = 9 \times 10^3 + 8 \times 10^2 + 4 \times 10^1 + 5 \times 10^0$$


Representação de Números Binários

- ❑ Sistema binário (**base 2**) → usamos as mesmas duas regras:
 - **2** símbolos → 0 e 1 → *bits* (*binary digits*)
 - O valor de cada símbolo depende de sua posição.
 - Exemplo:

$$26_{10} = 11010_2 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

Pesos dos Números Binários

- Pesos aumentar da **direita para a esquerda**



Peso:	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Número Binário	1	1	0	1	1	0	1
	↓ MSB						↓ LSB


- (LSB – *least significant bit*) → bit menos significativo, com peso 2^0 .
- (MSB – *most significant bit*) → bit mais significativo, com peso 2^{n-1} , sendo **n** o tamanho da palavra de bits. (No exemplo exemplo **n = 7**)

Pesos dos Números Binários



■ **CAUIDADO** → ENDIANNESS

■ *Exemplo: 110*


■ *Big Endian*





Peso:	2^2	2^1	2^0
Número Binário	1	1	0

 **MSB**  **LSB**

■ *Little Endian*



Peso:	2^0	2^1	2^2
Número Binário	0	1	1

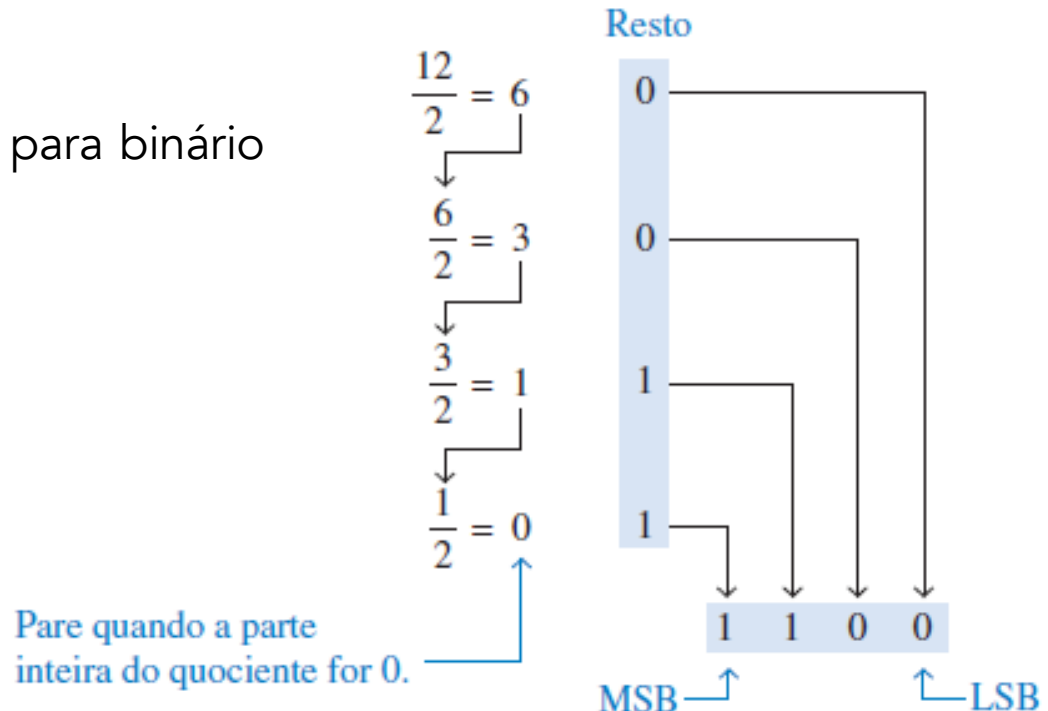
 **LSB**  **MSB**

Conversão Decimal – Binário

Método da Divisão Sucessiva

1. Divide-se o número por 2
2. O resto é o bit menos significativo (LSB)
3. O quociente (inteiro) é dividido novamente por 2
4. O resto é o próximo bit menos significativo
5. Repete-se o processo até que o quociente seja 0

Exemplo → Converter 12 para binário



Representação de Números Binários

□ Com n bits podemos representar até 2^n números

■ $n = 5 \rightarrow 2^5 \rightarrow 32 \text{ números (0 a 31)}$

■ $n = 6 \rightarrow 2^6 \rightarrow 64 \text{ números (0 a 63)}$

Parte fracionária

□ Pesos da parte fracionária

- diminuem da esquerda para a direita
- potência negativa

→ Exemplo Base 10:

$$1984,56 = 1 \times 10^3 + 9 \times 10^2 + 8 \times 10^1 + 4 \times 10^0 + 5 \times 10^{-1} + 6 \times 10^{-2}$$

