



# Capítulo 1

## Representação da Informação

# 1. Representação de Informação

## ■ 1.1 Sistemas de numeração

- ☐ Notação numérica posicional
- ☐ Sistemas binário, octal e hexadecimal
- ☐ Aritmética em base binária

# 1.1 Sistemas de Numeração

## ■ Notação Numérica posicional

### □ Sistema numérico na base $B$

- Utiliza os algarismos  $\{0, 1, \dots, B-1\}$
- A localização de um algarismo  $a_i$  no número determina seu peso  $B^i$
- O valor do algarismo  $a_i$  é o produto *algarismo  $\times$  peso*

$$a_i \times B^i$$

- O valor  $x$  do número é a soma dos produtos *algarismo  $\times$  peso*


$$x = \sum_{i=0}^n a_i \times B^i = a_n \times B^n + a_{n-1} \times B^{n-1} + \dots + a_0 \times B^0$$

# 1.1 Sistemas de Numeração

## ■ Notação Numérica posicional - exemplo

### □ Sistema decimal (base 10)

- Utiliza os algarismos  $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- A localização de um algarismo no número determina seu peso:

$$367_{10} = 3 \times 10^2 + 6 \times 10^1 + 7 \times 10^0$$



**Para números inteiros: dígito mais à direita  $\Rightarrow$  posição 0**

# 1.1 Sistemas de Numeração

## ■ Notação Numérica posicional - exemplo

### □ Sistema binário (base 2)

- Utiliza os algarismos  $\{0, 1\}$
- A localização de um algarismo no número determina seu peso:

$$101101111_2 = 1 \times 2^8 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + \underbrace{1 \times 2^0}_{\substack{\uparrow \\ \text{posição 0}}} = 367_{10}$$



**Para números inteiros: dígito mais à direita  $\Rightarrow$  posição 0**

# 1.1 Sistemas de Numeração

## ■ Notação Numérica posicional - exemplo

### □ Sistema octal (base 8)

- Utiliza os algarismos  $\{0, 1, 2, 3, 4, 5, 6, 7\}$
- A localização de um algarismo no número determina seu peso:

$$557_8 = 5 \times 8^2 + 5 \times 8^1 + 7 \times 8^0 = 367_{10}$$



**Para números inteiros: dígito mais à direita  $\Rightarrow$  posição 0**

# 1.1 Sistemas de Numeração

## ■ Notação Numérica posicional - exemplo

### □ Sistema hexadecimal (base 16)

- Utiliza os algarismos  $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$
- A localização de um algarismo no número determina seu peso:

$$16F_{16} = 1 \times 16^2 + 6 \times 16^1 + 15 \times 16^0 = 367_{10}$$


**Para números inteiros: dígito mais à direita  $\Rightarrow$  posição 0**

# 1.1 Sistemas de Numeração

- Notação Numérica posicional para números com parte fracionária:

$$x = \sum_{i=-m}^n a_i \times B^i = a_n \times B^n + a_{n-1} \times B^{n-1} + \dots + a_{-m} \times B^{-m}$$

- Exemplo ( $B=10$ ):

$$5,875_{10} = 5 \times 10^0 + 8 \times 10^{-1} + 7 \times 10^{-2} + 5 \times 10^{-3}$$

- Exemplo ( $B=2$ ):

$$101,111_2 = 1 \times 2^2 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} = 5,875_{10}$$



# 1.1 Sistemas de Numeração

- Conversão entre os sistemas binário (base 2), octal (base 8) e hexadecimal (base 16)
  - A conversão entre esses sistemas de numeração é simplificada pela relação entre eles:
    - $8 = 2^3$
    - $16 = 2^4$

# 1.1 Sistemas de Numeração

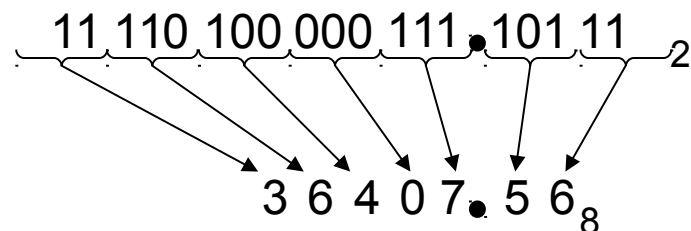
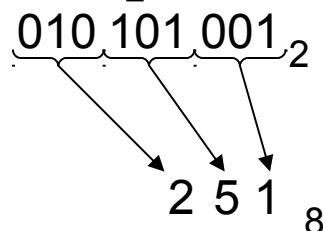
## ■ Sistema octal (base 8)

- Um dígito octal corresponde a um número binário com 3 bits (**binary digits**)

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

# 1.1 Sistemas de Numeração

- Conversão do sistema *binário* para o sistema *octal* (e vice-versa)
  - Começando à direita do número binário (caso o número seja inteiro), ou a partir da vírgula, agrupa-se os dígitos binários em grupos de 3 bits
  - Associa-se a cada agrupamento ao dígito octal correspondente



