Representação de números em máquinas

Leonardo F. Guidi

DMPA – IME
UFRGS

Cálculo Numérico

Índice

Sistema de numeração posicional

Sistema de numeração posicional

Um sistema de numeração é formado por uma coleção de símbolos e regras para representar conjuntos de números de maneira consistente. Um sistema de numeração que desempenhe satisfatoriamente o seu propósito deve possuir as seguintes propriedades:

1. Capacidade de representar um conjunto de números distintos de maneira padronizada.
2. Deve refletir as estruturas algébricas e aritméticas dos números

O sistema mais utilizado é o *sistema de numeração posicional de base* 10 ou *sistema de numeração base*-10 – o sistema decimal.
Em um sistema de numeração posicional a posição relativa dos símbolos guarda informação sobre o número que se quer representar. Mais especificamente, posições adjacentes estão relacionadas entre si por uma constante multiplicativa denominada *base* do sistema de numeração.

Sistema de numeração posicional

Sistemas de numeração base-b

Dado um natural $b > 1$, a coleção de símbolos "$-$", "," e os algarismos $\{0, 1, \ldots, b-1\}$, o numeral

$$(d_n d_{n-1} \cdots d_1 d_0, d_{-1} \cdots)_b \tag{1}$$

representa o número real positivo

$$\sum_{j \leq n} d_j b^j. \tag{2}$$

Aplicando o sinal "$-$" à frente do numeral representamos os reais negativos.

Sistema de numeração posicional

Observações

- Quando não há possibilidade de equívoco, dispensamos o subscrito 10 na representação decimal.
- Os algarismos 0 antes da vírgula também não são representados se não houver mais algum outro algarismo à esquerda.
- O mesmo ocorre para os algarismos zero à direita após a vírgula com exceção do uso nas ciências exatas quando faz-se necessário registrar a precisão de alguma medida. Assim, evitamos notações da forma $00\ldots02{,}3$ para indicar o racional $\frac{23}{10}$; o mesmo para notações da forma $2{,}30\ldots00$, com exceção dos casos em que se quer indicar não o racional $\frac{23}{10}$ mas uma medida com determinada precisão
- Quando a representação de um número possuir dízimas periódicas utiliza-se como notação uma barra sobre a sequência que se repete. Por exemplo, $\frac{1}{3}$ é representado como $0{,}\overline{3}$ e $\frac{67}{495}$ é representado como $0{,}1\overline{35}$.

Mudança de base

Seja, então, X um número representado na base b como $d_n d_{n-1} \cdots d_1 d_0, d_{-1} \ldots {}_b$. Encontrar a sua representação em uma outra base g significa encontrar a sequência de algarismos $\tilde{d}_m \tilde{d}_{m-1} \cdots \tilde{d}_1 \tilde{d}_0, \tilde{d}_{-1} \ldots {}_g$ tal que

$$X = \sum_{j \leq m} \tilde{d}_j g^j.$$

Mudança de base

Seja, então, X um número representado na base b como $d_n d_{n-1} \cdots d_1 d_0, d_{-1} \ldots {}_b$. Encontrar a sua representação em uma outra base g significa encontrar a sequência de algarismos $\tilde{d}_m \tilde{d}_{m-1} \cdots \tilde{d}_1 \tilde{d}_0, \tilde{d}_{-1} \ldots {}_g$ tal que

$$X = \sum_{j \leq m} \tilde{d}_j g^j.$$

1º passo: separar X nas suas as partes inteira $X^i = \sum_{j=0}^{m} \tilde{d}_j g^j$, e fracionária, $X^f = \sum_{j \leq -1} \tilde{d}_j g^j$. Elas serão tratadas separadamente.

Mudança de base

Seja, então, X um número representado na base b como $d_n d_{n-1} \cdots d_1 d_0, d_{-1} \ldots {}_b$. Encontrar a sua representação em uma outra base g significa encontrar a sequência de algarismos $\tilde{d}_m \tilde{d}_{m-1} \cdots \tilde{d}_1 \tilde{d}_0, \tilde{d}_{-1} \ldots {}_g$ tal que

$$X = \sum_{j \leq m} \tilde{d}_j g^j.$$

1º passo: separar X nas suas as partes inteira $X^i = \sum_{j=0}^{m} \tilde{d}_j g^j$, e fracionária, $X^f = \sum_{j \leq -1} \tilde{d}_j g^j$. Elas serão tratadas separadamente.
2º passo: vamos tratar inicialmente a parte inteira. Vamos dividir X^i por g:

$$\frac{X^i}{g} = \frac{\tilde{d}_0}{g} + \sum_{j=1}^{m} \tilde{d}_j g^{j-1}$$

O primeiro termo do lado direito é fracionário e o segundo é inteiro. Ou seja, o resto da divisão fornece o dígito $\tilde{d}_0$. Basta repetir o processo com a parte inteira.