→ Exemplo Base 2:

$$\begin{aligned} 100,11_2 &= 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} = \\ &= 4 + 0 + 0 + 1/2 + 1/4 = \\ &= 4 + 0 + 0 + 0,5 + 0,25 \\ &= 4,75_{10} \end{aligned}$$

Outras bases

- Podemos usar qualquer base (radix) para nossa representação numérica
 - Por exemplo, podemos usar base 3, 7, 42, etc..
 - Os antigos babilônios utilizavam base 60 e, por isso, temos 60 minutos em uma hora e 360 graus em um círculo.
- De todas as bases, além das bases 2 e 10, duas delas são mais importantes: as bases 8 (octal) e 16 (hexadecimal).

Representação de Números base K

□ Base k

- Usamos k símbolos.
- O valor de cada símbolo depende de sua posição.
- Exemplo:

$$D_3 D_2 D_1 D_0, D_{-1} D_{-2} D_{-3} \text{ } k = \\ D_3 \times k^3 + D_2 \times k^2 + D_1 \times k^1 + D_0 \times k^0 + D_{-1} \times k^{-1} + D_{-2} \times k^{-2} + D_{-3} \times k^{-3}$$

Sistema Octal

□ Sistema Octal → Base 8

■ Usamos 8 símbolos → 0, 1, 2, 3, 4, 5, 6, 7

■ Exemplo:

$$\begin{aligned} 171_8 &= 1 \times 8^2 + 7 \times 8^1 + 1 \times 8^0 \\ &= 1 \times 64 + 7 \times 8 + 1 \times 1 \\ &= 64 + 56 + 1 \\ &= 121_{10} \end{aligned}$$

Sistema Octal

- Fácil conversão **Octal** \leftrightarrow **Binário**
 - dígito octal \rightarrow número binário com **3 bits**

Binário			Dígito Octal
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Sistema Octal

Para converter **binário para octal** agrupa-se os bits de 3 em 3 da **direita para a esquerda**

■ Exemplos Binário → Octal

100110011₂
 ↓ ↓ ↓
 463₈

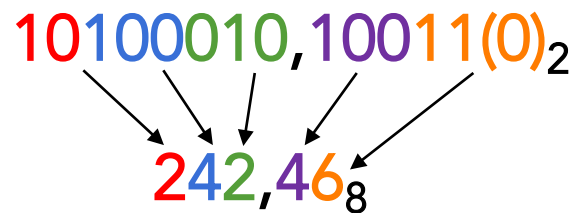
10100010₂
 ↓ ↓ ↓
 242₈

Binário			Digito Octal
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Parte Fracionária

□ Binário → Octal

- Se houver parte fracionária, agrupa-se de 3 em 3 bits **a partir da vírgula para a esquerda**
- Exemplo → 10100010,10011

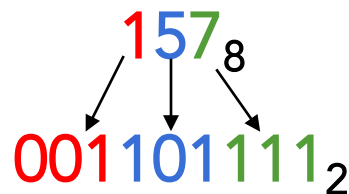


Sistema Octal

□ Fácil conversão Octal \leftrightarrow Binário

- Um dígito octal corresponde a um número binário com **3 bits**

- Exemplo Octal \rightarrow Binário



Binário			Dígito Octal
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Sistema Hexadecimal

□ Sistema Hexadecimal → Base 16

■ Usamos 16 símbolos → 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

■ Para converter para decimal:

HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
DEC	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

■ Exemplo:

$$\text{FACA}_{16} = 15 \times 16^3 + 10 \times 16^2 + 12 \times 16^1 + 10 \times 16^0 = 64202_{10}$$

Sistema Hexadecimal

□ Fácil conversão Hexadecimal \leftrightarrow Binário

- Um dígito hexadecimal corresponde a um número binário com **4 bits**

Binário				Dígito Hexadec.
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

Sistema Hexadecimal

Para converter binário para hexadecimal agrupa-se os bits de 4 em 4 da **direita para a esquerda**

■ Exemplos Binário → Hexadec.

