CÁLCULO NUMÉRICO
IME-UERJ

01 - Representação de números reais em binário

Rodrigo Madureira

rodrigo.madureira@ime.uerj.br

Github: https://github.com/rodrigolrmadureira/CalculoNumericoUERJ/
AVA-UERJ: https://ava.pr1.uerj.br/course/view.php?id=6409

Sumário

Sistema Decimal - Números inteiros

Números na base 10:

Cada posição digital de um número inteiro N representa uma potência de dez.

$$
\begin{aligned}
N &= (a_n a_{n-1} \ldots a_1 a_0)_{10} \\
&= a_n \times 10^n + a_{n-1} \times 10^{n-1} + \cdots + a_1 \times 10^1 + a_0 \times 10^0,
\end{aligned}
$$

onde $a_i \in \{0, 1, 2, \ldots, 9\}$.

Exemplos:

$(21)_{10} = 2 \times 10^1 + 1 \times 10^0.$

$(2001)_{10} = 2 \times 10^3 + 0 \times 10^2 + 0 \times 10^1 + 1 \times 10^0.$

$(23457)_{10} = 2 \times 10^4 + 3 \times 10^3 + 4 \times 10^2 + 5 \times 10^1 + 7 \times 10^0.$

Sistema Decimal - Números inteiros

Números na base 10:

Cada posição digital de um número inteiro N representa uma potência de dez.

$$
\begin{aligned}
N &= (a_n a_{n-1} \dots a_1 a_0)_{10} \\
&= a_n \times 10^n + a_{n-1} \times 10^{n-1} + \cdots + a_1 \times 10^1 + a_0 \times 10^0,
\end{aligned}
$$

onde $a_i \in \{0, 1, 2, \dots, 9\}$.

Exemplos:

$(21)_{10} = 2 \times 10^1 + 1 \times 10^0.$

$(2001)_{10} = 2 \times 10^3 + 0 \times 10^2 + 0 \times 10^1 + 1 \times 10^0.$

$(23457)_{10} = 2 \times 10^4 + 3 \times 10^3 + 4 \times 10^2 + 5 \times 10^1 + 7 \times 10^0.$

Sistema Decimal - Números inteiros

Números na base 10:

Cada posição digital de um número inteiro N representa uma potência de dez.

$$
\begin{aligned}
N &= (a_n a_{n-1} \ldots a_1 a_0)_{10} \\
&= a_n \times 10^n + a_{n-1} \times 10^{n-1} + \cdots + a_1 \times 10^1 + a_0 \times 10^0,
\end{aligned}
$$

onde $a_i \in \{0, 1, 2, \ldots, 9\}$.

Exemplos:

$(21)_{10} = 2 \times 10^1 + 1 \times 10^0$.

$(2001)_{10} = 2 \times 10^3 + 0 \times 10^2 + 0 \times 10^1 + 1 \times 10^0$.

$(23457)_{10} = 2 \times 10^4 + 3 \times 10^3 + 4 \times 10^2 + 5 \times 10^1 + 7 \times 10^0$.

Sistema Decimal - Números inteiros

Números na base 10:

Cada posição digital de um número inteiro N representa uma potência de dez.

$$N = (a_n a_{n-1} \ldots a_1 a_0)_{10}$$
$$= a_n \times 10^n + a_{n-1} \times 10^{n-1} + \cdots + a_1 \times 10^1 + a_0 \times 10^0,$$

onde $a_i \in \{0, 1, 2, \ldots, 9\}$.

Exemplos:

$(21)_{10} = 2 \times 10^1 + 1 \times 10^0$.

$(2001)_{10} = 2 \times 10^3 + 0 \times 10^2 + 0 \times 10^1 + 1 \times 10^0$.

$(23457)_{10} = 2 \times 10^4 + 3 \times 10^3 + 4 \times 10^2 + 5 \times 10^1 + 7 \times 10^0$.

Sistema Binário - Números inteiros

Base 2: usada nos computadores binários.

Cada posição digital de um número inteiro N representa uma potência de dois.

$$N = (a_n a_{n-1} \dots a_1 a_0)_2$$
$$= a_n \times 2^n + a_{n-1} \times 2^{n-1} + \cdots + a_1 \times 2^1 + a_0 \times 2^0,$$

onde $a_i \in \{0, 1\}$.

Exemplos:

$(101)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0.$

$(1010)_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0.$

$(110101)_2 = 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0.$

Sistema Binário - Números inteiros

Base 2: usada nos computadores binários.

Cada posição digital de um número inteiro N representa uma potência de dois.

$$\begin{aligned} N &= (a_n a_{n-1} \dots a_1 a_0)_2 \\ &= a_n \times 2^n + a_{n-1} \times 2^{n-1} + \cdots + a_1 \times 2^1 + a_0 \times 2^0, \end{aligned}$$

onde $a_i \in \{0, 1\}$.

Exemplos:

$(101)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$.

$(1010)_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$.

$(110101)_2 = 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$.

Sistema Binário - Números inteiros

Base 2: usada nos computadores binários.

Cada posição digital de um número inteiro N representa uma potência de dois.

$$
\begin{aligned}
N &= (a_n a_{n-1} \dots a_1 a_0)_2 \\
&= a_n \times 2^n + a_{n-1} \times 2^{n-1} + \cdots + a_1 \times 2^1 + a_0 \times 2^0,
\end{aligned}
$$

onde $a_i \in \{0, 1\}$.

Exemplos:

$(101)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$.

$(1010)_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$.

$(110101)_2 = 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$.

Sistema Binário - Números inteiros

Base 2: usada nos computadores binários.

Cada posição digital de um número inteiro N representa uma potência de dois.

$$\begin{aligned} N &= (a_n a_{n-1} \dots a_1 a_0)_2 \\ &= a_n \times 2^n + a_{n-1} \times 2^{n-1} + \cdots + a_1 \times 2^1 + a_0 \times 2^0, \end{aligned}$$

onde $a_i \in \{0, 1\}$.

Exemplos:

$(101)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$.

$(1010)_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$.

$(110101)_2 = 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$.

Conversão de bases

Um número na base β pode ser convertido para base decimal como

$$\begin{aligned} N &= (a_n a_{n-1} \dots a_1 a_0)_\beta \\ &= a_n \times \beta^n + a_{n-1} \times \beta^{n-1} + \dots + a_1 \times \beta^1 + a_0 \times \beta^0, \end{aligned}$$

onde a_i são os dígitos do número representado na base β.

Exemplo 1: Converta $(110)_2$ para a base decimal.

$(110)_2 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = (6)_{10}$.

Exemplo 2: Converta $(1001)_2$ para a base decimal.

$(1001)_2 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (9)_{10}$.

Conversão de bases

Um número na base β pode ser convertido para base decimal como

$$N = (a_n a_{n-1} \dots a_1 a_0)_\beta$$
$$= a_n \times \beta^n + a_{n-1} \times \beta^{n-1} + \dots + a_1 \times \beta^1 + a_0 \times \beta^0,$$

onde a_i são os dígitos do número representado na base β.

Exemplo 1: Converta $(110)_2$ para a base decimal.

$(110)_2 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = (6)_{10}$.

Exemplo 2: Converta $(1001)_2$ para a base decimal.

$(1001)_2 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (9)_{10}$.

Conversão de bases

Um número na base β pode ser convertido para base decimal como

$$
\begin{aligned}
N &= (a_n a_{n-1} \dots a_1 a_0)_\beta \\
&= a_n \times \beta^n + a_{n-1} \times \beta^{n-1} + \cdots + a_1 \times \beta^1 + a_0 \times \beta^0,
\end{aligned}
$$

onde a_i são os dígitos do número representado na base β.

Exemplo 1: Converta $(110)_2$ para a base decimal.

$(110)_2 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = (6)_{10}$.

Exemplo 2: Converta $(1001)_2$ para a base decimal.

$(1001)_2 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (9)_{10}$.

Conversão de bases

Um número na base β pode ser convertido para base decimal como

$$\begin{aligned} N &= (a_n a_{n-1} \dots a_1 a_0)_\beta \\ &= a_n \times \beta^n + a_{n-1} \times \beta^{n-1} + \dots + a_1 \times \beta^1 + a_0 \times \beta^0, \end{aligned}$$

onde a_i são os dígitos do número representado na base β.

Exemplo 1: Converta $(110)_2$ para a base decimal.

$(110)_2 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = (6)_{10}$.

Exemplo 2: Converta $(1001)_2$ para a base decimal.

$(1001)_2 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (9)_{10}$.

Conversão de bases

Um número na base β pode ser convertido para base decimal como

$$\begin{aligned} N &= (a_n a_{n-1} \dots a_1 a_0)_\beta \\ &= a_n \times \beta^n + a_{n-1} \times \beta^{n-1} + \dots + a_1 \times \beta^1 + a_0 \times \beta^0, \end{aligned}$$

onde a_i são os dígitos do número representado na base β.

Exemplo 1: Converta $(110)_2$ para a base decimal.

$(110)_2 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = (6)_{10}$.

Exemplo 2: Converta $(1001)_2$ para a base decimal.

$(1001)_2 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (9)_{10}$.

Conversão de bases

Conversão da base decimal para a base β:

Divisões sucessivas do número em base decimal por β até que o quociente seja igual a zero.

O número na base β é formado pela concatenação em ordem inversa dos restos das divisões.

Exemplo 1: Converta $(25)_{10}$ para a base 2.

25	2			
24	12	2		
1	12	6	2	
	0	6	3	2
		0	2	1
			1	

$\longleftarrow$

$$25 = 11001_2$$

Conversão de bases

Conversão da base decimal para a base β:

Divisões sucessivas do número em base decimal por β até que o quociente seja igual a zero.

O número na base β é formado pela concatenação em ordem inversa dos restos das divisões.

Exemplo 1: Converta $(25)_{10}$ para a base 2.

$$\begin{array}{ccccc}
25 & \multicolumn{1}{|c}{2} & & & \\ \cline{2-2}
24 & 12 & \multicolumn{1}{|c}{2} & & \\ \cline{1-1}\cline{3-3}
1 & 12 & 6 & \multicolumn{1}{|c}{2} & \\ \cline{2-2}\cline{4-4}
 & 0 & 6 & 3 & \multicolumn{1}{|c}{2} \\ \cline{3-3}\cline{5-5}
 & & 0 & 2 & 1 \\ \cline{4-4}
 & & & 1 &
\end{array}$$

$$\longleftarrow$$

$$25 = 11001_2$$

Conversão de bases

Conversão da base decimal para a base β:

Divisões sucessivas do número em base decimal por β até que o quociente seja igual a zero.

O número na base β é formado pela concatenação em ordem inversa dos restos das divisões.

Exemplo 1: Converta $(25)_{10}$ para a base 2.

25	2			
24	12	2		
1	12	6	2	
	0	6	3	2
		0	2	1
			1	

$\longleftarrow$

$25 = 11001_2$

Conversão de bases

Exemplo 2: Converta $(637)_{10}$ para a base 2.

637	2								
636	318	2							
1	318	159	2						
	0	158	79	2					
		1	78	39	2				
			1	38	19	2			
				1	18	9	2		
					1	8	4	2	
						1	4	2	2
							0	2	1
								0	

$\longleftarrow$

$637 = 1001111101_2$

Conversão de bases

Exemplo 2: Converta $(637)_{10}$ para a base 2.

637	2								
636	318	2							
1	318	159	2						
	0	158	79	2					
		1	78	39	2				
			1	38	19	2			
				1	18	9	2		
					1	8	4	2	
						1	4	2	2
							0	2	1
								0	

$\longleftarrow$

$$637 = 1001111101_2$$

Representação de Números Reais no Computador

Um número real positivo N na base β pode ser escrito como

$$
\begin{aligned}
N &= (\underbrace{a_n a_{n-1} \ldots a_1 a_0}_{N_{\text{int}}}\,,\,\underbrace{b_1 b_2 b_3 \ldots}_{N_{\text{frac}}})_\beta \\
&= a_n \times \beta^n + a_{n-1} \times \beta^{n-1} + \cdots + a_1 \times \beta^1 + a_0 \times \beta^0 \\
&\quad + b_1 \times \beta^{-1} + b_2 \times \beta^{-2} + b_3 \times \beta^{-3} \ldots \\
&= \underbrace{\sum_{i=0}^{n} a_i \times \beta^i}_{N_{\text{int}}} + \underbrace{\sum_{i=1}^{\infty} b_i \times \beta^{-i}}_{N_{\text{frac}}},
\end{aligned}
$$

onde N_{int} é a parte inteira de N e N_{frac} é a parte fracionária de N.

Exemplo 1:
$(123,45)_{10} = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$.

Exemplo 2:
$(101,101)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = (5,625)_{10}$

Representação de Números Reais no Computador

Um número real positivo N na base β pode ser escrito como

$$
\begin{aligned}
N &= (\underbrace{a_n a_{n-1} \dots a_1 a_0}_{N_{\text{int}}} \, , \, \underbrace{b_1 b_2 b_3 \dots}_{N_{\text{frac}}})_\beta \\
&= a_n \times \beta^n + a_{n-1} \times \beta^{n-1} + \cdots + a_1 \times \beta^1 + a_0 \times \beta^0 \\
&\quad + b_1 \times \beta^{-1} + b_2 \times \beta^{-2} + b_3 \times \beta^{-3} \dots \\
&= \underbrace{\sum_{i=0}^{n} a_i \times \beta^i}_{N_{\text{int}}} + \underbrace{\sum_{i=1}^{\infty} b_i \times \beta^{-i}}_{N_{\text{frac}}},
\end{aligned}
$$

onde N_{int} é a parte inteira de N e N_{frac} é a parte fracionária de N.

Exemplo 1:
$(123,45)_{10} = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$.

Exemplo 2:
$(101,101)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = (5,625)_{10}$.

Representação de Números Reais no Computador

Um número real positivo N na base β pode ser escrito como

$$
\begin{aligned}
N &= (\underbrace{a_n a_{n-1} \dots a_1 a_0}_{N_{\text{int}}}\,,\,\underbrace{b_1 b_2 b_3 \dots}_{N_{\text{frac}}})_\beta \\
&= a_n \times \beta^n + a_{n-1} \times \beta^{n-1} + \dots + a_1 \times \beta^1 + a_0 \times \beta^0 \\
&\quad + b_1 \times \beta^{-1} + b_2 \times \beta^{-2} + b_3 \times \beta^{-3} \dots \\
&= \underbrace{\sum_{i=0}^{n} a_i \times \beta^i}_{N_{\text{int}}} + \underbrace{\sum_{i=1}^{\infty} b_i \times \beta^{-i}}_{N_{\text{frac}}},
\end{aligned}
$$

onde N_{int} é a parte inteira de N e N_{frac} é a parte fracionária de N.

Exemplo 1:
$(123,45)_{10} = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$.

Exemplo 2:
$(101,101)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = (5,625)_{10}$.