Mudança de base

Seja, então, X um número representado na base b como $d_n d_{n-1} \cdots d_1 d_0, d_{-1} \ldots_b$. Encontrar a sua representação em uma outra base g significa encontrar a sequência de algarismos $\tilde{d}_m \tilde{d}_{m-1} \cdots \tilde{d}_1 \tilde{d}_0, \tilde{d}_{-1} \ldots_g$ tal que

$$X = \sum_{j \leq m} \tilde{d}_j g^j.$$

3º passo: vamos tratar a parte fracionária. Vamos multiplicar X^f por g:

$$g X^f = \tilde{d}_{-1} + \sum_{j \leq -2} \tilde{d}_j g^{j+1}.$$

O primeiro termo do lado direito é inteiro e o segundo é fracionário. Ou seja, a parte inteira da multiplicação fornece o dígito $\tilde{d}_{-1}$. Basta repetir o processo com a parte fracionária.

Exemplos

1º exemplo. Representar o número 9,2 em base binária.

Exemplos

1º exemplo. Representar o número 9,2 em base binária.
Inicialmente separamos a parte inteira, 9 da parte fracionária 0,2. Vamos tratar da parte inteira.

Exemplos

1º exemplo. Representar o número 9,2 em base binária.
Inicialmente separamos a parte inteira, 9 da parte fracionária 0,2. Vamos tratar da parte inteira.

$$\frac{9}{2} = 4 + \frac{\boxed{1}}{2}$$

Exemplos

1º exemplo. Representar o número 9,2 em base binária.
Inicialmente separamos a parte inteira, 9 da parte fracionária 0,2. Vamos tratar da parte inteira.

$$\frac{9}{2} = 4 + \frac{\boxed{1}}{2}$$

$$\frac{4}{2} = 2 + \frac{\boxed{0}}{2}$$

Exemplos

1º exemplo. Representar o número 9,2 em base binária.
Inicialmente separamos a parte inteira, 9 da parte fracionária 0,2. Vamos tratar da parte inteira.

$$\frac{9}{2} = 4 + \frac{\boxed{1}}{2}$$

$$\frac{4}{2} = 2 + \frac{\boxed{0}}{2}$$

$$\frac{2}{2} = 1 + \frac{\boxed{0}}{2}$$

Exemplos

1º exemplo. Representar o número 9,2 em base binária.
Inicialmente separamos a parte inteira, 9 da parte fracionária 0,2. Vamos tratar da parte inteira.

$$\frac{9}{2} = 4 + \frac{\boxed{1}}{2}$$

$$\frac{4}{2} = 2 + \frac{\boxed{0}}{2}$$

$$\frac{2}{2} = 1 + \frac{\boxed{0}}{2}$$

$$\frac{1}{2} = 0 + \frac{\boxed{1}}{2}$$

Exemplos

1º exemplo. Representar o número 9,2 em base binária.
Inicialmente separamos a parte inteira, 9 da parte fracionária 0,2. Vamos tratar da parte inteira.

$$\frac{9}{2} = 4 + \frac{\boxed{1}}{2}$$

$$\frac{4}{2} = 2 + \frac{\boxed{0}}{2}$$

$$\frac{2}{2} = 1 + \frac{\boxed{0}}{2}$$

$$\frac{1}{2} = 0 + \frac{\boxed{1}}{2}$$

$$9 = 1001_2$$

Exemplos

1º exemplo. Representar o número 9,2 em base binária.
Parte fracionária:

Exemplos

1º exemplo. Representar o número 9,2 em base binária.
Parte fracionária:

$$0{,}2 \times 2 = \boxed{0},4$$

Exemplos

1º exemplo. Representar o número 9,2 em base binária.
Parte fracionária:

$$0{,}2 \times 2 = \boxed{0},4$$
$$0{,}4 \times 2 = \boxed{0},8$$

Exemplos

1º exemplo. Representar o número 9,2 em base binária.
Parte fracionária:

$$0{,}2 \times 2 = \boxed{0},4$$
$$0{,}4 \times 2 = \boxed{0},8$$
$$0{,}8 \times 2 = \boxed{1},6$$