# 1.1 Sistemas de Numeração

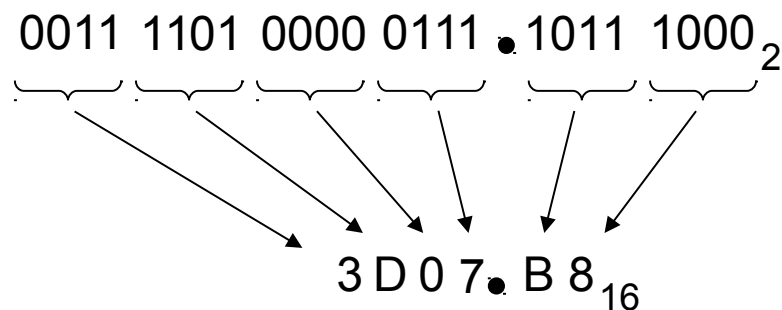
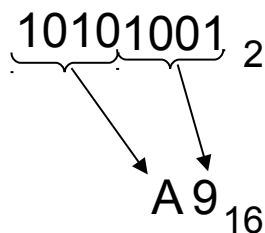
## ■ Sistema hexadecimal (base 16)

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

Binário	Dígito Hexadecimal
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

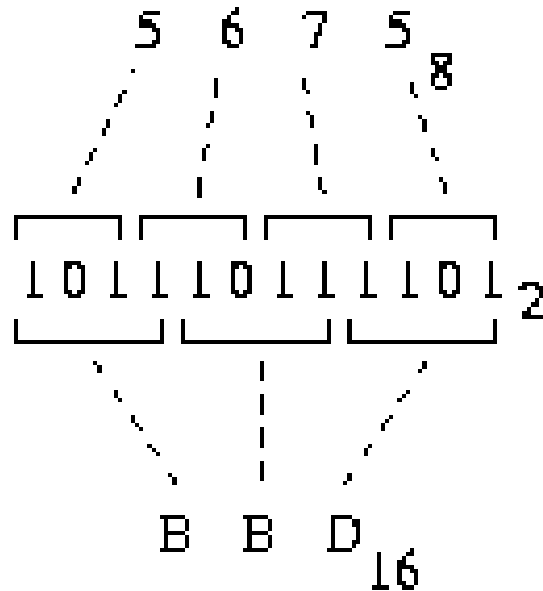
# 1.1 Sistemas de Numeração

- Conversão do sistema *binário* para o sistema *hexadecimal* ( e vice-versa)
  - Começando à direita do número binário (caso o número seja inteiro), ou a partir da vírgula, agrupa-se os dígitos binários em grupos de 4 bits
  - Associa-se a cada agrupamento ao dígito hexadecimal correspondente



# 1.1 Sistemas de Numeração

- Conversão do sistema *octal* para o *hexadecimal* (e vice-versa)
  - Converte-se inicialmente o número para o sistema binário:



# 1.1 Sistemas de Numeração

## ■ Aritmética na base-2

□ Tabelas de *adição e multiplicação*

+	0	1
0	0	1
1	1	10

×	0	1
0	0	0
1	0	1

# 1.1 Sistemas de Numeração

## ■ Adição na base-2

$$11010_2 + 1011_2 = 100101_2$$

Carry →

1			1		
1	1	0	1	0	
		1	0	1	1
					+
<hr/>					
1	0	0	1	0	1



# 1.1 Sistemas de Numeração

## ■ Subtração na base-2

$$1001_2 - 0111_2 = 0010_2$$

Borrow →

	1		1		
	1	0	0	1	
	0	1	1	1	–
	<hr/>				
	0	0	1	0	

# 1.1 Sistemas de Numeração

## ■ Multiplicação na base-2

$$1110_2 \times 1101_2 = 10110110_2$$

Carry → 1 1 1 1

							1	1
							1	1
							0	1
							1	
							1	1
							1	1
							0	
							1	1
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	
							1	0
							1	1
							0	

# 1.1 Sistemas de Numeração

## ■ Divisão na base-2

$$10111010_2 \div 1110_2 = 1101_2 \quad \text{Resto} = 100_2$$

1							
1	0	1	1	1	0	1	0
	1	1	1	0	–		
<hr/>							
	1						
	1	0	0	1	0		
		1	1	1	0	–	
<hr/>							
		1					
		0	1	0	0	1	0
			1	1	1	0	–
<hr/>							
			1	0	0		

100



# 1. Representação da Informação

## 1.1

## 1.2 Representação de N<sup>os</sup> Negativos

### 1.2.1 Representação complemento-a-dois

### 1.2.2 Outras representações

## 1.3 Representação de Números Reais

### 1.3.1 Representação em ponto fixo

### 1.3.2 Representação em ponto flutuante

### 1.3.3 Representação binária em ponto flutuante

## 1.2.1 Representação complemento-a-dois

- Valor associado a um número de  $n$  bits:

$$Valor = \underset{\uparrow}{-x_{n-1}} \cdot 2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i$$

$$x_{n-1} = \begin{cases} 1 & \text{para números } \textit{negativos} \\ 0 & \text{para números } \textit{positivos} \end{cases}$$