Representação de Números Reais no Computador

- Conversão de base binária para decimal
  - Similar ao caso inteiro

- Conversão de base decimal para binária
  - Converte-se a parte inteira
    - Divisões sucessivas e os restos na ordem reversa
  - Converte-se a parte fracionária
    - Multiplicações sucessivas e as partes inteiras na ordem direta

Exemplo: Converta $(12,675)_{10}$ para a base binária

Parte inteira: $(12)_{10} \Rightarrow$ Divisões sucessivas

Parte fracionária: $(0,675)_{10} \Rightarrow$ Multiplicações sucessivas

Representação de Números Reais no Computador

- Conversão de base binária para decimal
  - Similar ao caso inteiro
- Conversão de base decimal para binária
  - Converte-se a parte inteira
    - Divisões sucessivas e os restos na ordem reversa
  - Converte-se a parte fracionária
    - Multiplicações sucessivas e as partes inteiras na ordem direta

Exemplo: Converta $(12,675)_{10}$ para a base binária

Parte inteira: $(12)_{10} \Rightarrow$ Divisões sucessivas

Parte fracionária: $(0,675)_{10} \Rightarrow$ Multiplicações sucessivas

Representação de Números Reais no Computador

- Conversão de base binária para decimal
  - Similar ao caso inteiro

- Conversão de base decimal para binária
  - Converte-se a parte inteira
    - Divisões sucessivas e os restos na ordem reversa
  - Converte-se a parte fracionária
    - Multiplicações sucessivas e as partes inteiras na ordem direta

Exemplo: Converta $(12,675)_{10}$ para a base binária

Parte inteira: $(12)_{10} \Rightarrow$ Divisões sucessivas

Parte fracionária: $(0,675)_{10} \Rightarrow$ Multiplicações sucessivas

Representação de Números Reais no Computador

- Conversão de base binária para decimal
  - Similar ao caso inteiro

- Conversão de base decimal para binária
  - Converte-se a parte inteira
    - Divisões sucessivas e os restos na ordem reversa
  - Converte-se a parte fracionária
    - Multiplicações sucessivas e as partes inteiras na ordem direta

Exemplo: Converta $(12,675)_{10}$ para a base binária

Parte inteira: $(12)_{10} \Rightarrow$ Divisões sucessivas

Parte fracionária: $(0,675)_{10} \Rightarrow$ Multiplicações sucessivas

Representação de Números Reais no Computador

Parte inteira: $(12)_{10} \Rightarrow$ Divisões sucessivas

$$\begin{array}{cccc} 12 & 2 & & \\ 12 & 6 & 2 & \\ 0 & 6 & 3 & 2 \\ & 0 & 2 & 1 \\ & & 1 & \end{array}$$

$$\longleftarrow$$

$$12 = 1100_2$$

Parte fracionária: $(0,675)_{10} \Rightarrow$ Multiplicações sucessivas

Representação de Números Reais no Computador

Parte inteira: $(12)_{10} \Rightarrow$ Divisões sucessivas

$$
\begin{array}{c|cccc}
12 & 2 & & & \\ \cline{2-2}
12 & 6 & \multicolumn{1}{|c}{2} & & \\ \cline{1-1} \cline{3-3}
\color{red}{0} & 6 & 3 & \multicolumn{1}{|c}{2} & \\ \cline{2-2} \cline{4-4}
 & \color{red}{0} & 2 & \color{red}{1} & \\ \cline{3-3}
 & & \color{red}{1} & &
\end{array}
$$

$$\longleftarrow$$

$$12 = 1100_2$$

Parte fracionária: $(0,675)_{10} \Rightarrow$ Multiplicações sucessivas

Representação de Números Reais no Computador

Parte fracionária: $(0,675)_{10} \Rightarrow$ Multiplicações sucessivas

$0,675 \times 2 = 1,35$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,35 e multiplico por 2)

$0,35 \times 2 = 0,70$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,70 por 2)

$0,70 \times 2 = 1,40$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,40 e multiplico por 2)

$0,40 \times 2 = 0,80$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,80 por 2)

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$0,40 \times 2 = 0,80$

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$\ldots \qquad \Rightarrow (0,675)_{10} = (0,101011001100110\ldots)_2 = (0,101\overline{0110})_2$

Logo, $(12,675)_{10} = (1100,101011001100110\ldots)_2 = (1100,101\overline{0110})_2$

Representação de Números Reais no Computador

Parte fracionária: $(0,675)_{10} \Rightarrow$ Multiplicações sucessivas

$0,675 \times 2 = 1,35$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,35 e multiplico por 2)

$0,35 \times 2 = 0,70$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,70 por 2)

$0,70 \times 2 = 1,40$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,40 e multiplico por 2)

$0,40 \times 2 = 0,80$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,80 por 2)

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$0,40 \times 2 = 0,80$

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$\ldots \qquad \Rightarrow (0,675)_{10} = (0,101011001100110\ldots)_2 = (0,101\overline{0110})_2$

Logo, $(12,675)_{10} = (1100,101011001100110\ldots)_2 = (1100,101\overline{0110})_2$

Representação de Números Reais no Computador

Parte fracionária: $(0,675)_{10} \Rightarrow$ Multiplicações sucessivas

$0,675 \times 2 = 1,35$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,35 e multiplico por 2)

$0,35 \times 2 = 0,70$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,70 por 2)

$0,70 \times 2 = 1,40$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,40 e multiplico por 2)

$0,40 \times 2 = 0,80$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,80 por 2)

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$0,40 \times 2 = 0,80$

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$\ldots \quad \Rightarrow (0,675)_{10} = (0,10101100110011\ldots)_2 = (0,101\overline{0110})_2$

Logo, $(12,675)_{10} = (1100,10101100110011\ldots)_2 = (1100,101\overline{0110})_2$

Representação de Números Reais no Computador

Parte fracionária: $(0,675)_{10} \Rightarrow$ Multiplicações sucessivas

$0,675 \times 2 = 1,35$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,35 e multiplico por 2)

$0,35 \times 2 = 0,70$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,70 por 2)

$0,70 \times 2 = 1,40$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,40 e multiplico por 2)

$0,40 \times 2 = 0,80$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,80 por 2)

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$0,40 \times 2 = 0,80$

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$\ldots \quad \Rightarrow (0,675)_{10} = (0,1010110011001 10\ldots)_2 = (0,101\overline{0110})_2$

Logo, $(12,675)_{10} = (1100,1010110011001 10\ldots)_2 = (1100,101\overline{0110})_2$

Representação de Números Reais no Computador

Parte fracionária: $(0,675)_{10} \Rightarrow$ Multiplicações sucessivas

$0,675 \times 2 = 1,35$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,35 e multiplico por 2)

$0,35 \times 2 = 0,70$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,70 por 2)

$0,70 \times 2 = 1,40$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,40 e multiplico por 2)

$0,40 \times 2 = 0,80$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,80 por 2)

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$0,40 \times 2 = 0,80$

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$\ldots \quad \Rightarrow (0,675)_{10} = (0,10101100110110\ldots)_2 = (0,101\overline{0110})_2$

Logo, $(12,675)_{10} = (1100,10101100110110\ldots)_2 = (1100,101\overline{0110})_2$

Representação de Números Reais no Computador

Parte fracionária: $(0,675)_{10} \Rightarrow$ Multiplicações sucessivas

$0,675 \times 2 = 1,35$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,35 e multiplico por 2)

$0,35 \times 2 = 0,70$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,70 por 2)

$0,70 \times 2 = 1,40$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,40 e multiplico por 2)

$0,40 \times 2 = 0,80$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,80 por 2)

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$0,40 \times 2 = 0,80$

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$\ldots \quad \Rightarrow (0,675)_{10} = (0,10101100110011 0\ldots)_2 = (0,101\overline{0110})_2$

Logo, $(12,675)_{10} = (1100,101011001100110\ldots)_2 = (1100,101\overline{0110})_2$

Representação de Números Reais no Computador

Parte fracionária: $(0,675)_{10} \Rightarrow$ Multiplicações sucessivas

$0,675 \times 2 = 1,35$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,35 e multiplico por 2)

$0,35 \times 2 = 0,70$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,70 por 2)

$0,70 \times 2 = 1,40$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,40 e multiplico por 2)

$0,40 \times 2 = 0,80$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,80 por 2)

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$0,40 \times 2 = 0,80$

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$\ldots \quad \Rightarrow (0,675)_{10} = (0,10101100110011 0\ldots)_2 = (0,101\overline{0110})_2$

Logo, $(12,675)_{10} = (1100,10101100110011 0\ldots)_2 = (1100,101\overline{0110})_2$

Representação de Números Reais no Computador

Parte fracionária: $(0,675)_{10} \Rightarrow$ Multiplicações sucessivas

$0,675 \times 2 = 1,35$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,35 e multiplico por 2)

$0,35 \times 2 = 0,70$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,70 por 2)

$0,70 \times 2 = 1,40$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,40 e multiplico por 2)

$0,40 \times 2 = 0,80$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,80 por 2)

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$0,40 \times 2 = 0,80$

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$\ldots \quad \Rightarrow (0,675)_{10} = (0,10101100110011 0\ldots)_2 = (0,101\overline{0110})_2$

Logo, $(12,675)_{10} = (1100,10101100110011 0\ldots)_2 = (1100,101\overline{0110})_2$

Representação de Números Reais no Computador

Parte fracionária: $(0,675)_{10} \Rightarrow$ Multiplicações sucessivas

$0,675 \times 2 = 1,35$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,35 e multiplico por 2)

$0,35 \times 2 = 0,70$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,70 por 2)

$0,70 \times 2 = 1,40$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,40 e multiplico por 2)

$0,40 \times 2 = 0,80$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,80 por 2)

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$0,40 \times 2 = 0,80$

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$\ldots \quad \Rightarrow (0,675)_{10} = (0,10101100110011 0\ldots)_2 = (0,101\overline{0110})_2$

Logo, $(12,675)_{10} = (1100,10101100110011 0\ldots)_2 = (1100,101\overline{0110})_2$

Representação de Números Reais no Computador

Parte fracionária: $(0,675)_{10} \Rightarrow$ Multiplicações sucessivas

$0,675 \times 2 = 1,35$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,35 e multiplico por 2)

$0,35 \times 2 = 0,70$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,70 por 2)

$0,70 \times 2 = 1,40$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,40 e multiplico por 2)

$0,40 \times 2 = 0,80$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,80 por 2)

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$0,40 \times 2 = 0,80$

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$\ldots \quad \Rightarrow (0,675)_{10} = (0,10101100110 0110\ldots)_2 = (0,101\overline{0110})_2$

Logo, $(12,675)_{10} = (1100,10101100110 0110\ldots)_2 = (1100,101\overline{0110})_2$

Representação de Números Reais no Computador

Parte fracionária: $(0,675)_{10} \Rightarrow$ Multiplicações sucessivas

$0,675 \times 2 = 1,35$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,35 e multiplico por 2)

$0,35 \times 2 = 0,70$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,70 por 2)

$0,70 \times 2 = 1,40$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,40 e multiplico por 2)

$0,40 \times 2 = 0,80$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,80 por 2)

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$0,40 \times 2 = 0,80$

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$\ldots \qquad \Rightarrow (0,675)_{10} = (0,10101100110011 0\ldots)_2 = (0,101\overline{0110})_2$

Logo, $(12,675)_{10} = (1100,101011001100110\ldots)_2 = (1100,101\overline{0110})_2$

Representação de Números Reais no Computador

Parte fracionária: $(0,675)_{10} \Rightarrow$ Multiplicações sucessivas

$0,675 \times 2 = 1,35$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,35 e multiplico por 2)

$0,35 \times 2 = 0,70$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,70 por 2)

$0,70 \times 2 = 1,40$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,40 e multiplico por 2)

$0,40 \times 2 = 0,80$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,80 por 2)

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$0,40 \times 2 = 0,80$

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$\ldots \qquad \Rightarrow (0,675)_{10} = (0,101011001100110\ldots)_2 = (0,101\overline{0110})_2$

Logo, $(12,675)_{10} = (1100,101011001100110\ldots)_2 = (1100,101\overline{0110})_2$

Representação de Números Reais no Computador

Parte fracionária: $(0,675)_{10} \Rightarrow$ Multiplicações sucessivas

$0,675 \times 2 = 1,35$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,35 e multiplico por 2)

$0,35 \times 2 = 0,70$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,70 por 2)

$0,70 \times 2 = 1,40$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,40 e multiplico por 2)

$0,40 \times 2 = 0,80$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,80 por 2)

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$0,40 \times 2 = 0,80$

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$\ldots \qquad \Rightarrow (0,675)_{10} = (0,101011001100110\ldots)_2 = (0,101\overline{0110})_2$

Logo, $(12,675)_{10} = (1100,101011001100110\ldots)_2 = (1100,101\overline{0110})_2$

Representação de Números Reais no Computador

Parte fracionária: $(0,675)_{10} \Rightarrow$ Multiplicações sucessivas

$0,675 \times 2 = 1,35$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,35 e multiplico por 2)

$0,35 \times 2 = 0,70$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,70 por 2)

$0,70 \times 2 = 1,40$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,40 e multiplico por 2)

$0,40 \times 2 = 0,80$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,80 por 2)

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$0,40 \times 2 = 0,80$

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$\ldots \qquad \Rightarrow (0,675)_{10} = (0,101011001100110\ldots)_2 = (0,101\overline{0110})_2$

Logo, $(12,675)_{10} = (1100,101011001100110\ldots)_2 = (1100,101\overline{0110})_2$

Representação de Números Reais no Computador

Parte fracionária: $(0,675)_{10} \Rightarrow$ Multiplicações sucessivas

$0,675 \times 2 = 1,35$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,35 e multiplico por 2)

$0,35 \times 2 = 0,70$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,70 por 2)

$0,70 \times 2 = 1,40$ (Parte inteira: 1 $\Rightarrow$ Subtraio 1 de 1,40 e multiplico por 2)

$0,40 \times 2 = 0,80$ (Parte inteira: 0 $\Rightarrow$ Apenas multiplico 0,80 por 2)

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$0,40 \times 2 = 0,80$

$0,80 \times 2 = 1,60$

$0,60 \times 2 = 1,20$

$0,20 \times 2 = 0,40$

$\ldots \qquad \Rightarrow (0,675)_{10} = (0,101011001100110\ldots)_2 = (0,101\overline{0110})_2$

Logo, $(12,675)_{10} = (1100,101011001100110\ldots)_2 = (1100,101\overline{0110})_2$

Ponto fixo e ponto flutuante

Em todos esses exemplos, a posição da vírgula está fixa, separando a casa das unidades da primeira casa fracionária.

Entretanto, pode-se variar a posição da vírgula, corrigindo-se o valor com a potência da base, seja dez ou dois, dependendo do sistema que se use.

Exemplo 1 (Base decimal):

$(45{,}31)_{10} = 4{,}531 \times 10^1 = 0{,}4531 \times 10^2 = 453{,}1 \times 10^{-1}$

Exemplo 2 (Base binária):

$(1110{,}01)_2 = 1{,}11001 \times 2^3 = 0{,}111001 \times 2^4 = 11100{,}1 \times 2^{-1}$

Chama-se a isso **ponto flutuante (floating point)**, pois no lugar de se deixar sempre a posição da vírgula entre a casa das unidades e a primeira casa decimal, flutua-se a posição da vírgula e corrige-se com a potência da base.

Ponto fixo e ponto flutuante

Em todos esses exemplos, a posição da vírgula está fixa, separando a casa das unidades da primeira casa fracionária.

Entretanto, pode-se variar a posição da vírgula, corrigindo-se o valor com a potência da base, seja dez ou dois, dependendo do sistema que se use.

Exemplo 1 (Base decimal):

$$(45{,}31)_{10} = 4{,}531 \times 10^{1} = 0{,}4531 \times 10^{2} = 453{,}1 \times 10^{-1}$$

Exemplo 2 (Base binária):

$$(1110{,}01)_{2} = 1{,}11001 \times 2^{3} = 0{,}111001 \times 2^{4} = 11100{,}1 \times 2^{-1}$$

Chama-se a isso **ponto flutuante (floating point)**, pois no lugar de se deixar sempre a posição da vírgula entre a casa das unidades e a primeira casa decimal, flutua-se a posição da vírgula e corrige-se com a potência da base.

Ponto fixo e ponto flutuante

Em todos esses exemplos, a posição da vírgula está fixa, separando a casa das unidades da primeira casa fracionária.

Entretanto, pode-se variar a posição da vírgula, corrigindo-se o valor com a potência da base, seja dez ou dois, dependendo do sistema que se use.

Exemplo 1 (Base decimal):

$(45{,}31)_{10} = 4{,}531 \times 10^1 = 0{,}4531 \times 10^2 = 453{,}1 \times 10^{-1}$

Exemplo 2 (Base binária):

$(1110{,}01)_2 = 1{,}11001 \times 2^3 = 0{,}111001 \times 2^4 = 11100{,}1 \times 2^{-1}$

Chama-se a isso **ponto flutuante (floating point)**, pois no lugar de se deixar sempre a posição da vírgula entre a casa das unidades e a primeira casa decimal, flutua-se a posição da vírgula e corrige-se com a potência da base.