100011110101₂
8F5₁₆

1010100001₂
2A1₁₆

Binário				Dígito Hexadec.
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

Parte Fracionária

□ Binário → Hexadecimal

- Se houver parte fracionária, agrupa-se de 4 em 4 bits a partir da vírgula para a esquerda
- Exemplo → 10100010,10011

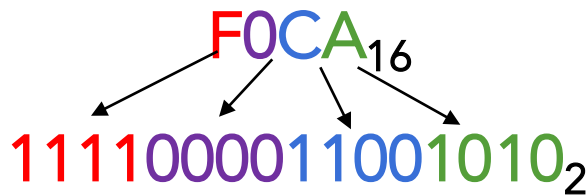
$$10100010,10011(000)_2$$
$$\swarrow \quad \searrow \quad \swarrow \quad \searrow$$
$$A2,98_{16}$$

Sistema Hexadecimal

□ Fácil conversão Hexadecimal \leftrightarrow Binário

- Um dígito hexadecimal corresponde a um número binário com **4 bits**

- Exemplo Hexadec. \rightarrow Binário



Binário				Dígito Hexadec.
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

Adição Binária

- A adição de números binários funciona da mesma maneira que para números decimais

The diagram shows a binary addition problem. At the top, a yellow box contains the text "carry (vai um)". Below this, two red "1"s are positioned above the first two columns of the addition. Arrows point from these red "1"s to the "carry" box. The addition itself is written as follows:

$$\begin{array}{r} 99 \\ + 59 \\ \hline 158 \end{array}$$

The result of the addition is 158, which is written below a horizontal line. The digits 1, 5, and 8 are aligned under the first, second, and third columns respectively.

Adição Binária

- A adição de números binários funciona da mesma maneira que para números decimais

1 carry (vai um)

1 1 0 0

+ 1 0 1 0

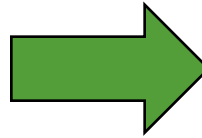
1 0 1 1 0

	Resultado	Carry (Vai um)
0 + 0	0	0
0 + 1	1	0
1 + 0	1	0
1 + 1	0	1

Subtração Binária

- A subtração de números binários funciona da mesma maneira que para números decimais

$$\begin{array}{r} 78 \\ - 59 \\ \hline \end{array}$$



borrow (pediu emprestado)

$$\begin{array}{r} \overset{6}{\cancel{7}} \overset{1}{8} \\ - 59 \\ \hline 19 \end{array}$$

Subtração Binária

- A subtração de números binários funciona da mesma maneira que para números decimais

borrow (pediu emprestado)

$$\begin{array}{r} 11\cancel{0}100 \\ + 10010 \\ \hline 01010 \end{array}$$

The diagram illustrates binary subtraction with borrowing. The top number is 110100, and the bottom number is 10010. A horizontal line is drawn under the bottom number. The result is 01010. A blue '0' is written above the third digit from the right, and a red '1' is written below the fourth digit from the right. An arrow points from the red '1' to the blue '0', indicating the borrowing process.

	Resultado	Borrow (Pedi emprestado)
0 - 0	0	0
0 - 1	1	1
1 - 0	1	0
1 - 1	0	0

Multiplicação Binária

- A multiplicação de números binários funciona da mesma maneira que para números decimais

$$\begin{array}{r} 15 \\ \times 21 \\ \hline 15 \\ 30 \\ \hline 315 \end{array}$$

Multiplicação Binária

- A multiplicação de números binários funciona da mesma maneira que para números decimais

a multiplicação de um número de n bits por um número de m bits requer $n + m$ bits

$$\begin{array}{r} 1101 \\ \times 1001 \\ \hline 1101 \\ 0000 \\ 0000 \\ 1101 \\ \hline 01110101 \end{array}$$

Multiplicação Binária

- A multiplicação de números binários funciona da mesma maneira que para números decimais

a multiplicação de um número de n bits por um número de m bits requer $n + m$ bits

$$\begin{array}{r} 1 1 0 1 \\ x 1 0 0 1 \\ \hline 1 1 0 1 \\ 0 0 0 0 \\ 0 1 1 0 1 \\ 0 0 0 0 \\ 0 0 1 1 0 1 \\ 1 1 0 1 \\ \hline 0 1 1 1 0 1 0 1 \end{array}$$

Divisão Binária

- A divisão em binário é baseada no algoritmo de deslocar e subtrair.