- O dígito mais significativo – MSB (Most Significant Bit) – sinaliza o sinal

## 1.2.1 Representação complemento-a-dois

- Valor associado a um número de  $n$  bits:

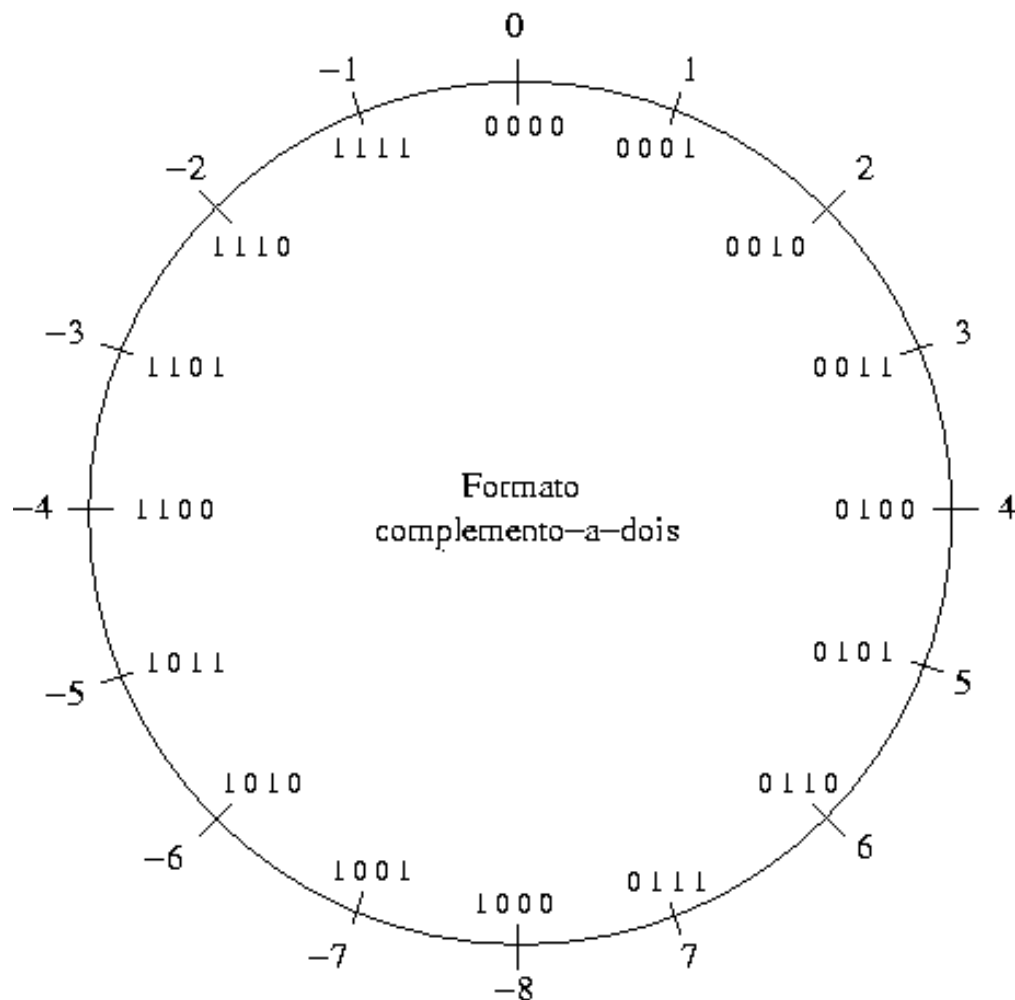
$$Valor = -x_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i$$

$$-2^{n-1} \leq Valor \leq 2^{n-1} - 1$$

## 1.2.1 Representação complemento-a-dois

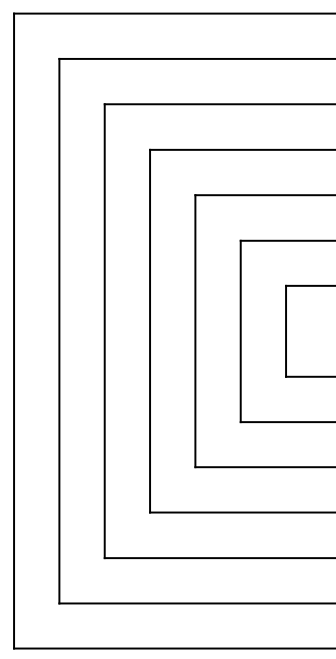
- Largura de palavra  $n = 4$

Binário	Valor
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	-7
1 0 1 0	-6
1 0 1 1	-5
1 1 0 0	-4
1 1 0 1	-3
1 1 1 0	-2
1 1 1 1	-1