Ponto fixo e ponto flutuante

Em todos esses exemplos, a posição da vírgula está fixa, separando a casa das unidades da primeira casa fracionária.

Entretanto, pode-se variar a posição da vírgula, corrigindo-se o valor com a potência da base, seja dez ou dois, dependendo do sistema que se use.

Exemplo 1 (Base decimal):

$$(45{,}31)_{10} = 4{,}531 \times 10^1 = 0{,}4531 \times 10^2 = 453{,}1 \times 10^{-1}$$

Exemplo 2 (Base binária):

$$(1110{,}01)_2 = 1{,}11001 \times 2^3 = 0{,}111001 \times 2^4 = 11100{,}1 \times 2^{-1}$$

Chama-se a isso **ponto flutuante (floating point)**, pois no lugar de se deixar sempre a posição da vírgula entre a casa das unidades e a primeira casa decimal, flutua-se a posição da vírgula e corrige-se com a potência da base.

Ponto fixo e ponto flutuante

Em todos esses exemplos, a posição da vírgula está fixa, separando a casa das unidades da primeira casa fracionária.

Entretanto, pode-se variar a posição da vírgula, corrigindo-se o valor com a potência da base, seja dez ou dois, dependendo do sistema que se use.

Exemplo 1 (Base decimal):

$$(45{,}31)_{10} = 4{,}531 \times 10^{1} = 0{,}4531 \times 10^{2} = 453{,}1 \times 10^{-1}$$

Exemplo 2 (Base binária):

$$(1110{,}01)_{2} = 1{,}11001 \times 2^{3} = 0{,}111001 \times 2^{4} = 11100{,}1 \times 2^{-1}$$

Chama-se a isso **ponto flutuante (floating point)**, pois no lugar de se deixar sempre a posição da vírgula entre a casa das unidades e a primeira casa decimal, flutua-se a posição da vírgula e corrige-se com a potência da base.

Ponto fixo e ponto flutuante

Em todos esses exemplos, a posição da vírgula está fixa, separando a casa das unidades da primeira casa fracionária.

Entretanto, pode-se variar a posição da vírgula, corrigindo-se o valor com a potência da base, seja dez ou dois, dependendo do sistema que se use.

Exemplo 1 (Base decimal):

$$(45{,}31)_{10} = 4{,}531 \times 10^1 = 0{,}4531 \times 10^2 = 453{,}1 \times 10^{-1}$$

Exemplo 2 (Base binária):

$$(1110{,}01)_2 = 1{,}11001 \times 2^3 = 0{,}111001 \times 2^4 = 11100{,}1 \times 2^{-1}$$

Chama-se a isso **ponto flutuante (floating point)**, pois no lugar de se deixar sempre a posição da vírgula entre a casa das unidades e a primeira casa decimal, flutua-se a posição da vírgula e corrige-se com a potência da base.

Ponto fixo e ponto flutuante

Em todos esses exemplos, a posição da vírgula está fixa, separando a casa das unidades da primeira casa fracionária.

Entretanto, pode-se variar a posição da vírgula, corrigindo-se o valor com a potência da base, seja dez ou dois, dependendo do sistema que se use.

Exemplo 1 (Base decimal):

$$(45{,}31)_{10} = 4{,}531 \times 10^1 = 0{,}4531 \times 10^2 = 453{,}1 \times 10^{-1}$$

Exemplo 2 (Base binária):

$$(1110{,}01)_2 = 1{,}11001 \times 2^3 = 0{,}111001 \times 2^4 = 11100{,}1 \times 2^{-1}$$

Chama-se a isso **ponto flutuante (floating point)**, pois no lugar de se deixar sempre a posição da vírgula entre a casa das unidades e a primeira casa decimal, flutua-se a posição da vírgula e corrige-se com a potência da base.

Ponto fixo e ponto flutuante

Em todos esses exemplos, a posição da vírgula está fixa, separando a casa das unidades da primeira casa fracionária.

Entretanto, pode-se variar a posição da vírgula, corrigindo-se o valor com a potência da base, seja dez ou dois, dependendo do sistema que se use.

Exemplo 1 (Base decimal):

$$(45{,}31)_{10} = 4{,}531 \times 10^{1} = 0{,}4531 \times 10^{2} = 453{,}1 \times 10^{-1}$$

Exemplo 2 (Base binária):

$$(1110{,}01)_{2} = 1{,}11001 \times 2^{3} = 0{,}111001 \times 2^{4} = 11100{,}1 \times 2^{-1}$$

Chama-se a isso **ponto flutuante (floating point)**, pois no lugar de se deixar sempre a posição da vírgula entre a casa das unidades e a primeira casa decimal, flutua-se a posição da vírgula e corrige-se com a potência da base.

Ponto fixo e ponto flutuante

Em todos esses exemplos, a posição da vírgula está fixa, separando a casa das unidades da primeira casa fracionária.

Entretanto, pode-se variar a posição da vírgula, corrigindo-se o valor com a potência da base, seja dez ou dois, dependendo do sistema que se use.

Exemplo 1 (Base decimal):

$$(45{,}31)_{10} = 4{,}531 \times 10^1 = 0{,}4531 \times 10^2 = 453{,}1 \times 10^{-1}$$

Exemplo 2 (Base binária):

$$(1110{,}01)_2 = 1{,}11001 \times 2^3 = 0{,}111001 \times 2^4 = 11100{,}1 \times 2^{-1}$$

Chama-se a isso **ponto flutuante (floating point)**, pois no lugar de se deixar sempre a posição da vírgula entre a casa das unidades e a primeira casa decimal, flutua-se a posição da vírgula e corrige-se com a potência da base.

Forma normalizada

Como se vê, há diferentes maneiras de escrever o mesmo número.

Chama-se **forma normalizada** aquela que apresenta **um único dígito diferente de zero** antes da vírgula.

Exemplo 1 na forma normalizada: $(45,31)_{10} = 4,531 \times 10^1$

Exemplo 2 na forma normalizada: $(1110,01)_2 = 1,11001 \times 2^3$

Outros exemplos:

$(110101)_2 = 1,10101x2^5 = (53)_{10}$.

$(0,0011)_2 = 1,1x2^{-3} = (0,1875)_{10}$.

$(0,1001)_2 = 1,001x2^{-1} = (0,5625)_{10}$.

Forma normalizada

Como se vê, há diferentes maneiras de escrever o mesmo número.

Chama-se **forma normalizada** aquela que apresenta
um único dígito diferente de zero antes da vírgula.

Exemplo 1 na forma normalizada: $(45,31)_{10} = 4,531 \times 10^1$

Exemplo 2 na forma normalizada: $(1110,01)_2 = 1,11001 \times 2^3$

Outros exemplos:

$(110101)_2 = 1,10101x2^5 = (53)_{10}$.

$(0,0011)_2 = 1,1x2^{-3} = (0,1875)_{10}$.

$(0,1001)_2 = 1,001x2^{-1} = (0,5625)_{10}$.

Forma normalizada

Como se vê, há diferentes maneiras de escrever o mesmo número.

Chama-se **forma normalizada** aquela que apresenta **um único dígito diferente de zero** antes da vírgula.

Exemplo 1 na forma normalizada: $(45,31)_{10} = 4,531 \times 10^1$

Exemplo 2 na forma normalizada: $(1110,01)_2 = 1,11001 \times 2^3$

Outros exemplos:

$(110101)_2 = 1,10101x2^5 = (53)_{10}$.

$(0,0011)_2 = 1,1x2^{-3} = (0,1875)_{10}$.

$(0,1001)_2 = 1,001x2^{-1} = (0,5625)_{10}$.

Forma normalizada

Como se vê, há diferentes maneiras de escrever o mesmo número.

Chama-se **forma normalizada** aquela que apresenta **um único dígito diferente de zero** antes da vírgula.

Exemplo 1 na forma normalizada: $(45,31)_{10} = 4,531 \times 10^1$

Exemplo 2 na forma normalizada: $(1110,01)_2 = 1,11001 \times 2^3$

Outros exemplos:

$(110101)_2 = 1,10101x2^5 = (53)_{10}$.

$(0,0011)_2 = 1,1x2^{-3} = (0,1875)_{10}$.

$(0,1001)_2 = 1,001x2^{-1} = (0,5625)_{10}$.

Forma normalizada

Como se vê, há diferentes maneiras de escrever o mesmo número.

Chama-se **forma normalizada** aquela que apresenta **um único dígito diferente de zero** antes da vírgula.

Exemplo 1 na forma normalizada: $(45,31)_{10} = 4,531 \times 10^1$

Exemplo 2 na forma normalizada: $(1110,01)_2 = 1,11001 \times 2^3$

Outros exemplos:

$(110101)_2 = 1,10101x2^5 = (53)_{10}$.

$(0,0011)_2 = 1,1x2^{-3} = (0,1875)_{10}$.

$(0,1001)_2 = 1,001x2^{-1} = (0,5625)_{10}$.

Representação de Números Reais no Computador

O computador representa os números em sistema binário.

- A representação é finita
  - Números como o $\pi = 3,1415\ldots$ são aproximados.

Vamos usar a representação em ponto flutuante na forma normalizada.

Exemplo: $110101 = 1,10101 \times 2^5$

Chama-se **mantissa** ao número $1,10101$ e **expoente** ao número 5.

Para se definir a maneira como o computador armazenará o número real em ponto flutuante, é preciso definir o número de bits que ele usará para representar a mantissa e o número de bits para o expoente.

Representação de Números Reais no Computador

O computador representa os números em sistema binário.

- A representação é finita
  - Números como o $\pi = 3,1415\ldots$ são aproximados.

Vamos usar a representação em ponto flutuante na forma normalizada.

Exemplo: $110101 = 1,10101 \times 2^5$

Chama-se **mantissa** ao número $1,10101$ e **expoente** ao número 5.

Para se definir a maneira como o computador armazenará o número real em ponto flutuante, é preciso definir o número de bits que ele usará para representar a mantissa e o número de bits para o expoente.

Representação de Números Reais no Computador

O computador representa os números em sistema binário.

- A representação é finita
  - Números como o $\pi = 3,1415\ldots$ são aproximados.

Vamos usar a representação em ponto flutuante na forma normalizada.

Exemplo: $110101 = 1,10101 \times 2^5$

Chama-se **mantissa** ao número $1,10101$ e **expoente** ao número 5.

Para se definir a maneira como o computador armazenará o número real em ponto flutuante, é preciso definir o número de bits que ele usará para representar a mantissa e o número de bits para o expoente.

Representação de Números Reais no Computador

O computador representa os números em sistema binário.

- A representação é finita
  - Números como o $\pi = 3,1415\ldots$ são aproximados.

Vamos usar a representação em ponto flutuante na forma normalizada.

Exemplo: $110101 = 1,10101 \times 2^5$

Chama-se **mantissa** ao número $1,10101$ e **expoente** ao número 5.

Para se definir a maneira como o computador armazenará o número real em ponto flutuante, é preciso definir o número de bits que ele usará para representar a mantissa e o número de bits para o expoente.

Representação de Números Reais no Computador

O computador representa os números em sistema binário.

- A representação é finita
  - Números como o $\pi = 3,1415\ldots$ são aproximados.

Vamos usar a representação em ponto flutuante na forma normalizada.

Exemplo: $110101 = 1,10101 \times 2^5$

Chama-se **mantissa** ao número $1,10101$ e **expoente** ao número 5.

Para se definir a maneira como o computador armazenará o número real em ponto flutuante, é preciso definir o número de bits que ele usará para representar a mantissa e o número de bits para o expoente.

Representação de Números Reais no Computador

O computador representa os números em sistema binário.

- A representação é finita
  - Números como o $\pi = 3,1415\ldots$ são aproximados.

Vamos usar a representação em ponto flutuante na forma normalizada.

Exemplo: $110101 = 1,10101 \times 2^5$

Chama-se **mantissa** ao número $1,10101$ e **expoente** ao número 5.

Para se definir a maneira como o computador armazenará o número real em ponto flutuante, é preciso definir o número de bits que ele usará para representar a mantissa e o número de bits para o expoente.

Representação de Números Reais no Computador

Para armazenar a mantissa, é dispensável representar o **”1,”**, por estar sempre presente, sendo também desnecessário armazenar o **2**, base do sistema.

Suponha-se que um determinado computador reserve **1 byte**, isto é, **8 bits**, para representar os números reais.

Admita-se que usa o primeiro bit para sinal do número, três bits seguintes para o expoente e os últimos quatro bits para o restante da mantissa.

bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7

sinal do número — expoente — mantissa

Tabela: Representação em ponto flutuante

Representação de Números Reais no Computador

Para armazenar a mantissa, é dispensável representar o **"1,"**, por estar sempre presente, sendo também desnecessário armazenar o **2**, base do sistema.

Suponha-se que um determinado computador reserve **1 byte**, isto é, **8 bits**, para representar os números reais.

Admita-se que usa o primeiro bit para sinal do número, três bits seguintes para o expoente e os últimos quatro bits para o restante da mantissa.

bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7

sinal do número — expoente — mantissa

Tabela: Representação em ponto flutuante

Representação de Números Reais no Computador

Para armazenar a mantissa, é dispensável representar o **”1,”**, por estar sempre presente, sendo também desnecessário armazenar o **2**, base do sistema.

Suponha-se que um determinado computador reserve **1 byte**, isto é, **8 bits**, para representar os números reais.

Admita-se que usa o primeiro bit para sinal do número, três bits seguintes para o expoente e os últimos quatro bits para o restante da mantissa.

bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7

sinal do número — expoente — mantissa

Tabela: Representação em ponto flutuante

Representação de Números Reais no Computador

Para armazenar a mantissa, é dispensável representar o **”1,”**, por estar sempre presente, sendo também desnecessário armazenar o **2**, base do sistema.

Suponha-se que um determinado computador reserve **1 byte**, isto é, **8 bits**, para representar os números reais.

Admita-se que usa o primeiro bit para sinal do número, três bits seguintes para o expoente e os últimos quatro bits para o restante da mantissa.

bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7

sinal do número | expoente | mantissa

Tabela: Representação em ponto flutuante

Representação de Números Reais no Computador

O bit 0 indica o sinal do número: 0 positivo, 1 negativo.

Os bits 1, 2 e 3 constituem o expoente e precisam representar tanto expoentes positivos quanto expoentes negativos.

bits	expoente	valor em decimal
000	Desnormalizado (-2)	
001	-2	1
010	-1	2
011	0	3
100	$+1$	4
101	$+2$	5
110	$+3$	6
111	NaN, Inf., Indet.	

Tabela: Tabela para os bits do expoente do exemplo

Representação de Números Reais no Computador

O bit 0 indica o sinal do número: 0 positivo, 1 negativo.

Os bits 1, 2 e 3 constituem o expoente e precisam representar tanto expoentes positivos quanto expoentes negativos.

bits	expoente	valor em decimal
000	Desnormalizado (-2)	
001	-2	1
010	-1	2
011	0	3
100	$+1$	4
101	$+2$	5
110	$+3$	6
111	NaN, Inf., Indet.	

Tabela: Tabela para os bits do expoente do exemplo

Representação de Números Reais no Computador

O bit 0 indica o sinal do número: 0 positivo, 1 negativo.

Os bits 1, 2 e 3 constituem o expoente e precisam representar tanto expoentes positivos quanto expoentes negativos.

bits	expoente	valor em decimal
000	Desnormalizado (-2)	
001	-2	1
010	-1	2
011	0	3
100	$+1$	4
101	$+2$	5
110	$+3$	6
111	NaN, Inf., Indet.	

Tabela: Tabela para os bits do expoente do exemplo

Representação de Números Reais no Computador

Dessa maneira, o número que representa o expoente será o valor em decimal menos três.

Os bits da mantissa representam os dígitos que aparecem depois da vírgula na forma normalizada.