Exemplos

1º exemplo. Representar o número 9,2 em base binária.
Parte fracionária:

$$0{,}2 \times 2 = \boxed{0},4$$
$$0{,}4 \times 2 = \boxed{0},8$$
$$0{,}8 \times 2 = \boxed{1},6$$
$$0{,}6 \times 2 = \boxed{1},2$$

Exemplos

1º exemplo. Representar o número 9,2 em base binária.

$$0{,}2 \times 2 = \boxed{0},4$$
$$0{,}4 \times 2 = \boxed{0},8$$
$$0{,}8 \times 2 = \boxed{1},6$$
$$0{,}6 \times 2 = \boxed{1},2$$

A parte fracionária no último produto é 0,2. Portanto toda a sequência será repetida sucessivamente, ou seja,

$$0{,}2 = 0{,}\overline{0011}_2.$$

Exemplos

1º exemplo. Representar o número 9,2 em base binária.
A parte fracionária no último produto é 0,2. Portanto toda a sequência será repetida sucessivamente, ou seja,

$$0{,}2 = 0{,}\overline{0011}_2.$$

E finalmente

$$9{,}2 = 1001{,}\overline{0011}_2.$$

Exemplos

2º Exemplo. Vamos agora representar o número $53,2\overline{05}_6$ no sistema base-8.

Exemplos

2º Exemplo. Vamos agora representar o número $53,2\overline{05}_6$ no sistema base-8.

Como o número é conhecido na base 6, todas as operações devem ser feitas nessa base.
Para facilitar as operações é conveniente utilizar uma tabela de multiplicações em base 6 (tabuada em base 6):

	1_6	2_6	3_6	4_6	5_6
1_6	1_6	2_6	3_6	4_6	5_6
2_6	2_6	4_6	10_6	12_6	14_6
3_6	3_6	10_6	13_6	20_6	23_6
4_6	4_6	12_6	20_6	24_6	32_6
5_6	5_6	14_6	23_6	32_6	41_6

Exemplos

2º Exemplo. Vamos agora representar o número $53,2\overline{05}_6$ no sistema base-8.

Parte inteira:

Exemplos

2º Exemplo. Vamos agora representar o número $53,2\overline{05}_6$ no sistema base-8.

Parte inteira:

Observação

Como o número é conhecido na representação em base-6, é nessa base que as operações são feitas. Em particular, a nova base na representação base-6 é $8 = 12_6$.

Exemplos

2º Exemplo. Vamos agora representar o número $53,2\overline{05}_6$ no sistema base-8.

Parte inteira:

$$\frac{53_6}{12_6} = 4 + \frac{\boxed{1}}{12_6}$$

Exemplos

2º Exemplo. Vamos agora representar o número $53,2\overline{05}_6$ no sistema base-8.

Parte inteira:

$$\frac{53_6}{12_6} = 4 + \frac{\boxed{1}}{12_6}$$

$$\frac{4}{12_6} = 0 + \frac{\boxed{4}}{12_6}$$

Exemplos

2º Exemplo. Vamos agora representar o número $53,2\overline{05}_6$ no sistema base-8.

Parte inteira:

$$\frac{53_6}{12_6} = 4 + \frac{\boxed{1}}{12_6}$$

$$\frac{4}{12_6} = 0 + \frac{\boxed{4}}{12_6}$$

$$53_6 = 41_8$$

Exemplos

2º Exemplo. Vamos agora representar o número $53,2\overline{05}_6$ no sistema base-8.

Parte fracionária:

Exemplos

2º Exemplo. Vamos agora representar o número $53,2\overline{05}_6$ no sistema base-8.

Parte fracionária:

$$0,2\overline{05}_6 \times 12_6 = \begin{array}{r} 0,2\overline{05}_6 \\ \times \quad 12_6 \\ \hline \end{array}$$

Exemplos

2º Exemplo. Vamos agora representar o número $53,2\overline{05}_6$ no sistema base-8.

Parte fracionária:

$$0,2\overline{05}_6 \times 12_6 = \begin{array}{r} 0,2\overline{05}_6 \\ \times \quad 12_6 \\ \hline 0,\overline{41}_6 \end{array}$$

Exemplos

2º Exemplo. Vamos agora representar o número $53,2\overline{05}_6$ no sistema base-8.