## 1.2.1 Representação complemento-a-dois

- O *complemento-a-dois* de um número nessa representação corresponde ao mesmo número com *senal oposto*



Binário	Valor
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	-7
1 0 1 0	-6
1 0 1 1	-5
1 1 0 0	-4
1 1 0 1	-3
1 1 1 0	-2
1 1 1 1	-1



## 1.2.1 Representação complemento-a-dois

### ■ *Complemento-a-dois por complemento e adição binária*

1. *Complementar todos os bits do número X*
2. *Adicionar um ao resultado*

(Valor = +6) → 0 1 1 0

Complemento bit-a-bit ↓

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

(Valor = -6) → 1 0 1 0

(Valor = -3) → 1 1 0 1

Complemento bit-a-bit ↓

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

(Valor = +3) → 0 0 1 1

## 1.2.1 Representação complemento-a-dois

### ■ Operação de *subtração*

- Podem ser realizadas através da *soma* do *minuendo* com o *complemento-a-dois* do *subtraendo*:

$$\begin{array}{r} \text{minuendo} \\ \text{subtraendo} \quad - \\ \hline \end{array} \quad \equiv \quad \begin{array}{r} \text{minuendo} \\ (- \text{ subtraendo}) \quad + \\ \hline \end{array}$$

- Exemplo ( $n=4$ ):  $3 - 5 = 3 + (-5) = -2$

(X = +5)	0	1	0	1
complementar	↓	↓	↓	
(X = -5)	1	0	1	1

		1	1		
(Valor = +3)	0	0	1	1	
(Valor = -5)	1	0	1	1	+
(Valor = -2)	1	1	1	0	

## 1.2.1 Representação complemento-a-dois

### ■ Operação de *subtração*

$$4 - 3 = 4 + (-3) = 1$$

(X = +3)	0	0	1	1
complementar	↓	↓	↓	
(X = -3)	1	1	0	1

(Valor = +4)	1	1	0	0
(Valor = -3)	1	1	0	1
	+			
(Valor = +1)	0	0	0	1

Esse *carry* pode ser descartado nos casos em que não ocorrer *overflow*, ou seja, quando o valor do resultado estiver dentro da faixa de valores representáveis pela largura da palavra.



## 1.2.1 Representação complemento-a-dois

### ■ Operação de *soma/subtração*

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

$$-4 - 5 = (-4) + (-5) = -9$$

(X = +4)      0   1   0   0

complementar ↓

(X = -4)      1   1   0   0

(X = +5)      0   1   0   1

complementar ↓

(X = -5)      1   0   1   1

	1				
(Valor = -4)	1	1	0	0	
(Valor = -5)	1	0	1	1	+
<hr/>					
(Valor = +7!)	0	1	1	1	

## 1.2.1 Representação complemento-a-dois

### ■ Operação de *soma/subtração*

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

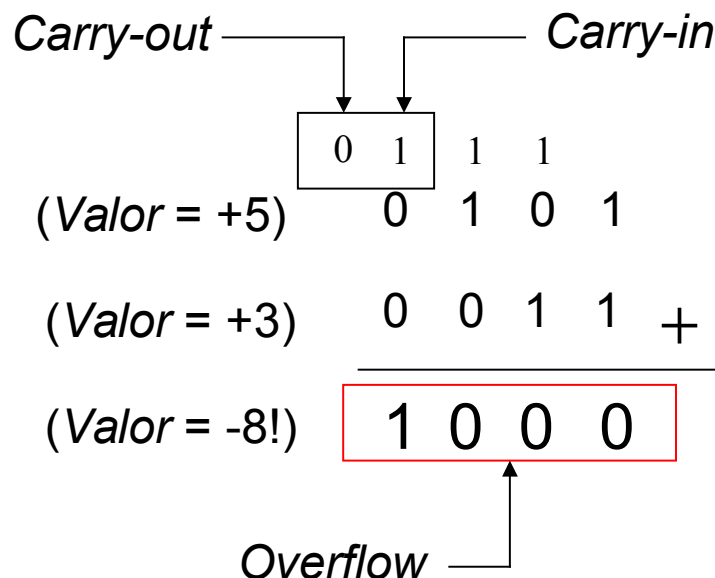
$$5 + 6 = 11$$

	1				
( <i>Valor</i> = +5)	0	1	0	1	
( <i>Valor</i> = +6)	0	1	1	0	+
					<hr/>
( <i>Valor</i> = -5!)	1	0	1	1	