Exemplo 1:
$(3,5)_{10} = (11,1)_2 = 1,11 \times 2^1 = 1,1100 \times 2^1 \Rightarrow$
bit de sinal: 0; bits do expoente: 100; mantissa: 1100

0	1	0	0	1	1	0	0
sinal do número	expoente			mantissa			

Tabela: Exemplo 1

Representação de Números Reais no Computador

Dessa maneira, o número que representa o expoente será o valor em decimal menos três.

Os bits da mantissa representam os dígitos que aparecem depois da vírgula na forma normalizada.

Exemplo 1:
$(3,5)_{10} = (11,1)_2 = 1,11 \times 2^1 = 1,1100 \times 2^1 \Rightarrow$
bit de sinal: 0; bits do expoente: 100; mantissa: 1100

0	1	0	0	1	1	0	0

$\underbrace{\quad}_{\text{sinal do número}}$ $\underbrace{\quad}_{\text{expoente}}$ $\underbrace{\quad}_{\text{mantissa}}$

Tabela: Exemplo 1

Representação de Números Reais no Computador

Dessa maneira, o número que representa o expoente será o valor em decimal menos três.

Os bits da mantissa representam os dígitos que aparecem depois da vírgula na forma normalizada.

Exemplo 1:
$(3,5)_{10} = (11,1)_2 = 1,11 \times 2^1 = 1,1100 \times 2^1 \Rightarrow$
bit de sinal: 0; bits do expoente: 100; mantissa: 1100

0	1	0	0	1	1	0	0

$\underbrace{\qquad\qquad}_{\text{sinal do número}}$ $\underbrace{\qquad\qquad}_{\text{expoente}}$ $\underbrace{\qquad\qquad}_{\text{mantissa}}$

Tabela: Exemplo 1

Representação de Números Reais no Computador

Exemplo 2:

$(-7,25)_{10} = (-111,01)_2 = -1,1101 \times 2^2 \Rightarrow$

bit de sinal: 1; bits do expoente: 101; mantissa: 1101

1	1	0	1	1	1	0	1
sinal do número	expoente			mantissa			

Tabela: Exemplo 2

Exemplo 3 - Maior número positivo:

0	1	1	0	1	1	1	1
sinal do número	expoente			mantissa			

Tabela: Exemplo 3

$1,1111 \times 2^3 = (15,5)_{10}$

Representação de Números Reais no Computador

Exemplo 2:

$(-7,25)_{10} = (-111,01)_2 = -1,1101 \times 2^2 \Rightarrow$

bit de sinal: $1;$ bits do expoente: $101;$ mantissa: 1101

1	1	0	1	1	1	0	1

$\underbrace{\hspace{3cm}}_{\text{sinal do número}} \quad \underbrace{\hspace{2cm}}_{\text{expoente}} \quad \underbrace{\hspace{2cm}}_{\text{mantissa}}$

Tabela: Exemplo 2

Exemplo 3 - Maior número positivo:

0	1	1	0	1	1	1	1

$\underbrace{\hspace{3cm}}_{\text{sinal do número}} \quad \underbrace{\hspace{2cm}}_{\text{expoente}} \quad \underbrace{\hspace{2cm}}_{\text{mantissa}}$

Tabela: Exemplo 3

$1,1111 \times 2^3 = (15,5)_{10}$

Representação de Números Reais no Computador

Exemplo 2:

$(-7,25)_{10} = (-111,01)_2 = -1,1101 \times 2^2 \Rightarrow$

bit de sinal: 1; bits do expoente: 101; mantissa: 1101

1	1	0	1	1	1	0	1

sinal do número expoente mantissa

Tabela: Exemplo 2

Exemplo 3 - Maior número positivo:

0	1	1	0	1	1	1	1

sinal do número expoente mantissa

Tabela: Exemplo 3

$1,1111 \times 2^3 = (15,5)_{10}$

Representação de Números Reais no Computador

Exemplo 2:

$(-7,25)_{10} = (-111,01)_2 = -1,1101 \times 2^2 \Rightarrow$

bit de sinal: 1; bits do expoente: 101; mantissa: 1101

1	1	0	1	1	1	0	1

$\underbrace{\hspace{3em}}_{\text{sinal do número}}$ $\underbrace{\hspace{3em}}_{\text{expoente}}$ $\underbrace{\hspace{3em}}_{\text{mantissa}}$

Tabela: Exemplo 2

Exemplo 3 - Maior número positivo:

0	1	1	0	1	1	1	1

$\underbrace{\hspace{3em}}_{\text{sinal do número}}$ $\underbrace{\hspace{3em}}_{\text{expoente}}$ $\underbrace{\hspace{3em}}_{\text{mantissa}}$

Tabela: Exemplo 3

$1,1111 \times 2^3 = (15,5)_{10}$

Representação de Números Reais no Computador

Exemplo 4 - Menor número positivo:

Devemos lembrar que os expoentes 000 e 111 possuem tratamento especial.

Usa-se o expoente 000 para indicar que o número não está normalizado.

Assim, 000 será configurado como o menor expoente, isto é, -2 neste exemplo.

Na forma desnormalizada, a mantissa passa a ser $0, b_1 b_2 b_3 b_4$, onde $b_1, \ldots, b_4$ são os bits da mantissa neste exemplo.

Assim, o menor número positivo armazenado pelo computador de 8 bits recebe a seguinte configuração:

0	0	0	0	0	0	0	1

$\underbrace{}_{\text{sinal do número}}$ $\underbrace{}_{\text{expoente}}$ $\underbrace{}_{\text{mantissa}}$

Tabela: Exemplo 4

Representação de Números Reais no Computador

Exemplo 4 - Menor número positivo:

Devemos lembrar que os expoentes 000 e 111 possuem tratamento especial.

Usa-se o expoente 000 para indicar que o número não está normalizado.

Assim, 000 será configurado como o menor expoente, isto é, -2 neste exemplo.

Na forma desnormalizada, a mantissa passa a ser $0, b_1 b_2 b_3 b_4$, onde $b_1, \ldots, b_4$ são os bits da mantissa neste exemplo.

Assim, o menor número positivo armazenado pelo computador de 8 bits recebe a seguinte configuração:

0	0	0	0	0	0	0	1
sinal do número	expoente			mantissa			

Tabela: Exemplo 4

Representação de Números Reais no Computador

Exemplo 4 - Menor número positivo:

Devemos lembrar que os expoentes 000 e 111 possuem tratamento especial.

Usa-se o expoente 000 para indicar que o número não está normalizado.

Assim, 000 será configurado como o menor expoente, isto é, -2 neste exemplo.

Na forma desnormalizada, a mantissa passa a ser $0, b_1 b_2 b_3 b_4$, onde $b_1, \ldots, b_4$ são os bits da mantissa neste exemplo.

Assim, o menor número positivo armazenado pelo computador de 8 bits recebe a seguinte configuração:

0	0	0	0	0	0	0	1
sinal do número	expoente			mantissa			

Tabela: Exemplo 4

Representação de Números Reais no Computador

Exemplo 4 - Menor número positivo:

Devemos lembrar que os expoentes 000 e 111 possuem tratamento especial.

Usa-se o expoente 000 para indicar que o número não está normalizado.

Assim, 000 será configurado como o menor expoente, isto é, -2 neste exemplo.

Na forma desnormalizada, a mantissa passa a ser $0, b_1 b_2 b_3 b_4$, onde $b_1, \ldots, b_4$ são os bits da mantissa neste exemplo.

Assim, o menor número positivo armazenado pelo computador de 8 bits recebe a seguinte configuração:

0	0	0	0	0	0	0	1
sinal do número	expoente			mantissa			

Tabela: Exemplo 4

Representação de Números Reais no Computador

Exemplo 4 - Menor número positivo:

Devemos lembrar que os expoentes 000 e 111 possuem tratamento especial.

Usa-se o expoente 000 para indicar que o número não está normalizado.

Assim, 000 será configurado como o menor expoente, isto é, -2 neste exemplo.

Na forma desnormalizada, a mantissa passa a ser $0, b_1 b_2 b_3 b_4$, onde $b_1, \ldots, b_4$ são os bits da mantissa neste exemplo.

Assim, o menor número positivo armazenado pelo computador de 8 bits recebe a seguinte configuração:

0	0	0	0	0	0	0	1
sinal do número	expoente			mantissa			

Tabela: Exemplo 4

Representação de Números Reais no Computador

Exemplo 4 - Menor número positivo:

Devemos lembrar que os expoentes 000 e 111 possuem tratamento especial.

Usa-se o expoente 000 para indicar que o número não está normalizado.

Assim, 000 será configurado como o menor expoente, isto é, -2 neste exemplo.

Na forma desnormalizada, a mantissa passa a ser $0, b_1 b_2 b_3 b_4$, onde $b_1, \ldots, b_4$ são os bits da mantissa neste exemplo.

Assim, o menor número positivo armazenado pelo computador de 8 bits recebe a seguinte configuração:

0	0	0	0	0	0	0	1
sinal do número	expoente			mantissa			

Tabela: Exemplo 4

Representação de Números Reais no Computador

Exemplo 4 - Menor número positivo:

Devemos lembrar que os expoentes 000 e 111 possuem tratamento especial.

Usa-se o expoente 000 para indicar que o número não está normalizado.

Assim, 000 será configurado como o menor expoente, isto é, -2 neste exemplo.

Na forma desnormalizada, a mantissa passa a ser $0, b_1 b_2 b_3 b_4$, onde $b_1, \ldots, b_4$ são os bits da mantissa neste exemplo.

Assim, o menor número positivo armazenado pelo computador de 8 bits recebe a seguinte configuração:

0	0	0	0	0	0	0	1
sinal do número	expoente			mantissa			

Tabela: Exemplo 4

Representação de Números Reais no Computador

A configuração

0	0	0	0	0	0	0	0

representa $+0$,

enquanto a configuração

1	0	0	0	0	0	0	0

representa -0, devendo ambos serem reconhecidos como iguais nas comparações.

O expoente 111 é reservado para representar $+\infty$,

0	1	1	1	0	0	0	0

e $-\infty$, que é representado por

1	1	1	1	0	0	0	0

bastando trocar o bit de sinal do número para 1 por ser negativo.
As demais combinações com o expoente 111 não são válidas, sendo consideradas (NaN, ou Not a Number).

Representação de Números Reais no Computador

A configuração

0	0	0	0	0	0	0	0

representa $+0$,

enquanto a configuração

1	0	0	0	0	0	0	0

representa -0, devendo ambos serem reconhecidos como iguais nas comparações.

O expoente 111 é reservado para representar $+\infty$,

0	1	1	1	0	0	0	0

e $-\infty$, que é representado por

1	1	1	1	0	0	0	0

bastando trocar o bit de sinal do número para 1 por ser negativo.
As demais combinações com o expoente 111 não são válidas, sendo consideradas (NaN, ou Not a Number).

Representação de Números Reais no Computador

A configuração

0	0	0	0	0	0	0	0

representa $+0$,

enquanto a configuração

1	0	0	0	0	0	0	0

representa -0, devendo ambos serem reconhecidos como iguais nas comparações.

O expoente 111 é reservado para representar $+\infty$,

0	1	1	1	0	0	0	0

e $-\infty$, que é representado por

1	1	1	1	0	0	0	0

bastando trocar o bit de sinal do número para 1 por ser negativo.
As demais combinações com o expoente 111 não são válidas, sendo consideradas (NaN, ou Not a Number).

Representação de Números Reais no Computador

E quando a mantissa não cabe nos 4 bits?

Somos obrigados a arredondar a mantissa para que ela caiba nos 4 bits.

Vamos, assim, perder precisão no número e ele não mais representará, exatamente, o número desejado.

Exemplo 5:
$(9,375)_{10} = (1001,011)_2 = 1,001011 \times 2^3$

$\Rightarrow$ Possui 6 dígitos depois da vírgula, mas o computador só tem reservados 4 bits para a mantissa.

$\Rightarrow$ Deve ser feito o arredondamento para 4 casas decimais: se o dígito abandonado após as 4 casas decimais for menor que 1 (ou seja, se for 0), os dígitos anteriores devem ser mantidos. Se o dígito abandonado for igual a 1, somo 1 ao dígito anterior.

Representação de Números Reais no Computador

E quando a mantissa não cabe nos 4 bits?

Somos obrigados a arredondar a mantissa para que ela caiba nos 4 bits.

Vamos, assim, perder precisão no número e ele não mais representará, exatamente, o número desejado.

Exemplo 5:
$(9,375)_{10} = (1001,011)_2 = 1,001011 \times 2^3$

$\Rightarrow$ Possui 6 dígitos depois da vírgula, mas o computador só tem reservados 4 bits para a mantissa.

$\Rightarrow$ Deve ser feito o arredondamento para 4 casas decimais: se o dígito abandonado após as 4 casas decimais for menor que 1 (ou seja, se for 0), os dígitos anteriores devem ser mantidos. Se o dígito abandonado for igual a 1, somo 1 ao dígito anterior.

Representação de Números Reais no Computador

E quando a mantissa não cabe nos 4 bits?

Somos obrigados a arredondar a mantissa para que ela caiba nos 4 bits.

Vamos, assim, perder precisão no número e ele não mais representará, exatamente, o número desejado.

Exemplo 5:
$(9,375)_{10} = (1001,011)_2 = 1,001011 \times 2^3$

$\Rightarrow$ Possui 6 dígitos depois da vírgula, mas o computador só tem reservados 4 bits para a mantissa.

$\Rightarrow$ Deve ser feito o arredondamento para 4 casas decimais: se o dígito abandonado após as 4 casas decimais for menor que 1 (ou seja, se for 0), os dígitos anteriores devem ser mantidos. Se o dígito abandonado for igual a 1, somo 1 ao dígito anterior.

Representação de Números Reais no Computador

E quando a mantissa não cabe nos 4 bits?

Somos obrigados a arredondar a mantissa para que ela caiba nos 4 bits.

Vamos, assim, perder precisão no número e ele não mais representará, exatamente, o número desejado.

Exemplo 5:
$(9,375)_{10} = (1001,011)_2 = 1,001011 \times 2^3$

$\Rightarrow$ Possui 6 dígitos depois da vírgula, mas o computador só tem reservados 4 bits para a mantissa.

$\Rightarrow$ Deve ser feito o arredondamento para 4 casas decimais: se o dígito abandonado após as 4 casas decimais for menor que 1 (ou seja, se for 0), os dígitos anteriores devem ser mantidos. Se o dígito abandonado for igual a 1, somo 1 ao dígito anterior.

Representação de Números Reais no Computador

E quando a mantissa não cabe nos 4 bits?

Somos obrigados a arredondar a mantissa para que ela caiba nos 4 bits.

Vamos, assim, perder precisão no número e ele não mais representará, exatamente, o número desejado.

Exemplo 5:
$(9,375)_{10} = (1001,011)_2 = 1,001011 \times 2^3$

$\Rightarrow$ Possui 6 dígitos depois da vírgula, mas o computador só tem reservados 4 bits para a mantissa.

$\Rightarrow$ Deve ser feito o arredondamento para 4 casas decimais: se o dígito abandonado após as 4 casas decimais for menor que 1 (ou seja, se for 0), os dígitos anteriores devem ser mantidos. Se o dígito abandonado for igual a 1, somo 1 ao dígito anterior.

Representação de Números Reais no Computador

E quando a mantissa não cabe nos 4 bits?

Somos obrigados a arredondar a mantissa para que ela caiba nos 4 bits.

Vamos, assim, perder precisão no número e ele não mais representará, exatamente, o número desejado.

Exemplo 5:
$(9,375)_{10} = (1001,011)_2 = 1,001011 \times 2^3$

$\Rightarrow$ Possui 6 dígitos depois da vírgula, mas o computador só tem reservados 4 bits para a mantissa.

$\Rightarrow$ Deve ser feito o arredondamento para 4 casas decimais: se o dígito abandonado após as 4 casas decimais for menor que 1 (ou seja, se for 0), os dígitos anteriores devem ser mantidos. Se o dígito abandonado for igual a 1, somo 1 ao dígito anterior.