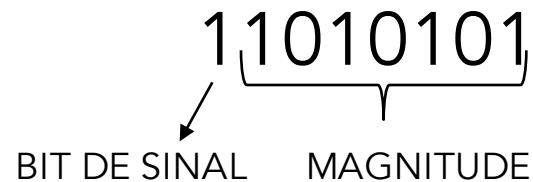
$$\begin{array}{r} 11101 \overline{) 101} \\ \underline{101} \\ 01001 \\ \underline{101} \\ 100 \end{array}$$

Representação de binários negativos

- Existem várias formas de representar números negativos. Algumas delas são:
- Sinal-Magnitude
- Complemento de 1
- Complemento de 2

Sinal-Magnitude

- Definimos um tamanho de palavra
- Usamos o MSB para o sinal:
 - 0 representa o sinal positivo +
 - 1 representa o sinal negativo –



- Exemplos: (tamanho de palavra = 8)
- $01010101_2 = +85_{10}$
- $11010101_2 = -85_{10}$
- $01111111_2 = +127_{10}$
- $11111111_2 = -127_{10}$

Sinal-Magnitude

- Representa uma quantidade igual de números positivos e negativos.
- Com n bits podemos representar de $-(2^{n-1}-1)$ a $(2^{n-1}-1)$
 - Exemplo: 8 bits \rightarrow -127 a +127
- Temos dois valores para 0 (-0 e +0)
 - Exemplo: 8 bits \rightarrow 10000000 e 00000000

Complemento de 1

- Números positivos → igual sinal-magnitude
- Números negativos → complemento do número positivo correspondente
 - Complemento?
 - $0 \rightarrow 1$ e $1 \rightarrow 0$
 - Exemplo:
 - $+25_{10} = 00011001_2$
 - $-25_{10} = 11100110_2$

Complemento de 1

- Representa uma quantidade igual de números positivos e negativos.
- Com n bits podemos representar de $-(2^{n-1}-1)$ a $(2^{n-1}-1)$
 - Exemplo: 8 bits \rightarrow -127 a +127
- Temos dois valores para 0 (-0 e +0)
 - Exemplo: 8 bits \rightarrow 11111111 e 00000000

Complemento de 1

□ Números positivos → igual sinal-magnitude e Complemento de 1

□ Números negativos → (número negativo em complemento de 1) + 1

■ Exemplo:

$$\text{➤ } +25_{10} = 00011001_2$$

$$11100110_2 (\text{complemento de 1})$$

$$\begin{array}{r} + 1 \\ \hline \end{array}$$

$$\text{➤ } +25_{10} = 11100111 (\text{complemento de 2})$$

Complemento de 1

- Com n bits podemos representar de -2^{n-1} a $(2^{n-1}-1)$
 - Exemplo: 8 bits \rightarrow -128 a +127
- Temos apenas um valor para 0
 - Exemplo: 8 bits \rightarrow 00000000
- É o sistema mais usado: adição e a subtração são fáceis

Como se troca de sinal em Compl. de 2?

- 1) Complementar o número ($0 \rightarrow 1$ e $1 \leftarrow 0$)
- 2) Somar 1

Exemplo (palavras de 8 bits):

$$\begin{array}{r} 11011011(-37_{10}) \\ 00100100 \\ + \quad \quad \quad 1 \\ \hline 00100101(+37_{10}) \end{array}$$

Adição em Complemento de 2

- Soma Normalmente
- Descarta o carry

$$\begin{array}{r} 00000111 \\ + 00000100 \\ \hline 00001011 \end{array}$$

$$\begin{array}{r} 7 \\ + 4 \\ \hline 11 \end{array}$$

Subtração em Complemento de 2

- Subtração é igual a adição
 - Exemplo: **fazer $10 - 5$ é o mesmo que $10 + (-5)$**
- Soma Normalmente, independente do sinal
- Descarta o carry

	00001111	15
	+ 11111010	+ -6
	<hr/>	<hr/>
Carry descartado →	1 00001001	9

Exemplo

□ Palavras de 8 bits

□ 31 – 10

■ 31 → 00011111

■ 10 → 00001010

11110101 (complemento)

 +1

■ -10 → 11110110

1 1 1 1 1 1 1

0 0 0 1 1 1 1 1 (31)

+ 1 1 1 1 0 1 1 0 (-10)

0 0 0 1 0 1 0 1

Overflow

O *overflow* pode ser identificado através da inconsistência do sinal do resultado

Palavra de 4 bits

$$\begin{array}{r} 0 \ 1 \ 0 \ 0 \ (+4) \\ + \ 0 \ 1 \ 0 \ 1 \ (+5) \\ \hline 1 \ 0 \ 0 \ 1 \ (-7) \end{array}$$