## 1.2.1 Representação complemento-a-dois

### ■ Operação de *soma/subtração*

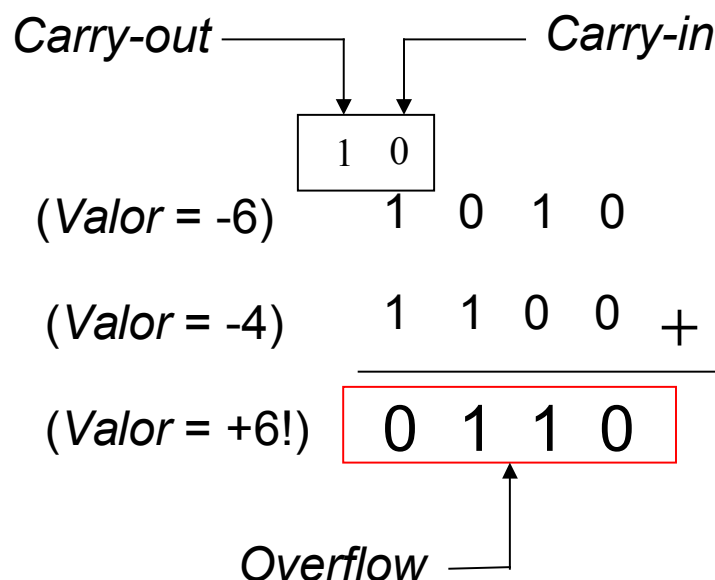
- O *overflow* ocorre quando o *carry-in* para o MSB difere do *carry-out* proveniente do MSB.



## 1.2.1 Representação complemento-a-dois

### ■ Operação de *soma/subtração*

- O *overflow* ocorre quando o *carry-in* para o MSB difere do *carry-out* proveniente do MSB.

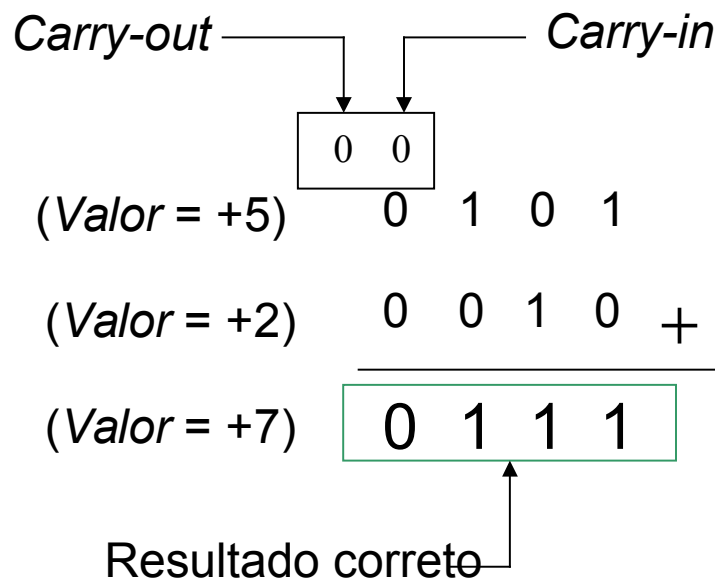




## 1.2.1 Representação complemento-a-dois

### ■ Operação de *soma/subtração*

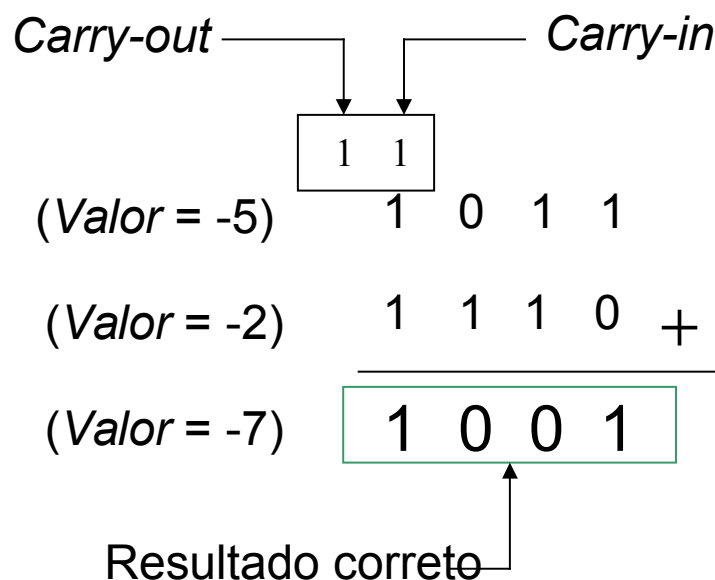
- Não há *overflow* quando o *carry-in* para o MSB é igual ao *carry-out* proveniente do MSB.



## 1.2.1 Representação complemento-a-dois

### ■ Operação de *soma/subtração*

- Não há *overflow* quando o *carry-in* para o MSB é igual ao *carry-out* proveniente do MSB.



## 1.3 Representação de Números Reais

### ■ Representação em ponto fixo

- Reserva-se um número de bits para representar a parte fracionária
- Assume-se uma posição fixa para o *ponto binário* (que não é armazenado explicitamente).