Representação de Números Reais no Computador

Exemplo 5:

$(9,375)_{10} = (1001,011)_2 = 1,001011 \times 2^3 = 1,0011 \times 2^3$

0	1	1	0	0	0	1	1
sinal do número	expoente			mantissa			

Exemplo 6: $(9,125)_{10} = (1001,001)_2 = 1,001001 \times 2^3 = 1,0010 \times 2^3$

0	1	1	0	0	0	1	0
sinal do número	expoente			mantissa			

Representação de Números Reais no Computador

Exemplo 5:
$(9,375)_{10} = (1001,011)_2 = 1,001011 \times 2^3 = 1,0011 \times 2^3$

0	1	1	0	0	0	1	1

$\underbrace{\hphantom{0000}}_{\text{sinal do número}}$ $\underbrace{\hphantom{000}}_{\text{expoente}}$ $\underbrace{\hphantom{000}}_{\text{mantissa}}$

Exemplo 6: $(9,125)_{10} = (1001,001)_2 = 1,001001 \times 2^3 = 1,0010 \times 2^3$

0	1	1	0	0	0	1	0

$\underbrace{\hphantom{0000}}_{\text{sinal do número}}$ $\underbrace{\hphantom{000}}_{\text{expoente}}$ $\underbrace{\hphantom{000}}_{\text{mantissa}}$

Representação de Números Reais no Computador

Exemplo 5:
$(9,375)_{10} = (1001,011)_2 = 1,001011 \times 2^3 = 1,0011 \times 2^3$

$$\underbrace{0}_{\text{sinal do número}}\ \underbrace{1\ 1\ 0}_{\text{expoente}}\ \underbrace{0\ 0\ 1\ 1}_{\text{mantissa}}$$

Exemplo 6: $(9,125)_{10} = (1001,001)_2 = 1,001001 \times 2^3 = 1,0010 \times 2^3$

$$\underbrace{0}_{\text{sinal do número}}\ \underbrace{1\ 1\ 0}_{\text{expoente}}\ \underbrace{0\ 0\ 1\ 0}_{\text{mantissa}}$$

Representação de Números Reais no Computador

Exemplo 5:
$(9,375)_{10} = (1001,011)_2 = 1,001011 \times 2^3 = 1,0011 \times 2^3$

0	1	1	0	0	0	1	1

sinal do número expoente mantissa

Exemplo 6: $(9,125)_{10} = (1001,001)_2 = 1,001001 \times 2^3 = 1,0010 \times 2^3$

0	1	1	0	0	0	1	0

sinal do número expoente mantissa

Representação de Números Reais no Computador

Exemplo 5:
$(9,375)_{10} = (1001,011)_2 = 1,001011 \times 2^3 = 1,0011 \times 2^3$

0	1	1	0	0	0	1	1

sinal do número expoente mantissa

Exemplo 6: $(9,125)_{10} = (1001,001)_2 = 1,001001 \times 2^3 = 1,0010 \times 2^3$

0	1	1	0	0	0	1	0

sinal do número expoente mantissa

Representação de Números Reais no Computador

Exemplo 5:
$(9,375)_{10} = (1001,011)_2 = 1,001011 \times 2^3 = 1,0011 \times 2^3$

$$\underbrace{\boxed{0}}_{\text{sinal do número}}\ \underbrace{\boxed{1}\boxed{1}\boxed{0}}_{\text{expoente}}\ \underbrace{\boxed{0}\boxed{0}\boxed{1}\boxed{1}}_{\text{mantissa}}$$

Exemplo 6: $(9,125)_{10} = (1001,001)_2 = 1,001001 \times 2^3 = 1,0010 \times 2^3$

$$\underbrace{\boxed{0}}_{\text{sinal do número}}\ \underbrace{\boxed{1}\boxed{1}\boxed{0}}_{\text{expoente}}\ \underbrace{\boxed{0}\boxed{0}\boxed{1}\boxed{0}}_{\text{mantissa}}$$

Representação de Números Reais no Computador

Vamos ver como ficariam os expoentes da tabela se passarmos a usar um computador que opera com 1 bit para o sinal do número, 5 bits para o expoente e 6 bits para a mantissa.

Por padrão, o expoente 0 recebe a configuração em que o primeiro bit é igual a 0 e os restantes são iguais a 1. Assim, neste exemplo,

A configuração 01111 recebe o valor 0 para o expoente.

Sabemos que 01111 em decimal é 15, pois a configuração seguinte, 10000, vale $1 \times 2^4 = 16$ em decimal. Logo, $(01111)_2 = (16)_{10} - 1 = (15)_{10}$.

Logo, a configuração 01111 recebe o valor $0 = 15 - 15$ para o expoente.

Como os valores em binário e decimal na tabela são consecutivos e uso a convenção de que 01111 representa o expoente 0, então, para as demais configurações de bits, o valor do expoente será sempre, neste exemplo, o valor em decimal menos 15.

Representação de Números Reais no Computador

Vamos ver como ficariam os expoentes da tabela se passarmos a usar um computador que opera com 1 bit para o sinal do número, 5 bits para o expoente e 6 bits para a mantissa.

Por padrão, o expoente 0 recebe a configuração em que o primeiro bit é igual a 0 e os restantes são iguais a 1. Assim, neste exemplo,

A configuração 01111 recebe o valor 0 para o expoente.

Sabemos que 01111 em decimal é 15, pois a configuração seguinte, 10000, vale $1 \times 2^4 = 16$ em decimal. Logo, $(01111)_2 = (16)_{10} - 1 = (15)_{10}$.

Logo, a configuração 01111 recebe o valor $0 = 15 - 15$ para o expoente.

Como os valores em binário e decimal na tabela são consecutivos e uso a convenção de que 01111 representa o expoente 0, então, para as demais configurações de bits, o valor do expoente será sempre, neste exemplo, o valor em decimal menos 15.

Representação de Números Reais no Computador

Vamos ver como ficariam os expoentes da tabela se passarmos a usar um computador que opera com 1 bit para o sinal do número, 5 bits para o expoente e 6 bits para a mantissa.

Por padrão, o expoente 0 recebe a configuração em que o primeiro bit é igual a 0 e os restantes são iguais a 1. Assim, neste exemplo,

A configuração 01111 recebe o valor 0 para o expoente.

Sabemos que 01111 em decimal é 15, pois a configuração seguinte, 10000, vale $1 \times 2^4 = 16$ em decimal. Logo, $(01111)_2 = (16)_{10} - 1 = (15)_{10}$.

Logo, a configuração 01111 recebe o valor $0 = 15 - 15$ para o expoente.

Como os valores em binário e decimal na tabela são consecutivos e uso a convenção de que 01111 representa o expoente 0, então, para as demais configurações de bits, o valor do expoente será sempre, neste exemplo, o valor em decimal menos 15.

Representação de Números Reais no Computador

Vamos ver como ficariam os expoentes da tabela se passarmos a usar um computador que opera com 1 bit para o sinal do número, 5 bits para o expoente e 6 bits para a mantissa.

Por padrão, o expoente 0 recebe a configuração em que o primeiro bit é igual a 0 e os restantes são iguais a 1. Assim, neste exemplo,

A configuração 01111 recebe o valor 0 para o expoente.

Sabemos que 01111 em decimal é 15, pois a configuração seguinte, 10000, vale $1 \times 2^4 = 16$ em decimal. Logo, $(01111)_2 = (16)_{10} - 1 = (15)_{10}$.

Logo, a configuração 01111 recebe o valor $0 = 15 - 15$ para o expoente.

Como os valores em binário e decimal na tabela são consecutivos e uso a convenção de que 01111 representa o expoente 0, então, para as demais configurações de bits, o valor do expoente será sempre, neste exemplo, o valor em decimal menos 15.

Representação de Números Reais no Computador

Vamos ver como ficariam os expoentes da tabela se passarmos a usar um computador que opera com 1 bit para o sinal do número, 5 bits para o expoente e 6 bits para a mantissa.

Por padrão, o expoente 0 recebe a configuração em que o primeiro bit é igual a 0 e os restantes são iguais a 1. Assim, neste exemplo,

A configuração 01111 recebe o valor 0 para o expoente.

Sabemos que 01111 em decimal é 15, pois a configuração seguinte, 10000, vale $1 \times 2^4 = 16$ em decimal. Logo, $(01111)_2 = (16)_{10} - 1 = (15)_{10}$.

Logo, a configuração 01111 recebe o valor $0 = 15 - 15$ para o expoente.

Como os valores em binário e decimal na tabela são consecutivos e uso a convenção de que 01111 representa o expoente 0, então, para as demais configurações de bits, o valor do expoente será sempre, neste exemplo, o valor em decimal menos 15.

Representação de Números Reais no Computador

Seguindo essa lógica, o maior expoente do computador, que é sempre aquele em que o último bit vale 0 e os anteriores valem 1, fica representado da seguinte maneira:

11110 em decimal: $1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 30$.

Logo, a configuração 11110 recebe o valor $30 - 15 = 15$ para o expoente.

Uma outra maneira mais esperta de obter o valor do expoente para 11110 é saber que a configuração seguinte da tabela vale 11111 e depois na sequência vem 100000, que não está na tabela e vale $2^5 = 32$ em decimal. Logo, 11111 em decimal vale $32 - 1 = 31$ e 11110 em decimal vale $31 - 1 = 30$. Portanto, a configuração 01111 recebe o expoente $30 - 15 - 15$.

O **menor expoente na forma normalizada** é dado pela configuração 00001, que em decimal vale 1. Logo, essa configuração recebe o expoente $1 - 15 = -14$.

O **menor expoente na forma desnormalizada** é dado pela configuração 00000, e recebe também o valor -14.

Representação de Números Reais no Computador

Seguindo essa lógica, o maior expoente do computador, que é sempre aquele em que o último bit vale 0 e os anteriores valem 1, fica representado da seguinte maneira:

11110 em decimal: $1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 30.$

Logo, a configuração 11110 recebe o valor $30 - 15 = 15$ para o expoente.

Uma outra maneira mais esperta de obter o valor do expoente para 11110 é saber que a configuração seguinte da tabela vale 11111 e depois na sequência vem 100000, que não está na tabela e vale $2^5 = 32$ em decimal. Logo, 11111 em decimal vale $32 - 1 = 31$ e 11110 em decimal vale $31 - 1 = 30$. Portanto, a configuração 01111 recebe o expoente $30 - 15 - 15$.

O **menor expoente na forma normalizada** é dado pela configuração 00001, que em decimal vale 1. Logo, essa configuração recebe o expoente $1 - 15 = -14$.

O **menor expoente na forma desnormalizada** é dado pela configuração 00000, e recebe também o valor -14.

Representação de Números Reais no Computador

Seguindo essa lógica, o maior expoente do computador, que é sempre aquele em que o último bit vale 0 e os anteriores valem 1, fica representado da seguinte maneira:

11110 em decimal: $1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 30$.

Logo, a configuração 11110 recebe o valor $30 - 15 = 15$ para o expoente.

Uma outra maneira mais esperta de obter o valor do expoente para 11110 é saber que a configuração seguinte da tabela vale 11111 e depois na sequência vem 100000, que não está na tabela e vale $2^5 = 32$ em decimal. Logo, 11111 em decimal vale $32 - 1 = 31$ e 11110 em decimal vale $31 - 1 = 30$. Portanto, a configuração 01111 recebe o expoente $30 - 15 - 15$.

O **menor expoente na forma normalizada** é dado pela configuração 00001, que em decimal vale 1. Logo, essa configuração recebe o expoente $1 - 15 = -14$.

O **menor expoente na forma desnormalizada** é dado pela configuração 00000, e recebe também o valor -14.

Representação de Números Reais no Computador

Seguindo essa lógica, o maior expoente do computador, que é sempre aquele em que o último bit vale 0 e os anteriores valem 1, fica representado da seguinte maneira:

11110 em decimal: $1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 30$.

Logo, a configuração 11110 recebe o valor $30 - 15 = 15$ para o expoente.

Uma outra maneira mais esperta de obter o valor do expoente para 11110 é saber que a configuração seguinte da tabela vale 11111 e depois na sequência vem 100000, que não está na tabela e vale $2^5 = 32$ em decimal. Logo, 11111 em decimal vale $32 - 1 = 31$ e 11110 em decimal vale $31 - 1 = 30$. Portanto, a configuração 01111 recebe o expoente $30 - 15 - 15$.

O **menor expoente na forma normalizada** é dado pela configuração 00001, que em decimal vale 1. Logo, essa configuração recebe o expoente $1 - 15 = -14$.

O **menor expoente na forma desnormalizada** é dado pela configuração 00000, e recebe também o valor -14.

Representação de Números Reais no Computador

Seguindo essa lógica, o maior expoente do computador, que é sempre aquele em que o último bit vale 0 e os anteriores valem 1, fica representado da seguinte maneira:

11110 em decimal: $1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 30$.

Logo, a configuração 11110 recebe o valor $30 - 15 = 15$ para o expoente.

Uma outra maneira mais esperta de obter o valor do expoente para 11110 é saber que a configuração seguinte da tabela vale 11111 e depois na sequência vem 100000, que não está na tabela e vale $2^5 = 32$ em decimal. Logo, 11111 em decimal vale $32 - 1 = 31$ e 11110 em decimal vale $31 - 1 = 30$. Portanto, a configuração 01111 recebe o expoente $30 - 15 - 15$.

O **menor expoente na forma normalizada** é dado pela configuração 00001, que em decimal vale 1. Logo, essa configuração recebe o expoente $1 - 15 = -14$.

O **menor expoente na forma desnormalizada** é dado pela configuração 00000, e recebe também o valor -14.

Representação de Números Reais no Computador

bits	expoente	valor em decimal
00000	Desnormalizado (-14)	
00001	$-14 = 1 - 15$	1
00010	$-13 = 2 - 15$	2
...	...	...
01101	$-2 = 13 - 15$	13
01110	$-1 = 14 - 15$	14
01111	$0 = 15 - 15$	15
10000	$+1 = 16 - 15$	16
10001	$+2 = 17 - 15$	17
...	...	...
11110	$+15 = 30 - 15$	30
11111	NaN, Inf., Indet.	

Tabela: Tabela para os bits do expoente do exemplo

Representação de Números Reais no Computador

bits	expoente	valor em decimal
00000	Desnormalizado (-14)	
00001	-14	$1 = (2^4 - 1) - 14$
00010	-13	$2 = (2^4 - 1) - 13$
...	...	...
01101	-2	$13 = (2^4 - 1) - 2$
01110	-1	$14 = (2^4 - 1) - 1$
01111	0	$15 = 2^4 - 1$
10000	$+1$	$16 = (2^4 - 1) + 1$
10001	$+2$	$17 = (2^4 - 1) + 2$
...	...	...
11110	$+15$	$30 = (2^4 - 1) + 15$
11111	NaN, Inf., Indet.	

Tabela: Tabela para os bits do expoente do exemplo

Representação de Números Reais no Computador

Sem olhar para a tabela, qual seria a configuração de bits para o expoente -7?

Maneira 1:

Vimos na tabela que: $E = D - D_{zero}$, onde:

- E: Valor do expoente. Neste exemplo, $E = -7$.
- D: Valor em decimal da representação em binário dos bits do expoente.
- D_{zero}: Valor em decimal da representação em binário dos bits do expoente zero. Aqui, $D_{zero} = (01111)_2 = 2^4 - 1 = (15)_{10}$.

Logo, $E = D - D_{zero} \Rightarrow -7 = D - 15 \Rightarrow D = (8)_{10} = (01000)_2$.

Portanto, 01000 é a configuração de bits para o expoente -7 neste exemplo.

Maneira 2:

$E = D - D_{zero} \Rightarrow D = D_{zero} + E$

- $D_{zero} = (01111)_2 = 2^4 - 1 = (15)_{10}$
- $E = -7$

Logo, $D = 15 - 7 = (2^4 - 1) - 7 = (8)_{10} = (01000)_2$

Representação de Números Reais no Computador

Sem olhar para a tabela, qual seria a configuração de bits para o expoente -7?