Parte fracionária:

$$0,2\overline{05}_6 \times 12_6 = \begin{array}{rr} & 0,2\overline{05}_6 \\ \times & 12_6 \\ \hline & 0,\overline{41}_6 \\ + & 2,\overline{05}_6 \\ \hline \end{array}$$

Exemplos

2º Exemplo. Vamos agora representar o número $53,2\overline{05}_6$ no sistema base-8.

Parte fracionária:

$$0,2\overline{05}_6 \times 12_6 = \begin{array}{rr} & 0,2\overline{05}_6 \\ \times & 12_6 \\ \hline & 0,\overline{41}_6 \\ + & 2,\overline{05}_6 \\ \hline & \boxed{2},\overline{50}_6 \end{array}$$

$d_{-1} = 2$

Exemplos

2º Exemplo. Vamos agora representar o número $53,2\overline{05}_6$ no sistema base-8.

Parte fracionária:

$$0,\overline{50}_6 \times 12_6 = \begin{array}{r} 0,\overline{50}_6 \\ \times \quad 12_6 \\ \hline \end{array}$$

$d_{-1} = 2,\ d_{-2} = \ldots$

Exemplos

2º Exemplo. Vamos agora representar o número $53,2\overline{05}_6$ no sistema base-8.

Parte fracionária:

$$0,\overline{50}_6 \times 12_6 = \begin{array}{rr} & 0,\overline{50}_6 \\ \times & 12_6 \\ \hline & 1,\overline{41}_6 \\ + & 5,\overline{05}_6 \\ \hline & \boxed{10},\overline{50}_6 \end{array}$$

$d_{-1} = 2$, $d_{-2} = 6$ pois $10_6 = 6$.

Exemplos

2º Exemplo. Vamos agora representar o número $53,2\overline{05}_6$ no sistema base-8.

Parte fracionária:

$$0,\overline{50}_6 \times 12_6 = \begin{array}{rr} & 0,\overline{50}_6 \\ \times & 12_6 \\ \hline & 1,\overline{41}_6 \\ + & 5,\overline{05}_6 \\ \hline & \boxed{10},\overline{50}_6 \end{array}$$

$d_{-1} = 2$, $d_{-2} = 6$ pois $10_6 = 6$.
Como a fração restante é $\overline{50}_6$, teremos $d_{-3} = d_{-4} = \ldots = 6$. Ou seja,

$$0,2\overline{05}_6 = 0,2\overline{6}_8.$$

E

$$53,2\overline{05}_6 = 41,2\overline{6}_8.$$

Aritmética

O que é necessário para realizar uma operação de soma? Vamos analisar as etapas de uma operação em base decimal.

Aritmética

O que é necessário para realizar uma operação de soma? Vamos analisar as etapas de uma operação em base decimal.

$$\begin{array}{rrrr} & 3 & 1 & 7 \\ + & 5 & 8 & 5 \\ \hline \end{array}$$

Aritmética

O que é necessário para realizar uma operação de soma? Vamos analisar as etapas de uma operação em base decimal.

$$\begin{array}{rrrr} & 3 & 1 & 7 \\ + & 5 & 8 & 5 \\ \hline \end{array} \iff \begin{array}{rr} & 3\times10^2+1\times10^1+7\times10^0 \\ + & 5\times10^2+8\times10^1+5\times10^0 \\ \hline \end{array}$$

Aritmética

O que é necessário para realizar uma operação de soma? Vamos analisar as etapas de uma operação em base decimal.

$$\begin{array}{rccc} & 3 & 1 & 7 \\ + & 5 & 8 & 5 \\ \hline \end{array} \iff \begin{array}{rc} & 3\times10^2+1\times10^1+7\times10^0 \\ + & 5\times10^2+8\times10^1+5\times10^0 \\ \hline & (7+5)\times10^0 \end{array}$$

Aritmética

O que é necessário para realizar uma operação de soma? Vamos analisar as etapas de uma operação em base decimal.

$$\begin{array}{rrrr} & 3 & 1 & 7 \\ + & 5 & 8 & 5 \\ \hline \end{array} \iff \begin{array}{rr} & 3\times10^2+1\times10^1+7\times10^0 \\ + & 5\times10^2+8\times10^1+5\times10^0 \\ \hline & (10+2)\times10^0 \end{array}$$

Aritmética

O que é necessário para realizar uma operação de soma? Vamos analisar as etapas de uma operação em base decimal.