## 1.3.1 Representação em Ponto Fixo

Binário	Valor Decimal
1 0 1 0 0 1 1 1 .	167
1 0 1 0 0 1 1 . 1	83.5
1 0 1 0 0 1 . 1 1	41.75
1 0 1 0 0 . 1 1 1	20.875
1 0 1 0 . 0 1 1 1	10.4375
1 0 1 . 0 0 1 1 1	5.21875
1 0 . 1 0 0 1 1 1	2.609375
1 . 0 1 0 0 1 1 1	1.3046875
. 1 0 1 0 0 1 1 1	0.65234375

## 1.3.1 Representação em Ponto Fixo

- Note que, para números de  $n$  bits, a quantidade de valores representáveis é  $2^n$
- Supondo
  - $i$  bits para a parte inteira
  - $f$  bits para a parte fracionária
  - $n = i + f$
- Separação entre números:  $2^{-f}$
- Maior número positivo:  $(2^{n-1} - 1) / 2^{-f}$
- Número mais negativo:  $-2^{n-1} / 2^{-f}$

# 1.3.1 Representação em Ponto Fixo

- Para palavras de largura  $n=8$

$n$	$i$	$f$	Valores	Menor valor	Maior valor	Intervalo
8	8	0	256	-128	127	1
8	7	1	256	-64	63.5	0.5
8	6	2	256	-32	31.75	0.25
8	5	3	256	-16	15.875	0.125.
8	4	4	256	-8	7.9375	0.0625
8	3	5	256	-4	3.96875	0.03125
8	2	6	256	-2	1.984375	0.015625
8	1	7	256	-1	0.9921875	0.0078125
8	0	8	256	-0.5	0.4960937 5	0.0039062 5

## 1.3.2 Representação em Ponto Flutuante

- Um número representado em *ponto flutuante* apresenta quatro partes
  - Sinal
  - Mantissa
  - Base
  - Expoente
- Esses quatro componentes se combinam para formar o valor  $X$ :

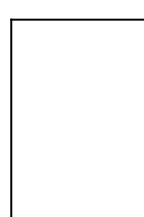
$$X = sinal * mantissa * base^{expoente}$$

## 1.3.2 Representação em Ponto Flutuante

- Números em ponto flutuante podem ser representados de diversas maneiras

□ Exemplo:  $X = -3$

Sinal	Mantissa	Exponente
-1	30	-1
-1	3	0
-1	0.3	1
-1	0.03	2



- Para números *normalizados*, a mantissa está na faixa

$$\frac{1}{base} \leq mantissa \leq 1$$



## 1.3.2 Representação em Ponto Flutuante

- Números *normalizados* apresentam uma mantissa com um dígito não-nulo imediatamente à direita da vírgula, e o dígito nulo imediatamente à esquerda da vírgula.

$$\frac{1}{base} \leq mantissa \leq 1$$

- Note que não há representação normalizada para o valor 0 (zero).

## 1.3.2 Representação Binária em 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

## 1.3.2 Representação Binária em Ponto Flutuante

### ■ Sinal

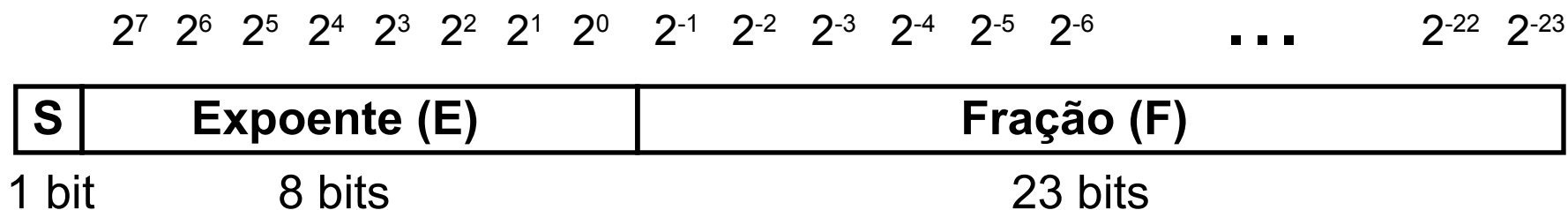
- 1 indica número negativo
- 0 indica número positivo

### ■ Mantissa

- A posição do ponto binário é assumida como sendo aquela adotada na *notação científica normalizada*, com exatamente um dígito diferente de zero antes do ponto binário
- Como esse dígito tem que necessariamente ser “1”, ele não é armazenado (bit implícito), o que aumenta em um bit a precisão do número!