Maneira 1:

Vimos na tabela que: $E = D - D_{zero}$, onde:

- E: Valor do expoente. Neste exemplo, $E = -7$.
- D: Valor em decimal da representação em binário dos bits do expoente.
- D_{zero}: Valor em decimal da representação em binário dos bits do expoente zero. Aqui, $D_{zero} = (01111)_2 = 2^4 - 1 = (15)_{10}$.

Logo, $E = D - D_{zero} \Rightarrow -7 = D - 15 \Rightarrow D = (8)_{10} = (01000)_2$.

Portanto, 01000 é a configuração de bits para o expoente -7 neste exemplo.

Maneira 2:

$E = D - D_{zero} \Rightarrow D = D_{zero} + E$

- $D_{zero} = (01111)_2 = 2^4 - 1 = (15)_{10}$
- $E = -7$

Logo, $D = 15 - 7 = (2^4 - 1) - 7 = (8)_{10} = (01000)_2$

Representação de Números Reais no Computador

Sem olhar para a tabela, qual seria a configuração de bits para o expoente -7?

Maneira 1:

Vimos na tabela que: $E = D - D_{zero}$, onde:

- E: Valor do expoente. Neste exemplo, $E = -7$.
- D: Valor em decimal da representação em binário dos bits do expoente.
- D_{zero}: Valor em decimal da representação em binário dos bits do expoente zero. Aqui, $D_{zero} = (01111)_2 = 2^4 - 1 = (15)_{10}$.

Logo, $E = D - D_{zero} \Rightarrow -7 = D - 15 \Rightarrow D = (8)_{10} = (01000)_2$.

Portanto, 01000 é a configuração de bits para o expoente -7 neste exemplo.

Maneira 2:

$E = D - D_{zero} \Rightarrow D = D_{zero} + E$

- $D_{zero} = (01111)_2 = 2^4 - 1 = (15)_{10}$
- $E = -7$

Logo, $D = 15 - 7 = (2^4 - 1) - 7 = (8)_{10} = (01000)_2$

Representação de Números Reais no Computador

Sem olhar para a tabela, qual seria a configuração de bits para o expoente -7?

Maneira 1:

Vimos na tabela que: $E = D - D_{zero}$, onde:

- E: Valor do expoente. Neste exemplo, $E = -7$.
- D: Valor em decimal da representação em binário dos bits do expoente.
- D_{zero}: Valor em decimal da representação em binário dos bits do expoente zero. Aqui, $D_{zero} = (01111)_2 = 2^4 - 1 = (15)_{10}$.

Logo, $E = D - D_{zero} \Rightarrow -7 = D - 15 \Rightarrow D = (8)_{10} = (01000)_2$.

Portanto, 01000 é a configuração de bits para o expoente -7 neste exemplo.

Maneira 2:

$E = D - D_{zero} \Rightarrow D = D_{zero} + E$

- $D_{zero} = (01111)_2 = 2^4 - 1 = (15)_{10}$
- $E = -7$

Logo, $D = 15 - 7 = (2^4 - 1) - 7 = (8)_{10} = (01000)_2$

Representação de Números Reais no Computador

Sem olhar para a tabela, qual seria a configuração de bits para o expoente -7?

Maneira 1:

Vimos na tabela que: $E = D - D_{zero}$, onde:

- E: Valor do expoente. Neste exemplo, $E = -7$.
- D: Valor em decimal da representação em binário dos bits do expoente.
- D_{zero}: Valor em decimal da representação em binário dos bits do expoente zero. Aqui, $D_{zero} = (01111)_2 = 2^4 - 1 = (15)_{10}$.

Logo, $E = D - D_{zero} \Rightarrow -7 = D - 15 \Rightarrow D = (8)_{10} = (01000)_2$.

Portanto, 01000 é a configuração de bits para o expoente -7 neste exemplo.

Maneira 2:

$E = D - D_{zero} \Rightarrow D = D_{zero} + E$

- $D_{zero} = (01111)_2 = 2^4 - 1 = (15)_{10}$
- $E = -7$

Logo, $D = 15 - 7 = (2^4 - 1) - 7 = (8)_{10} = (01000)_2$

Representação de Números Reais no Computador

Sem olhar para a tabela, qual seria a configuração de bits para o expoente -7?

Maneira 1:

Vimos na tabela que: $E = D - D_{zero}$, onde:

- E: Valor do expoente. Neste exemplo, $E = -7$.
- D: Valor em decimal da representação em binário dos bits do expoente.
- D_{zero}: Valor em decimal da representação em binário dos bits do expoente zero. Aqui, $D_{zero} = (01111)_2 = 2^4 - 1 = (15)_{10}$.

Logo, $E = D - D_{zero} \Rightarrow -7 = D - 15 \Rightarrow D = (8)_{10} = (01000)_2$.

Portanto, 01000 é a configuração de bits para o expoente -7 neste exemplo.

Maneira 2:

$E = D - D_{zero} \Rightarrow D = D_{zero} + E$

- $D_{zero} = (01111)_2 = 2^4 - 1 = (15)_{10}$
- $E = -7$

Logo, $D = 15 - 7 = (2^4 - 1) - 7 = (8)_{10} = (01000)_2$

Representação de Números Reais no Computador

Sem olhar para a tabela, qual seria a configuração de bits para o expoente -7?

Maneira 1:

Vimos na tabela que: $E = D - D_{zero}$, onde:

• E: Valor do expoente. Neste exemplo, $E = -7$.
• D: Valor em decimal da representação em binário dos bits do expoente.
• D_{zero}: Valor em decimal da representação em binário dos bits do expoente zero. Aqui, $D_{zero} = (01111)_2 = 2^4 - 1 = (15)_{10}$.

Logo, $E = D - D_{zero} \Rightarrow -7 = D - 15 \Rightarrow D = (8)_{10} = (01000)_2$.

Portanto, 01000 é a configuração de bits para o expoente -7 neste exemplo.

Maneira 2:

$E = D - D_{zero} \Rightarrow D = D_{zero} + E$

• $D_{zero} = (01111)_2 = 2^4 - 1 = (15)_{10}$
• $E = -7$

Logo, $D = 15 - 7 = (2^4 - 1) - 7 = (8)_{10} = (01000)_2$.

Representação de Números Reais no Computador

Sem olhar para a tabela, qual seria a configuração de bits para o expoente -7?

Maneira 1:

Vimos na tabela que: $E = D - D_{zero}$, onde:

• E: Valor do expoente. Neste exemplo, $E = -7$.
• D: Valor em decimal da representação em binário dos bits do expoente.
• D_{zero}: Valor em decimal da representação em binário dos bits do expoente zero. Aqui, $D_{zero} = (01111)_2 = 2^4 - 1 = (15)_{10}$.

Logo, $E = D - D_{zero} \Rightarrow -7 = D - 15 \Rightarrow D = (8)_{10} = (01000)_2$.

Portanto, 01000 é a configuração de bits para o expoente -7 neste exemplo.

Maneira 2:

$E = D - D_{zero} \Rightarrow D = D_{zero} + E$
• $D_{zero} = (01111)_2 = 2^4 - 1 = (15)_{10}$
• $E = -7$
Logo, $D = 15 - 7 = (2^4 - 1) - 7 = (8)_{10} = (01000)_2$.

Representação de Números Reais no Computador

Sem olhar para a tabela, qual seria a configuração de bits para o expoente -7?

Maneira 1:

Vimos na tabela que: $E = D - D_{zero}$, onde:

- E: Valor do expoente. Neste exemplo, $E = -7$.
- D: Valor em decimal da representação em binário dos bits do expoente.
- D_{zero}: Valor em decimal da representação em binário dos bits do expoente zero. Aqui, $D_{zero} = (01111)_2 = 2^4 - 1 = (15)_{10}$.

Logo, $E = D - D_{zero} \Rightarrow -7 = D - 15 \Rightarrow D = (8)_{10} = (01000)_2$.

Portanto, 01000 é a configuração de bits para o expoente -7 neste exemplo.

Maneira 2:

$E = D - D_{zero} \Rightarrow D = D_{zero} + E$

- $D_{zero} = (01111)_2 = 2^4 - 1 = (15)_{10}$
- $E = -7$

Logo, $D = 15 - 7 = (2^4 - 1) - 7 = (8)_{10} = (01000)_2$.

Representação de Números Reais no Computador

Sem olhar para a tabela, qual seria a configuração de bits para o expoente -7?

Maneira 1:

Vimos na tabela que: $E = D - D_{zero}$, onde:

• E: Valor do expoente. Neste exemplo, $E = -7$.
• D: Valor em decimal da representação em binário dos bits do expoente.
• D_{zero}: Valor em decimal da representação em binário dos bits do expoente zero. Aqui, $D_{zero} = (01111)_2 = 2^4 - 1 = (15)_{10}$.

Logo, $E = D - D_{zero} \Rightarrow -7 = D - 15 \Rightarrow D = (8)_{10} = (01000)_2$.

Portanto, 01000 é a configuração de bits para o expoente -7 neste exemplo.

Maneira 2:

$E = D - D_{zero} \Rightarrow D = D_{zero} + E$
• $D_{zero} = (01111)_2 = 2^4 - 1 = (15)_{10}$
• $E = -7$
Logo, $D = 15 - 7 = (2^4 - 1) - 7 = (8)_{10} = (01000)_2$.

IEEE 754

A norma IEEE 754, publicada em 1985, procurou uniformizar a maneira como as diferentes máquinas representam os números em ponto flutuante, bem como devem operá-los.

Essa norma define alguns formatos básicos para os números em ponto flutuante na memória do computador, entre eles:

- Formato meia precisão (*FP16* ou *float16*): 16 bits - Primeiro bit para o sinal do número (0, positivo e 1, negativo), 5 bits para o expoente e 10 bits para a mantissa.
- Formato simples (*FP32* ou *float32*): 32 bits - Primeiro bit para o sinal do número (0, positivo e 1, negativo), 8 bits para o expoente e 23 bits para a mantissa.
- Formato duplo (*double*, *FP64* ou *float64*): 64 bits - Primeiro bit para o sinal do número (0, positivo e 1, negativo), 11 bits para o expoente e 52 bits para a mantissa.

IEEE 754

A norma IEEE 754, publicada em 1985, procurou uniformizar a maneira como as diferentes máquinas representam os números em ponto flutuante, bem como devem operá-los.

Essa norma define alguns formatos básicos para os números em ponto flutuante na memória do computador, entre eles:

- Formato meia precisão (*FP16* ou *float16*): 16 bits - Primeiro bit para o sinal do número (0, positivo e 1, negativo), 5 bits para o expoente e 10 bits para a mantissa.
- Formato simples (*FP32* ou *float32*): 32 bits - Primeiro bit para o sinal do número (0, positivo e 1, negativo), 8 bits para o expoente e 23 bits para a mantissa.
- Formato duplo (*double*, *FP64* ou *float64*): 64 bits - Primeiro bit para o sinal do número (0, positivo e 1, negativo), 11 bits para o expoente e 52 bits para a mantissa.

IEEE 754

A norma IEEE 754, publicada em 1985, procurou uniformizar a maneira como as diferentes máquinas representam os números em ponto flutuante, bem como devem operá-los.

Essa norma define alguns formatos básicos para os números em ponto flutuante na memória do computador, entre eles:

- Formato meia precisão (*FP16* ou *float16*): 16 bits - Primeiro bit para o sinal do número (0, positivo e 1, negativo), 5 bits para o expoente e 10 bits para a mantissa.
- Formato simples (*FP32* ou *float32*): 32 bits - Primeiro bit para o sinal do número (0, positivo e 1, negativo), 8 bits para o expoente e 23 bits para a mantissa.
- Formato duplo (*double*, *FP64* ou *float64*): 64 bits - Primeiro bit para o sinal do número (0, positivo e 1, negativo), 11 bits para o expoente e 52 bits para a mantissa.

IEEE 754

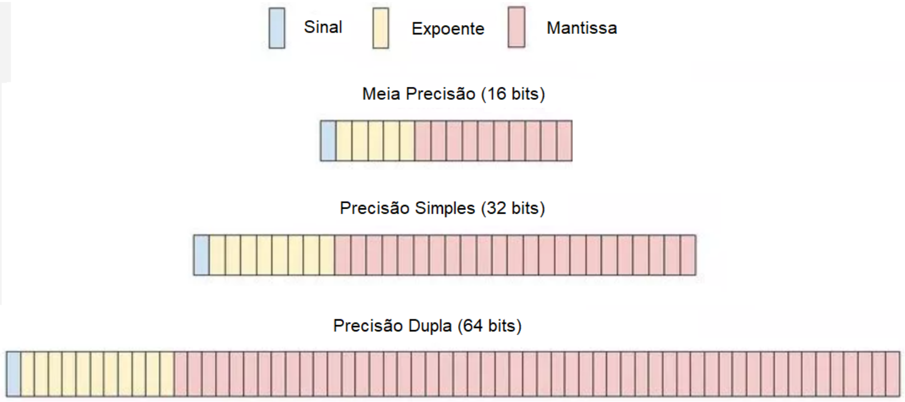

Figura: IEEE-754 - Formatos de 16, 32, 64 bits

IEEE 754

A norma IEEE 754 usa a notação *bias* (*viés*, em português).
Ou seja, o *valor em excesso*. É o valor em decimal para os bits do expoente zero (D_{zero}).

Em geral, $bias = 2^{n_{bits\_exp}-1} - 1$, onde n_{bits_exp} é o número de bits do expoente.

Alguns exemplos:

- Formato meia precisão (16 bits):
  $n_{bits\_exp} = 5 \Rightarrow bias = 2^{5-1} - 1 = 2^4 - 1 = 15$.
- Formato simples (32 bits): $n_{bits\_exp} = 8 \Rightarrow bias = 2^{8-1} - 1 = 2^7 - 1 = 127$.
- Formato duplo (64 bits):
  $n_{bits\_exp} = 11 \Rightarrow bias = 2^{11-1} - 1 = 2^{10} - 1 = 1023$.

Em geral, o expoente E é calculado subtraindo-se *bias (valor em excesso)* do valor em decimal dos n_{bits_exp} bits reservados para o expoente. Ou seja, $E = D - bias$, onde $bias = D_{zero}$.

IEEE 754

A norma IEEE 754 usa a notação *bias* (*viés*, em português).
Ou seja, o *valor em excesso*. É o valor em decimal para os bits do expoente zero (D_{zero}).

Em geral, $bias = 2^{n_{bits\_exp}-1} - 1$, onde n_{bits_exp} é o número de bits do expoente.

Alguns exemplos:

- Formato meia precisão (16 bits):
  $n_{bits\_exp} = 5 \Rightarrow bias = 2^{5-1} - 1 = 2^4 - 1 = 15$.
- Formato simples (32 bits): $n_{bits\_exp} = 8 \Rightarrow bias = 2^{8-1} - 1 = 2^7 - 1 = 127$.
- Formato duplo (64 bits):
  $n_{bits\_exp} = 11 \Rightarrow bias = 2^{11-1} - 1 = 2^{10} - 1 = 1023$.

Em geral, o expoente E é calculado subtraindo-se *bias (valor em excesso)* do valor em decimal dos n_{bits_exp} bits reservados para o expoente. Ou seja, $E = D - bias$, onde $bias = D_{zero}$.

IEEE 754

A norma IEEE 754 usa a notação *bias* (*viés*, em português).
Ou seja, o *valor em excesso*. É o valor em decimal para os bits do expoente zero (D_{zero}).

Em geral, $bias = 2^{n_{bits\_exp}-1} - 1$, onde n_{bits_exp} é o número de bits do expoente.