$$\begin{array}{rccc} & & 1 & \\ & 3 & 1 & 7 \\ + & 5 & 8 & 5 \\ \hline & & & 2 \end{array} \iff \begin{array}{rc} & 3\times 10^2+1\times 10^1+7\times 10^0 \\ + & 5\times 10^2+8\times 10^1+5\times 10^0 \\ \hline & (1+1+8)\times 10^1+2\times 10^0 \end{array}$$

Aritmética

O que é necessário para realizar uma operação de soma? Vamos analisar as etapas de uma operação em base decimal.

$$\begin{array}{rccc} & & 1 & \\ & 3 & 1 & 7 \\ + & 5 & 8 & 5 \\ \hline & & & 2 \end{array} \iff \begin{array}{rc} & 3\times10^2+1\times10^1+7\times10^0 \\ + & 5\times10^2+8\times10^1+5\times10^0 \\ \hline & (1+1+8)\times10^1+2\times10^0 \end{array}$$

Aritmética

O que é necessário para realizar uma operação de soma? Vamos analisar as etapas de uma operação em base decimal.

$$\begin{array}{rcccc} & & 1 & & \\ & 3 & 1 & 7 & \\ + & 5 & 8 & 5 & \\ \hline & & & 2 & \end{array} \quad \Longleftrightarrow \quad \begin{array}{rr} & 3\times 10^2+1\times 10^1+7\times 10^0 \\ + & 5\times 10^2+8\times 10^1+5\times 10^0 \\ \hline & (10+0)\times 10^1+2\times 10^0 \end{array}$$

Aritmética

O que é necessário para realizar uma operação de soma? Vamos analisar as etapas de uma operação em base decimal.

$$\begin{array}{rccc} & {\scriptstyle 1} & {\scriptstyle 1} & \\ & 3 & 1 & 7 \\ + & 5 & 8 & 5 \\ \hline & & 0 & 2 \end{array} \iff \begin{array}{r} 3\times 10^2 + 1\times 10^1 + 7\times 10^0 \\ + \quad 5\times 10^2 + 8\times 10^1 + 5\times 10^0 \\ \hline (1+3+5)\times 10^2 + 0\times 10^1 + 2\times 10^0 \end{array}$$

Aritmética

O que é necessário para realizar uma operação de soma? Vamos analisar as etapas de uma operação em base decimal.

$$\begin{array}{rrrr} & \color{green}{1} & \color{blue}{1} & \\ & \color{green}{3} & 1 & 7 \\ + & \color{green}{5} & 8 & 5 \\ \hline & & \color{red}{0} & \color{red}{2} \end{array} \iff \begin{array}{r} 3\times 10^2 + 1\times 10^1 + 7\times 10^0 \\ + \quad 5\times 10^2 + 8\times 10^1 + 5\times 10^0 \\ \hline (\color{green}{1}+\color{green}{3}+\color{green}{5})\times 10^2 + \color{red}{0}\times 10^1 + \color{red}{2}\times 10^0 \end{array}$$

Aritmética

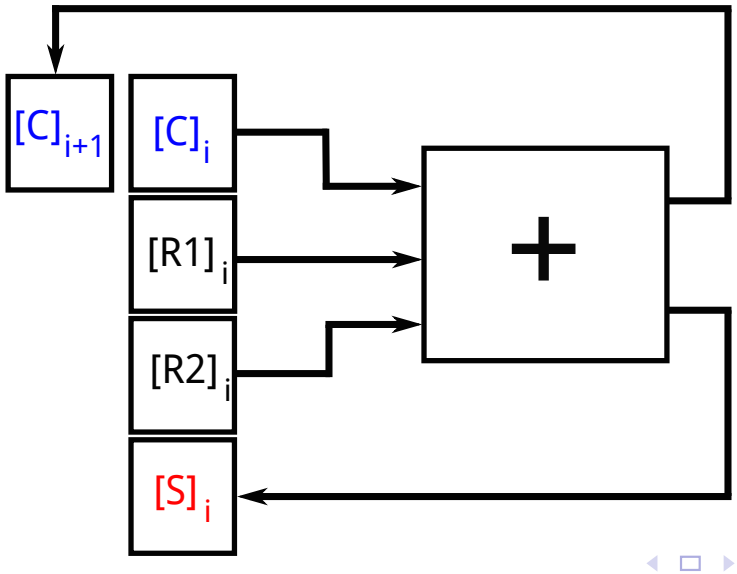
O que é necessário para realizar uma operação de soma? Vamos analisar as etapas de uma operação em base decimal.