## 1.3.2 Representação Binária em Ponto Flutuante

- Atribuição dos bits para precisão simples:



- Valor decimal correspondente:

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

- Para números normalizados:

$$1 \leq E \leq 254$$

- Ou seja:  $-126 \leq e \leq 127$

## 1.3.2 Representação Binária em Ponto Flutuante

### ■ Expoente


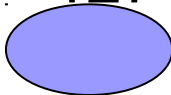
- Note que foi adicionada uma polarização (bias) que deve ser somada ao *expoente real* ( $e$ ) para obter o *expoente armazenado* ( $E$ ):
- Valor da polarização
  - 127 na precisão simples:

$$E = e + 127$$

## 1.3.2 Representação Binária em Ponto Flutuante

### ■ Expoente

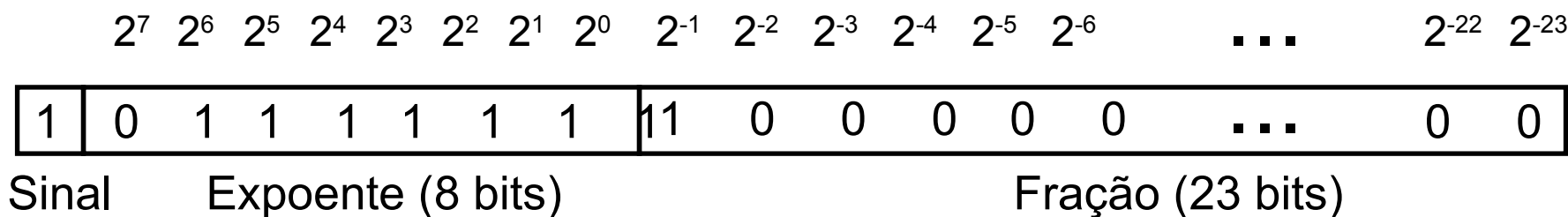
□ Precisão simples:  $E = e + 127$

Expoente armazenado ( <b>E</b> )	Expoente real ( <b>e</b> )
0 0 0 0 0 0 0 0	
0 0 0 0 0 0 0 1	-126
⋮	⋮
0 1 1 1 1 1 1 0	-1
0 1 1 1 1 1 1 1	0
1 0 0 0 0 0 0 0	+1
⋮	⋮
1 1 1 1 1 1 1 0	+127
1 1 1 1 1 1 1 1	

Reservados para valores especiais

## 1.3.2 Representação Binária em Ponto Flutuante

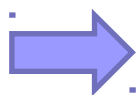
### ■ Exemplo 1



$$S = 1 \quad E = 127$$

$$F = 0.5$$

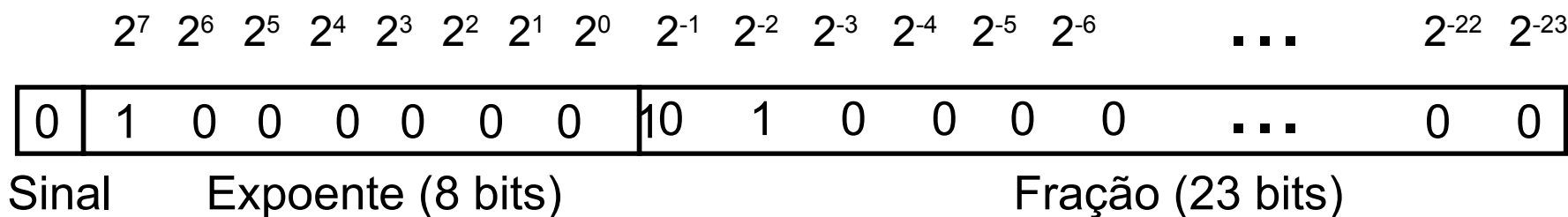
$$Y = (-1)^S (1 + F) * 2^{E-127} = (-1)^1 (1 + 0.5) * 2^{127-127}$$



$$Y = -1.5$$

## 1.3.2 Representação Binária em Ponto Flutuante

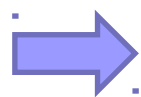
### ■ Exemplo 2



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

$$F = 0.25$$

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


$$Y = 1.25 \times 2^2 = 5$$



## 1.3.2 Representação Binária em Ponto Flutuante

### ■ Exemplo 3

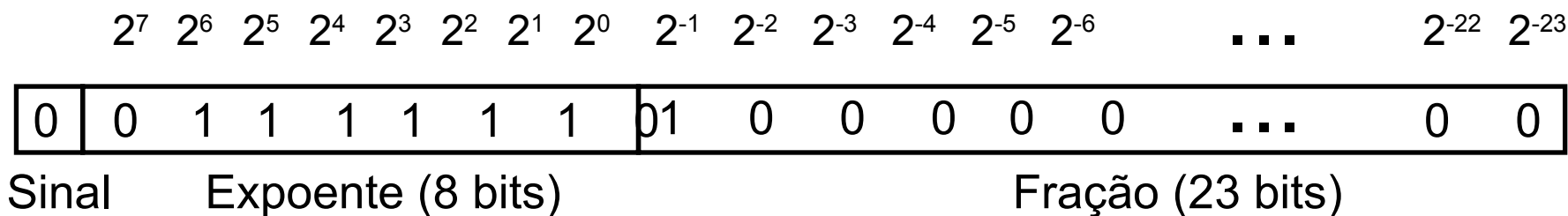
$$Y = 0.75 = \frac{1}{2} + \frac{1}{4} = \frac{3}{4} = 3 \times 2^{-2} = (11)_2 \times 2^{-2} = (1.1)_2 \times 2^{-1}$$