SINAL DIFERENTE → **OVERFLOW!!**

Overflow

O *overflow* pode ser identificado através da inconsistência do sinal do resultado

Palavra de 4 bits

$$\begin{array}{r} \overset{1}{} \\ 1 \ 1 \ 0 \ 0 \ (-4) \\ + \ 1 \ 0 \ 1 \ 1 \ (-5) \\ \hline 0 \ 1 \ 1 \ 1 \ (+7) \end{array}$$

SINAL DIFERENTE → **OVERFLOW!!**

Overflow

O *overflow* pode ser identificado através da inconsistência do sinal do resultado

Palavra de 4 bits

$$\begin{array}{rcccccl} & & 1 & 1 & 1 & & \\ & 0 & 1 & 0 & 1 & (+5) & \\ + & 0 & 0 & 1 & 1 & (+3) & \\ \hline & 1 & 0 & 0 & 0 & (-8) & \end{array}$$

SINAL DIFERENTE → **OVERFLOW!!**

Ponto Flutuante

- ❑ Representar números inteiros muito grandes → **muitos bits**
- ❑ Sistema de numeração de ponto flutuante
 - representado em **notação científica normalizada**
 - capaz de representar números:
 - ❑ muito grandes e muito pequenos sem o aumento do número de bits
 - ❑ com parte inteira e fracionária

Representação Binária Ponto Flutuante

- Um número de ponto flutuante é representado sempre com o formato de **notação científica normalizada**

$$\begin{array}{c} (\text{sinal})\text{mantissa} \times 2^{\text{expoente}} \\ \downarrow \\ 1 + \text{fração} \end{array}$$

■ Exemplos:

- 1,28726752672 $\times 2^{10}$
- 1,7617168781 $\times 2^{45}$

Representação Binária Ponto Flutuante

- ❑ Padrão IEEE 754, de 1985
 - Precisão simples – 32 bits
 - Precisão dupla – 64 bits
- ❑ A base 2 é implícita, e apenas o sinal, a mantissa e o expoente são armazenados

- ❑ Número de bits

Precisão	Sinal	Expoente	Mantissa
Simple	1	8	23
Dupla	1	11	52

Representação Binária Ponto Flutuante

□ Sinal (S)

- 1 → negativo
- 0 → positivo

□ Mantissa → sempre começará por 1 + fração (F)

□ Expoente (e)

- somado com 127 na precisão simples
- Somado com 1023 na precisão dupla

Precisão simples

□ Fórmula

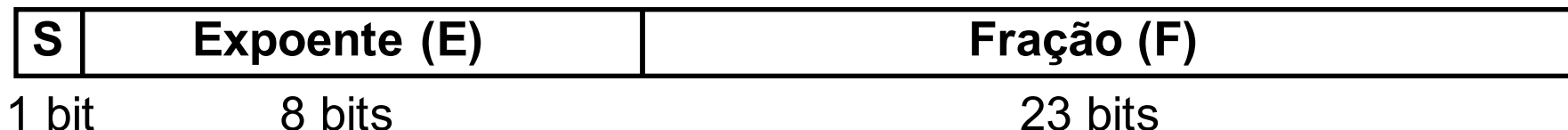
$$Y = (-1)^S (1 + F) * 2^{E-127}$$

■ Onde

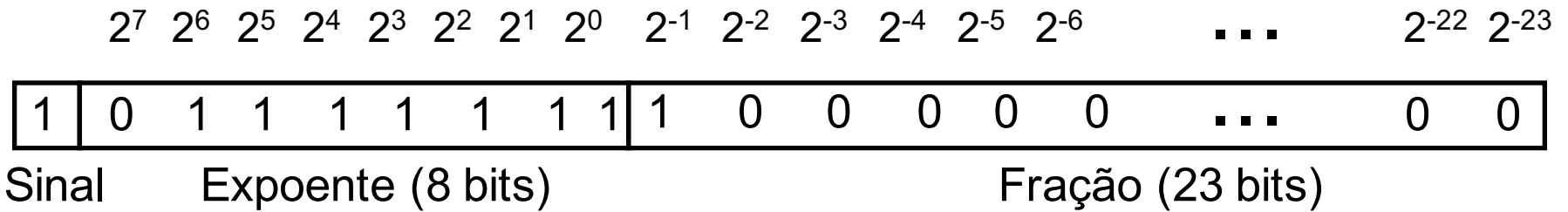
- $S \rightarrow$ sinal
- $F \rightarrow$ fração
- $E = e + 127$, onde $-126 \leq e \leq 127$