Alguns exemplos:

- Formato meia precisão (16 bits):
  $n_{bits\_exp} = 5 \Rightarrow bias = 2^{5-1} - 1 = 2^4 - 1 = 15$.
- Formato simples (32 bits): $n_{bits\_exp} = 8 \Rightarrow bias = 2^{8-1} - 1 = 2^7 - 1 = 127$.
- Formato duplo (64 bits):
  $n_{bits\_exp} = 11 \Rightarrow bias = 2^{11-1} - 1 = 2^{10} - 1 = 1023$.

Em geral, o expoente E é calculado subtraindo-se *bias (valor em excesso)* do valor em decimal dos n_{bits_exp} bits reservados para o expoente. Ou seja, $E = D - bias$, onde $bias = D_{zero}$.

IEEE 754

A norma IEEE 754 usa a notação *bias* (*viés*, em português).
Ou seja, o *valor em excesso*. É o valor em decimal para os bits do expoente zero (D_{zero}).

Em geral, $bias = 2^{n_{bits\_exp}-1} - 1$, onde n_{bits_exp} é o número de bits do expoente.

Alguns exemplos:

- Formato meia precisão (16 bits):
  $n_{bits\_exp} = 5 \Rightarrow bias = 2^{5-1} - 1 = 2^4 - 1 = 15$.
- Formato simples (32 bits): $n_{bits\_exp} = 8 \Rightarrow bias = 2^{8-1} - 1 = 2^7 - 1 = 127$.
- Formato duplo (64 bits):
  $n_{bits\_exp} = 11 \Rightarrow bias = 2^{11-1} - 1 = 2^{10} - 1 = 1023$.

Em geral, o expoente E é calculado subtraindo-se *bias (valor em excesso)* do valor em decimal dos n_{bits_exp} bits reservados para o expoente. Ou seja, $E = D - bias$, onde $bias = D_{zero}$.

IEEE 754

Assim, a forma normalizada pode ser definida como:

$$(-1)^s \cdot m \cdot 2^E = (-1)^s \cdot m \cdot 2^{D-bias},$$

onde:

- *bit_s*: bit do sinal do número (0, positivo e 1, negativo);
- m: mantissa, dada por $1, b_1 b_2 b_3 \ldots b_{bits\_mant}$, onde b_i são os bits da mantissa e *bits_mant* é o número de bits reservados para a mantissa;
- E: expoente, defnido como $D - bias$, onde D é o valor em decimal dos bits do expoente e *bias* é o *valor em excesso* (valor em decimal do expoente zero).

Portanto, a forma normalizada pode ser definida como:

$$(-1)^s \cdot 1, b_1 b_2 b_3 \ldots b_{bits\_mant} \cdot 2^{D-bias}$$

Erro relativo máximo de um número em ponto flutuante

Vamos admitir, como exemplo, que o número a seja na forma normalizada:

$$a = 1,101010110100101 \cdots \times 2^{c}$$

e que $m = 5$ seja o número de bits da mantissa do computador.

Seja $\overline{a}$ a aproximação para a neste computador.

Então,

$\overline{a} = 1,10101 \times 2^{c}$, pois houve truncamento (como após o último dígito da mantissa, aparece 0, então descarto os dígitos que vêm depois da mantissa).

No truncamento, $\overline{a} < a$.

O erro absoluto (E_{abs}) entre o valor verdadeiro a e o valor aproximado pela máquina $\overline{a}$ é dado por:

$$E_{\text{abs}} = a - \overline{a}.$$

Erro relativo máximo de um número em ponto flutuante

Vamos admitir, como exemplo, que o número a seja na forma normalizada:

$$a = 1,101010110100101 \cdots \times 2^c$$

e que $m = 5$ seja o número de bits da mantissa do computador.

Seja $\overline{a}$ a aproximação para a neste computador.

Então,

$\overline{a} = 1,10101 \times 2^c$, pois houve truncamento (como após o último dígito da mantissa, aparece 0, então descarto os dígitos que vêm depois da mantissa).

No truncamento, $\overline{a} < a$.

O erro absoluto (E_{abs}) entre o valor verdadeiro a e o valor aproximado pela máquina $\overline{a}$ é dado por:

$$E_{\text{abs}} = a - \overline{a}.$$

Erro relativo máximo de um número em ponto flutuante

Vamos admitir, como exemplo, que o número a seja na forma normalizada:

$a = 1,101010110100101 \cdots \times 2^c$

e que $m = 5$ seja o número de bits da mantissa do computador.

Seja $\overline{a}$ a aproximação para a neste computador.

Então,

$\overline{a} = 1,10101 \times 2^c$, pois houve truncamento (como após o último dígito da mantissa, aparece 0, então descarto os dígitos que vêm depois da mantissa).

No truncamento, $\overline{a} < a$.

O erro absoluto (E_{abs}) entre o valor verdadeiro a e o valor aproximado pela máquina $\overline{a}$ é dado por:

$E_{\text{abs}} = a - \overline{a}$.

Erro relativo máximo de um número em ponto flutuante

Vamos admitir, como exemplo, que o número a seja na forma normalizada:

$a = 1,101010110100101 \cdots \times 2^c$

e que $m = 5$ seja o número de bits da mantissa do computador.

Seja $\overline{a}$ a aproximação para a neste computador.

Então,

$\overline{a} = 1,10101 \times 2^c$, pois houve truncamento (como após o último dígito da mantissa, aparece 0, então descarto os dígitos que vêm depois da mantissa).

No truncamento, $\overline{a} < a$.

O erro absoluto (E_{abs}) entre o valor verdadeiro a e o valor aproximado pela máquina $\overline{a}$ é dado por:

$E_{\text{abs}} = a - \overline{a}$.

Erro relativo máximo de um número em ponto flutuante

Vamos admitir, como exemplo, que o número a seja na forma normalizada:

$a = 1,101010110100101 \cdots \times 2^c$

e que $m = 5$ seja o número de bits da mantissa do computador.

Seja $\overline{a}$ a aproximação para a neste computador.

Então,

$\overline{a} = 1,10101 \times 2^c$, pois houve truncamento (como após o último dígito da mantissa, aparece 0, então descarto os dígitos que vêm depois da mantissa).

No truncamento, $\overline{a} < a$.

O erro absoluto (E_{abs}) entre o valor verdadeiro a e o valor aproximado pela máquina $\overline{a}$ é dado por:

$E_{\text{abs}} = a - \overline{a}$.

Erro relativo máximo de um número em ponto flutuante

Ou seja, vamos fazer a seguinte operação de subtração (a parte destacada em vermelho está fora da mantissa):

$$
\begin{array}{rll}
 & a & = 1,10101\textcolor{red}{0110100101}\ldots \times 2^{c} \\
- & \bar{a} & = 1,10101\textcolor{red}{0000000000}\ldots \times 2^{c} \\
\hline
 & E_{\text{abs}} & = 0,00000\textcolor{red}{0110100101}\ldots \times 2^{c}
\end{array}
$$

Então, o erro absoluto é dado por:

$$
\begin{aligned}
E_{\text{abs}} &= 0,00000\textcolor{red}{0110100101}\cdots \times 2^{c} = 0,\textcolor{red}{0110100101}\cdots \times 2^{-5} \times 2^{c} \\
&= 0,\textcolor{red}{0110100101}\cdots \times 2^{c-5}
\end{aligned}
$$

Já o erro relativo (E_{rel}) é dado por:

$$
\begin{aligned}
E_{\text{rel}} = \frac{E_{\text{abs}}}{a} &= \frac{0,\textcolor{red}{0110100101}\cdots \times 2^{c-5}}{1,10101\textcolor{red}{0110100101}\ldots \times 2^{c}} \\
&= \frac{0,\textcolor{red}{0110100101}\ldots}{1,10101\textcolor{red}{0110100101}\ldots} \times 2^{-5}
\end{aligned}
$$

Erro relativo máximo de um número em ponto flutuante

Ou seja, vamos fazer a seguinte operação de subtração (a parte destacada em vermelho está fora da mantissa):

$$\begin{array}{rll} & a & = 1,10101\color{red}{0110100101}\ldots \times 2^{c} \\ - & \bar{a} & = 1,10101\color{red}{0000000000}\ldots \times 2^{c} \\ \hline & E_{\text{abs}} & = 0,00000\color{red}{0110100101}\ldots \times 2^{c} \end{array}$$

Então, o erro absoluto é dado por:

$$E_{\text{abs}} = 0,00000\color{red}{0110100101}\cdots \times 2^{c} = 0,\color{red}{0110100101}\cdots \times 2^{-5} \times 2^{c}$$

$$= 0,\color{red}{0110100101}\cdots \times 2^{c-5}$$

Já o erro relativo (E_{rel}) é dado por:

$$E_{\text{rel}} = \frac{E_{\text{abs}}}{a} = \frac{0,\color{red}{0110100101}\cdots \times 2^{c-5}}{1,10101\color{red}{0110100101}\ldots \times 2^{c}}$$

$$= \frac{0,\color{red}{0110100101}\ldots}{1,10101\color{red}{0110100101}\ldots} \times 2^{-5}$$

Erro relativo máximo de um número em ponto flutuante

Ou seja, vamos fazer a seguinte operação de subtração (a parte destacada em vermelho está fora da mantissa):

$$
\begin{array}{rll}
 & a & = 1,10101\color{red}{0110100101}\ldots \times 2^c \\
- & \underline{\bar{a}} & \underline{= 1,10101\color{red}{0000000000}\ldots \times 2^c} \\
 & E_{\text{abs}} & = 0,00000\color{red}{0110100101}\ldots \times 2^c
\end{array}
$$

Então, o erro absoluto é dado por:

$$
\begin{aligned}
E_{\text{abs}} &= 0,00000\color{red}{0110100101}\cdots \times 2^c = 0,\color{red}{0110100101}\cdots \times 2^{-5} \times 2^c \\
&= 0,\color{red}{0110100101}\cdots \times 2^{c-5}
\end{aligned}
$$

Já o erro relativo (E_{rel}) é dado por:

$$
\begin{aligned}
E_{\text{rel}} = \frac{E_{\text{abs}}}{a} &= \frac{0,\color{red}{0110100101}\cdots \times 2^{c-5}}{1,10101\color{red}{0110100101}\ldots \times 2^c} \\
&= \frac{0,\color{red}{0110100101}\ldots}{1,10101\color{red}{0110100101}\ldots} \times 2^{-5}
\end{aligned}
$$

Erro relativo máximo de um número em ponto flutuante

Colocando o numerador e o denominador do erro relativo na forma normalizada, temos:

$$E_{\text{rel}} = \frac{0,0110100101\ldots}{1,101010110100101\ldots} \times 2^{-5} = \frac{1,10100101\cdots \times 2^{-2}}{1,101010110100101\ldots} \times 2^{-5}$$

Portanto,

$$E_{\text{rel}} = \frac{1,10100101\ldots}{1,101010110100101\ldots} \times 2^{-2} \times 2^{-5}$$

Diante deste resultado, podemos deduzir que o erro relativo máximo é dado por:

$$E_{\text{rel}} = \frac{\text{maior numerador normalizado}}{\text{menor denominador normalizado}} \times 2^{-2} \times 2^{-5}$$

Logo,

$$(E_{\text{rel}})_{\text{MAX}} = \frac{1,11111111\ldots}{1,000000000000000\ldots} \times 2^{-2} \times 2^{-5} = 1,11111111\ldots \times 2^{-2} \times 2^{-5}$$

Erro relativo máximo de um número em ponto flutuante

Colocando o numerador e o denominador do erro relativo na forma normalizada, temos:

$$E_{\text{rel}} = \frac{0,0110100101\ldots}{1,101010110100101\ldots} \times 2^{-5} = \frac{1,10100101\cdots \times 2^{-2}}{1,101010110100101\ldots} \times 2^{-5}$$

Portanto,

$$E_{\text{rel}} = \frac{1,10100101\ldots}{1,101010110100101\ldots} \times 2^{-2} \times 2^{-5}$$

Diante deste resultado, podemos deduzir que o erro relativo máximo é dado por:

$$E_{\text{rel}} = \frac{\text{maior numerador normalizado}}{\text{menor denominador normalizado}} \times 2^{-2} \times 2^{-5}$$

Logo,

$$(E_{\text{rel}})_{\text{MAX}} = \frac{1,11111111\ldots}{1,000000000000000\ldots} \times 2^{-2} \times 2^{-5} = 1,11111111\ldots \times 2^{-2} \times 2^{-5}$$

Erro relativo máximo de um número em ponto flutuante

Colocando o numerador e o denominador do erro relativo na forma normalizada, temos:

$$E_{\text{rel}} = \frac{0,0110100101\ldots}{1,10101\,0110100101\ldots} \times 2^{-5} = \frac{1,10100101\cdots \times 2^{-2}}{1,10101\,0110100101\ldots} \times 2^{-5}$$

Portanto,

$$E_{\text{rel}} = \frac{1,10100101\ldots}{1,10101\,0110100101\ldots} \times 2^{-2} \times 2^{-5}$$

Diante deste resultado, podemos deduzir que o erro relativo máximo é dado por:

$$E_{\text{rel}} = \frac{\text{maior numerador normalizado}}{\text{menor denominador normalizado}} \times 2^{-2} \times 2^{-5}$$

Logo,

$$(E_{\text{rel}})_{\text{MAX}} = \frac{1,11111111\ldots}{1,00000\,0000000000\ldots} \times 2^{-2} \times 2^{-5} = 1,11111111\ldots \times 2^{-2} \times 2^{-5}$$

Erro relativo máximo de um número em ponto flutuante

Colocando o numerador e o denominador do erro relativo na forma normalizada, temos:

$$E_{\text{rel}} = \frac{0,0110100101\ldots}{1,10101011010101\ldots} \times 2^{-5} = \frac{1,10100101\cdots \times 2^{-2}}{1,10101011010101\ldots} \times 2^{-5}$$

Portanto,

$$E_{\text{rel}} = \frac{1,10100101\ldots}{1,10101011010101\ldots} \times 2^{-2} \times 2^{-5}$$

Diante deste resultado, podemos deduzir que o erro relativo máximo é dado por:

$$E_{\text{rel}} = \frac{\text{maior numerador normalizado}}{\text{menor denominador normalizado}} \times 2^{-2} \times 2^{-5}$$

Logo,

$$(E_{\text{rel}})_{\text{MAX}} = \frac{1,11111111\ldots}{1,000000000000000\ldots} \times 2^{-2} \times 2^{-5} = 1,11111111\ldots \times 2^{-2} \times 2^{-5}$$

Erro relativo máximo de um número em ponto flutuante

Ou seja,

$$(E_{\text{rel}})_{\text{MAX}} = 1,11111111\ldots \times 2^{-2} \times 2^{-5} = 0,0111111111\ldots \times 2^{-5}$$

Como $0,0111111111\ldots < 0,1000000000\ldots = 2^{-1}$, então

$$(E_{\text{rel}})_{\text{MAX}} < 2^{-1} \times 2^{-5} = 2^{-6} = \frac{1}{2^6} = \frac{1}{64} = 0,015625.$$

Lembre-se de neste exemplo, o número de bits da mantissa é $m = 5$.

Logo, podemos deduzir que num computador com m bits o erro relativo máximo de um número será dado por:

$$(E_{\text{rel}})_{\text{MAX}} < 2^{-1} \times 2^{-m} = 2^{-(m+1)}$$

Esta expressão para o erro máximo só é válida na região normalizada.

Erro relativo máximo de um número em ponto flutuante

Ou seja,

$$(E_{\text{rel}})_{\text{MAX}} = 1,11111111\ldots \times 2^{-2} \times 2^{-5} = 0,0111111111\ldots \times 2^{-5}$$

Como $0,0111111111\ldots < 0,1000000000\ldots = 2^{-1}$, então

$$(E_{\text{rel}})_{\text{MAX}} < 2^{-1} \times 2^{-5} = 2^{-6} = \frac{1}{2^6} = \frac{1}{64} = 0,015625.$$

Lembre-se de neste exemplo, o número de bits da mantissa é $m = 5$.