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

$$e = -1$$

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

$$F = 0.5$$



## 1.3.2 Representação Binária em Ponto Flutuante

■ **Exemplo 4**  $Y = -2,625 = -\left(2 + \frac{1}{2} + \frac{1}{8}\right) = -\frac{21}{8}$

$$Y = -21 \times 2^{-3} = -(10101)_2 \times 2^{-3} = -(1.0101)_2 \times 2^1$$

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

$$e = 1$$

$$S = 1 \quad E = 128$$

$$F = 2^{-2} + 2^{-4} = 0.3125$$

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

1	1	0	0	0	0	0	0	0	0	1	0	1	0	0	...	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	-----	---	---

Sinal

Expoente (8 bits)

Fração (23 bits)

## 1.3.2 Representação Binária em Ponto Flutuante

### ■ Exemplo 5

$$Y = \frac{37}{32} = 37 \times 2^{-5}$$

$$Y = (100101)_2 \times 2^{-5} = (1.00101)_2 \times 2^0$$

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

$$e = 0$$

$$S = 0 \quad E = 127$$

$$F = 2^{-3} + 2^{-5} = 0.15625$$

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

1	0	1	1	1	1	1	1	1	0	0	1	0	1	0	...	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	-----	---	---

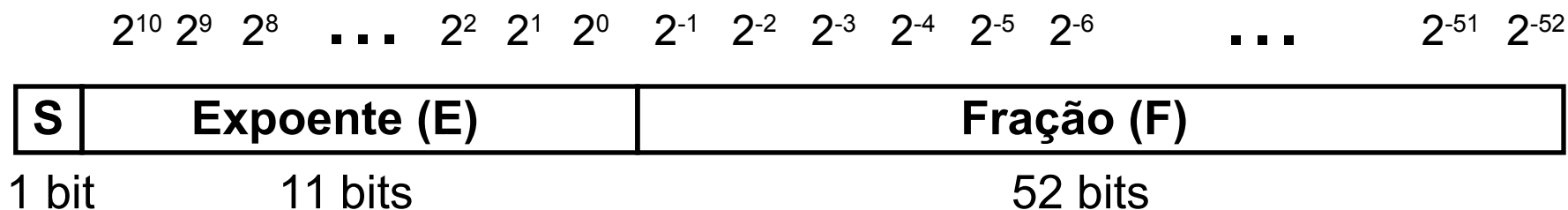
Sinal

Expoente (8 bits)

Fração (23 bits)

## 1.3.2 Representação Binária em Ponto Flutuante

- Atribuição dos bits para precisão dupla:



- Valor decimal correspondente:

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

- Para números normalizados:

$$1 \leq E \leq 2046$$

- Ou seja:  $-1022 \leq e \leq 1023$

## 1.3.2 Representação Binária em Ponto Flutuante

Table D-1 IEEE 754 Format Parameters

Parameter	Format			
	Single	Single-Extended	Double	Double-Extended
$p$	24	32	53	64
$e_{\max}$	+127	1023	+1023	> 16383

Table D-1 IEEE 754 Format Parameters

Parameter	Format			
	Single	Single-Extended	Double	Double-Extended
$e_{\min}$	-126	$\leq -1022$	-1022	$\leq -16382$
Exponent width in bits	8	$\leq 11$	11	15
Format width in bits	32	43	64	79

## 1.3.2 Representação Binária em Ponto Flutuante

Table D-2 IEEE 754 Special Values

Exponent	Fraction	Represents
$e = e_{\min} - 1$	$f = 0$	$\pm 0$
$e = e_{\min} - 1$	$f \neq 0$	$0.f \times 2^{e_{\min}}$
$e_{\min} \leq e \leq e_{\max}$	—	$1.f \times 2^e$
$e = e_{\max} + 1$	$f = 0$	$\infty$
$e = e_{\max} + 1$	$f \neq 0$	NaN

Table D-3 Operations That Produce a NaN

Operation	NaN Produced By
+	$\infty + (-\infty)$
x	$0 \times \infty$
/	$0/0, \infty/\infty$
REM	$x \text{ REM } 0, \infty \text{ REM } y$
$\sqrt{\phantom{x}}$	$\sqrt{x}$ (when $x < 0$ )

# Referências

- Goldberg, David (March 1991), *What every computer scientist should know about floating-point arithmetic*, Computing Surveys. (<http://www.validlab.com/goldberg/paper.ps>)
- Pedroni, Volnei A. (2010), *Eletrônica Digital Moderna e VHDL*, Elsevier