□ Bits para a precisão simples

$2^7 \ 2^6 \ 2^5 \ 2^4 \ 2^3 \ 2^2 \ 2^1 \ 2^0 \ 2^{-1} \ 2^{-2} \ 2^{-3} \ 2^{-4} \ 2^{-5} \ 2^{-6} \ \dots \ 2^{-22} \ 2^{-23}$



Exemplo 1

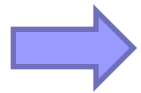


$$S = 1$$

$$E = 127$$

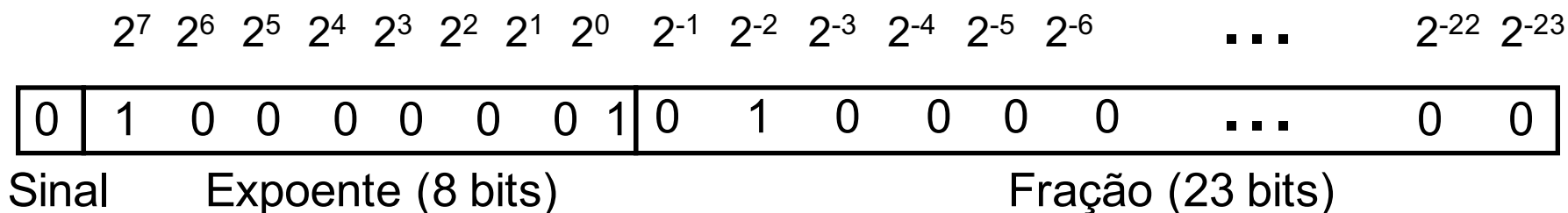
$$F = 0.5$$

$$Y = (-1)^S (1 + F) * 2^{E-127} = (-1)^1 (1 + 0.5) * 2^{127-127}$$



$$Y = -1.5$$

Exemplo 2



$$S = 0 \quad E = 129$$

$$F = 0.25$$

$$Y = (-1)^S (1 + F) * 2^{E-127} = (-1)^0 (1 + 0.25) * 2^{129-127}$$

➡ $Y = 1.25 \times 2^2 = 5$

Exemplo 3

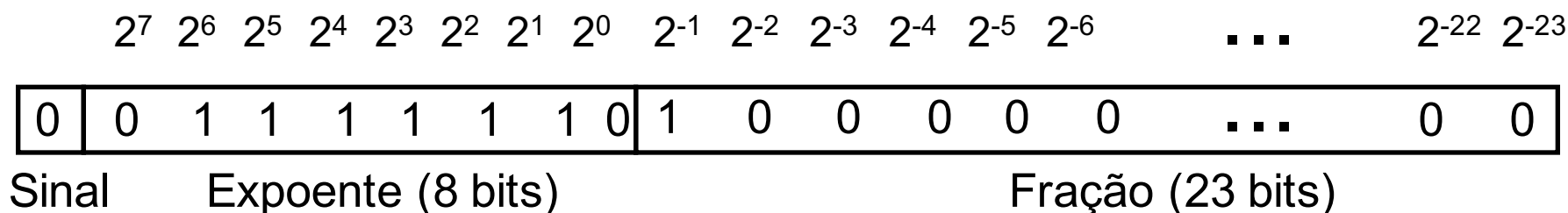
$$Y = 0.75 = \frac{1}{2} + \frac{1}{4} = \frac{3}{4} = 3 \times 2^{-2} = (11)_2 \times 2^{-2} = (1.1)_2 \times 2^{-1}$$

$$Y = (-1)^S (1 + F) * 2^e \quad E = e + 127$$

$$e = -1$$

$$S = 0 \quad E = 126$$

$$F = 0.5$$



Precisão Dupla

□ Fórmula

$$Y = (-1)^S (1 + F) * 2^{E-1023}$$

■ Onde

□ $S \rightarrow$ sinal

□ $F \rightarrow$ fração

□ $E = e + 1023$, onde $-1022 \leq e \leq 1023$

□ Bits para a precisão dupla

$2^{10} \ 2^9 \ 2^8 \ \dots \ 2^2 \ 2^1 \ 2^0 \ 2^{-1} \ 2^{-2} \ 2^{-3} \ 2^{-4} \ 2^{-5} \ 2^{-6} \ \dots \ 2^{-51} \ 2^{-52}$

S	Expoente (E)	Fração (F)
1 bit	11 bits	52 bits