Logo, podemos deduzir que num computador com m bits o erro relativo máximo de um número será dado por:

$$(E_{\text{rel}})_{\text{MAX}} < 2^{-1} \times 2^{-m} = 2^{-(m+1)}$$

Esta expressão para o erro máximo só é válida na região normalizada.

Erro relativo máximo de um número em ponto flutuante

Ou seja,

$$(E_{\text{rel}})_{\text{MAX}} = 1,11111111\ldots \times 2^{-2} \times 2^{-5} = 0,0111111111\ldots \times 2^{-5}$$

Como $0,0111111111\ldots < 0,1000000000\ldots = 2^{-1}$, então

$$(E_{\text{rel}})_{\text{MAX}} < 2^{-1} \times 2^{-5} = 2^{-6} = \frac{1}{2^6} = \frac{1}{64} = 0,015625.$$

Lembre-se de neste exemplo, o número de bits da mantissa é $m = 5$.

Logo, podemos deduzir que num computador com m bits o erro relativo máximo de um número será dado por:

$$(E_{\text{rel}})_{\text{MAX}} < 2^{-1} \times 2^{-m} = 2^{-(m+1)}$$

Esta expressão para o erro máximo só é válida na região normalizada.

Erro relativo máximo de um número em ponto flutuante

Ou seja,

$$(E_{\text{rel}})_{\text{MAX}} = 1,11111111\ldots \times 2^{-2} \times 2^{-5} = 0,0111111111\ldots \times 2^{-5}$$

Como $0,0111111111\ldots < 0,1000000000\ldots = 2^{-1}$, então

$$(E_{\text{rel}})_{\text{MAX}} < 2^{-1} \times 2^{-5} = 2^{-6} = \frac{1}{2^6} = \frac{1}{64} = 0,015625.$$

Lembre-se de neste exemplo, o número de bits da mantissa é $m = 5$.

Logo, podemos deduzir que num computador com m bits o erro relativo máximo de um número será dado por:

$$(E_{\text{rel}})_{\text{MAX}} < 2^{-1} \times 2^{-m} = 2^{-(m+1)}$$

Esta expressão para o erro máximo só é válida na região normalizada.

Erro relativo máximo de um número em ponto flutuante

Ou seja,

$$(E_{\text{rel}})_{\text{MAX}} = 1,11111111\ldots \times 2^{-2} \times 2^{-5} = 0,0111111111\ldots \times 2^{-5}$$

Como $0,0111111111\ldots < 0,1000000000\ldots = 2^{-1}$, então

$$(E_{\text{rel}})_{\text{MAX}} < 2^{-1} \times 2^{-5} = 2^{-6} = \frac{1}{2^6} = \frac{1}{64} = 0,015625.$$

Lembre-se de neste exemplo, o número de bits da mantissa é $m = 5$.

Logo, podemos deduzir que num computador com m bits o erro relativo máximo de um número será dado por:

$$(E_{\text{rel}})_{\text{MAX}} < 2^{-1} \times 2^{-m} = 2^{-(m+1)}$$

Esta expressão para o erro máximo só é válida na região normalizada.

Valor verdadeiro do número armazenado

Exemplo: Armazenar $0,8$ em um computador que reserva 23 bits para a mantissa (precisão simples).

Vamos converter $0,8$ para binário. Como é uma fração, devemos usar multiplicações sucessivas e usar ordem direta das partes inteiras como os dígitos em binário.

$$0,8 \times 2 = 1,6$$
$$0,6 \times 2 = 1,2$$
$$0,2 \times 2 = 0,4$$
$$0,4 \times 2 = 0,8$$
$$0,8 \times 2 = 1,6$$
$$0,6 \times 2 = 1,2$$
$$0,2 \times 2 = 0,4$$
$$0,4 \times 2 = 0,8$$
$$\ldots \qquad \Rightarrow (0,8)_{10} = (0,110011001100110011001100\ldots)_2 = (0,\overline{1100})_2$$

Valor verdadeiro do número armazenado

Exemplo: Armazenar $0,8$ em um computador que reserva 23 bits para a mantissa (precisão simples).

Vamos converter $0,8$ para binário. Como é uma fração, devemos usar multiplicações sucessivas e usar ordem direta das partes inteiras como os dígitos em binário.

$$0,8 \times 2 = 1,6$$
$$0,6 \times 2 = 1,2$$
$$0,2 \times 2 = 0,4$$
$$0,4 \times 2 = 0,8$$
$$0,8 \times 2 = 1,6$$
$$0,6 \times 2 = 1,2$$
$$0,2 \times 2 = 0,4$$
$$0,4 \times 2 = 0,8$$
$$\ldots \qquad \Rightarrow (0,8)_{10} = (0,110011001100110011001100\ldots)_2 = (0,\overline{1100})_2$$

Valor verdadeiro do número armazenado

Exemplo: Armazenar $0,8$ em um computador que reserva 23 bits para a mantissa (precisão simples).

Vamos converter $0,8$ para binário. Como é uma fração, devemos usar multiplicações sucessivas e usar ordem direta das partes inteiras como os dígitos em binário.

$0,8 \times 2 = 1,6$

$0,6 \times 2 = 1,2$

$0,2 \times 2 = 0,4$

$0,4 \times 2 = 0,8$

$0,8 \times 2 = 1,6$

$0,6 \times 2 = 1,2$

$0,2 \times 2 = 0,4$

$0,4 \times 2 = 0,8$

$\ldots \qquad \Rightarrow (0,8)_{10} = (0,11001100110011001100\ldots)_2 = (0,\overline{1100})_2$

Valor verdadeiro do número armazenado

Exemplo: Armazenar $0,8$ em um computador que reserva 23 bits para a mantissa (precisão simples).

Vamos converter $0,8$ para binário. Como é uma fração, devemos usar multiplicações sucessivas e usar ordem direta das partes inteiras como os dígitos em binário.

$$0,8 \times 2 = 1,6$$
$$0,6 \times 2 = 1,2$$
$$0,2 \times 2 = 0,4$$
$$0,4 \times 2 = 0,8$$
$$0,8 \times 2 = 1,6$$
$$0,6 \times 2 = 1,2$$
$$0,2 \times 2 = 0,4$$
$$0,4 \times 2 = 0,8$$
$$\ldots \quad \Rightarrow (0,8)_{10} = (0,11001100110011001100\ldots)_2 = (0,\overline{1100})_2$$

Valor verdadeiro do número armazenado

Qual será o valor verdadeiro armazenado neste computador?

Na forma normalizada, temos:

$$(0,8)_{10} = 1,\underbrace{10011001100110011001100}_{\text{23 dígitos}}11001100\ldots \times 2^{-1}$$

$$\approx 1,10011001100110011001101 \times 2^{-1}$$

Logo, houve arredondamento e o valor verdadeiro A do número $0,8$ representado no computador é

$$A = (1,10011001100110011001101)_2 \times 2^{-1}$$

Como houve arredondamento, $A > (0,8)_{10}$.

Valor verdadeiro do número armazenado

Qual será o valor verdadeiro armazenado neste computador?

Na forma normalizada, temos:

$$(0,8)_{10} = 1,\underbrace{10011001100110011001100}_{\text{23 dígitos}}11001100\ldots \times 2^{-1}$$

$$\approx 1,10011001100110011001101 \times 2^{-1}$$

Logo, houve arredondamento e o valor verdadeiro A do número $0,8$ representado no computador é

$$A = (1,10011001100110011001101)_2 \times 2^{-1}$$

Como houve arredondamento, $A > (0,8)_{10}$.

Valor verdadeiro do número armazenado

Qual será o valor verdadeiro armazenado neste computador?

Na forma normalizada, temos:

$$(0,8)_{10} = 1,\underbrace{10011001100110011001100}_{\text{23 dígitos}}11001100\ldots \times 2^{-1}$$

$$\approx 1,10011001100110011001101 \times 2^{-1}$$

Logo, houve arredondamento e o valor verdadeiro A do número $0,8$ representado no computador é

$$A = (1,10011001100110011001101)_2 \times 2^{-1}$$

Como houve arredondamento, $A > (0,8)_{10}$.

Valor verdadeiro do número armazenado

Qual será o valor verdadeiro armazenado neste computador?

Na forma normalizada, temos:

$$(0,8)_{10} = 1,\underbrace{10011001100110011001100}_{\text{23 dígitos}}\,11001100\ldots \times 2^{-1}$$

$$\approx 1,10011001100110011001101 \times 2^{-1}$$

Logo, houve arredondamento e o valor verdadeiro A do número $0,8$ representado no computador é

$$A = (1,10011001100110011001101)_2 \times 2^{-1}$$

Como houve arredondamento, $A > (0,8)_{10}$.

Valor verdadeiro do número armazenado

Então, subtraindo $0,8$ de A, temos:

$$
\begin{array}{rll}
 & A & = 1,1001100110011001100110\,1 \qquad\qquad \times 2^{-1} \\
- & \underline{(0,8)_{10}} & \underline{= 1,1001100110011001100110\,011001100\ldots \times 2^{-1}}
\end{array}
$$

Ao retirar os dígitos mais à esquerda de A e $(0,8)_{10}$ que são iguais, fazer essa conta é equivalente a fazer a conta a seguir:

$$
\begin{array}{rl}
 & 0,0000000000000000000000\,1 \qquad\qquad \times 2^{-1} \\
- & \underline{0,0000000000000000000000\,011001100\ldots \times 2^{-1}}
\end{array}
$$

Valor verdadeiro do número armazenado

Então, subtraindo $0,8$ de A, temos:

$$
\begin{array}{rll}
 & A & = 1,{\color{red}1001100110011001100110}1 \qquad\qquad\qquad \times 2^{-1} \\
- & \underline{(0,8)_{10}} & \underline{= 1,{\color{red}1001100110011001100110}011001100\ldots \times 2^{-1}}
\end{array}
$$

Ao retirar os dígitos mais à esquerda de A e $(0,8)_{10}$ que são iguais, fazer essa conta é equivalente a fazer a conta a seguir:

$$
\begin{array}{rl}
 & 0,{\color{red}0000000000000000000000}1 \qquad\qquad\qquad \times 2^{-1} \\
- & \underline{0,{\color{red}0000000000000000000000}011001100\ldots \times 2^{-1}}
\end{array}
$$

Valor verdadeiro do número armazenado

Ou seja,

$$
\begin{array}{rl}
 & 1,00000000\ldots \times 2^{-23} \times 2^{-1} \\
- & \underline{0,11001100\ldots \times 2^{-23} \times 2^{-1}}
\end{array}
$$

Assim,

$$
\begin{aligned}
A - (0,8)_{10} &= 1 \times 2^{-23} \times 2^{-1} - 0,11001100\cdots \times 2^{-23} \times 2^{-1} \\
&= (1 - 0,11001100\ldots) \times 2^{-23} \times 2^{-1}
\end{aligned}
$$

Lembre-se de que vimos lá no início que $(0,8)_{10} = (0,11001100\ldots)_2$. Então,

$$
\begin{aligned}
A - (0,8)_{10} &= (1 - 0,11001100\ldots) \times 2^{-23} \times 2^{-1} \\
&= (1 - (0,8)_{10}) \times 2^{-24} = (0,2)_{10} \times 2^{-24} = (1,19209\ldots)_{10} \times 10^{-8}.
\end{aligned}
$$

Valor verdadeiro do número armazenado

Ou seja,

$$\begin{array}{rl} & 1,00000000\ldots \times 2^{-23} \times 2^{-1} \\ - & \underline{0,11001100\ldots \times 2^{-23} \times 2^{-1}} \end{array}$$

Assim,

$$\begin{aligned} A - (0,8)_{10} &= 1 \times 2^{-23} \times 2^{-1} - 0,11001100\cdots \times 2^{-23} \times 2^{-1} \\ &= (1 - 0,11001100\ldots) \times 2^{-23} \times 2^{-1} \end{aligned}$$

Lembre-se de que vimos lá no início que $(0,8)_{10} = (0,11001100\ldots)_2$. Então,

$$\begin{aligned} A - (0,8)_{10} &= (1 - 0,11001100\ldots) \times 2^{-23} \times 2^{-1} \\ &= (1 - (0,8)_{10}) \times 2^{-24} = (0,2)_{10} \times 2^{-24} = (1,19209\ldots)_{10} \times 10^{-8}. \end{aligned}$$

Valor verdadeiro do número armazenado

Ou seja,

$$
\begin{array}{rl}
& 1,00000000\ldots \times 2^{-23} \times 2^{-1} \\
- & \underline{0,11001100\ldots \times 2^{-23} \times 2^{-1}}
\end{array}
$$

Assim,

$$
\begin{aligned}
A - (0,8)_{10} &= 1 \times 2^{-23} \times 2^{-1} - 0,11001100\cdots \times 2^{-23} \times 2^{-1} \\
&= (1 - 0,11001100\ldots) \times 2^{-23} \times 2^{-1}
\end{aligned}
$$

Lembre-se de que vimos lá no início que $(0,8)_{10} = (0,11001100\ldots)_2$.

Então,

$$
\begin{aligned}
A - (0,8)_{10} &= (1 - 0,11001100\ldots) \times 2^{-23} \times 2^{-1} \\
&= (1 - (0,8)_{10}) \times 2^{-24} = (0,2)_{10} \times 2^{-24} = (1,19209\ldots)_{10} \times 10^{-8}.
\end{aligned}
$$

Valor verdadeiro do número armazenado

Ou seja,

$$
\begin{array}{rl}
 & 1,00000000\ldots \times 2^{-23} \times 2^{-1} \\
- & \underline{0,11001100\ldots \times 2^{-23} \times 2^{-1}}
\end{array}
$$

Assim,

$$
\begin{aligned}
A - (0,8)_{10} &= 1 \times 2^{-23} \times 2^{-1} - 0,11001100\cdots \times 2^{-23} \times 2^{-1} \\
&= (1 - 0,11001100\ldots) \times 2^{-23} \times 2^{-1}
\end{aligned}
$$

Lembre-se de que vimos lá no início que $(0,8)_{10} = (0,11001100\ldots)_2$.
Então,

$$
\begin{aligned}
A - (0,8)_{10} &= (1 - 0,11001100\ldots) \times 2^{-23} \times 2^{-1} \\
&= (1 - (0,8)_{10}) \times 2^{-24} = (0,2)_{10} \times 2^{-24} = (1,19209\ldots)_{10} \times 10^{-8}.
\end{aligned}
$$

Valor verdadeiro do número armazenado

Logo, o valor verdadeiro armazenado neste computador para representar $(0,8)_{10}$ é

$$A = (0,8)_{10} + (1,19209\ldots)_{10} \times 10^{-8} = 0,8000000119209\ldots$$

Ou seja, um valor ligeiramente maior que $0,8$.

Exercício: Analogamente ao que foi feito no exemplo anterior, ache o valor verdadeiro de $(1,8)_{10}$ armazenado neste computador.

Valor verdadeiro do número armazenado

Logo, o valor verdadeiro armazenado neste computador para representar $(0,8)_{10}$ é

$$A = (0,8)_{10} + (1,19209\ldots)_{10} \times 10^{-8} = 0,8000000119209\ldots$$

Ou seja, um valor ligeiramente maior que $0,8$.

Exercício: Analogamente ao que foi feito no exemplo anterior, ache o valor verdadeiro de $(1,8)_{10}$ armazenado neste computador.

Valor verdadeiro do número armazenado

Logo, o valor verdadeiro armazenado neste computador para representar $(0,8)_{10}$ é

$A = (0,8)_{10} + (1,19209\ldots)_{10} \times 10^{-8} = 0,8000000119209\ldots$

Ou seja, um valor ligeiramente maior que $0,8$.

Exercício: Analogamente ao que foi feito no exemplo anterior, ache o valor verdadeiro de $(1,8)_{10}$ armazenado neste computador.

- Oliveira, R. **Capítulo 2 - Representação binária de números inteiros e reais** - Site: www.raymundodeoliveira.eng.br/binario.html
- **Representação de ponto flutuante IEEE** - Site: https://learn.microsoft.com/pt-br/cpp/build/ieee-floating-point-representation?view=msvc-170