$$\begin{array}{rccc} & {\scriptstyle 1} & {\scriptstyle 1} & \\ & 3 & 1 & 7 \\ + & 5 & 8 & 5 \\ \hline & 9 & 0 & 2 \end{array} \iff \begin{array}{rc} & 3\times10^2+1\times10^1+7\times10^0 \\ + & 5\times10^2+8\times10^1+5\times10^0 \\ \hline & 9\times10^2+0\times10^1+2\times10^0 \end{array}$$

Aritmética

Podemos notar que o processo de soma pode ser entendido como uma operação sobre os algarismos de cada termo (registros $R1$ e $R2$) e do carregamento (registro C)

$$\begin{array}{rcccc} & 1 & 1 & & \\ & 3 & 1 & 7 & \\ + & 5 & 8 & 5 & \\ \hline & 9 & 0 & 2 & \end{array} \iff \begin{array}{r|cccc} C: & 0 & 1 & 1 & 0 \\ R1: & 0 & 3 & 1 & 7 \\ R2: & 0 & 5 & 8 & 5 \\ S: & 0 & 9 & 0 & 2 \end{array}$$



Aritmética binária e Lógica

A adição possui essa estrutura em qualquer base. Por outro lado, as operações de soma e multiplicação de inteiros na base binária envolvem uma número menor de algarismos:

$+$	0	1
0	0	1
1	1	10

$\times$	0	1
0	0	0
1	0	1

Aritmética binária e Lógica

A adição possui essa estrutura em qualquer base. Por outro lado, as operações de soma e multiplicação de inteiros na base binária envolvem uma número menor de algarismos:

$+$	0	1
0	0	1
1	1	10

$\times$	0	1
0	0	0
1	0	1

Se considerarmos apenas o resultado do primeiro dígito nas operações de soma:

$+$	0	1
0	0	1
1	1	0

$\times$	0	1
0	0	0
1	0	1

Aritmética binária e Lógica

A adição possui essa estrutura em qualquer base. Por outro lado, as operações de soma e multiplicação de inteiros na base binária envolvem uma número menor de algarismos:

$+$	0	1
0	0	1
1	1	10

$\times$	0	1
0	0	0
1	0	1

Se considerarmos apenas o resultado do primeiro dígito nas operações de soma:

$+$	0	1
0	0	1
1	1	0

$\times$	0	1
0	0	0
1	0	1

Essas operações podem ser entendidas como operações lógicas (George Boole, "Mathematical Analysis of Logic" [1847]).

Aritmética binária e Lógica

A adição possui essa estrutura em qualquer base. Por outro lado, as operações de soma e multiplicação de inteiros na base binária envolvem uma número menor de algarismos:

$+$	0	1
0	0	1
1	1	10

$\times$	0	1
0	0	0
1	0	1

Se considerarmos apenas o resultado do primeiro dígito nas operações de soma:

$+$	F	V
F	F	V
V	V	F

$\times$	F	V
F	F	F
V	F	V

Essas operações podem ser entendidas como operações lógicas (George Boole, "Mathematical Analysis of Logic" [1847]). Se 0 representa o valor lógico "falso", F, e 1 representa valor lógico "verdadeiro", V, as tabelas acima codificam as operações "e lógico" ou "conjunção lógica, AND, e "ou lógico exclusivo" ou "disjunção exclusiva", XOR.

Aritmética binária e Lógica

A adição possui essa estrutura em qualquer base. Por outro lado, as operações de soma e multiplicação de inteiros na base binária envolvem uma número menor de algarismos:

$+$	0	1
0	0	1
1	1	10

$\times$	0	1
0	0	0
1	0	1

Se considerarmos apenas o resultado do primeiro dígito nas operações de soma:

XOR	F	V
F	F	V
V	V	F

AND	F	V
F	F	F
V	F	V

Essas operações podem ser entendidas como operações lógicas (George Boole, "Mathematical Analysis of Logic" [1847]). Se 0 representa o valor lógico "falso", F, e 1 representa valor lógico "verdadeiro", V, as tabelas acima codificam as operações "e lógico" ou "conjunção lógica, AND, e "ou lógico exclusivo" ou "disjunção exclusiva", XOR.

Aritmética binária e Lógica

A adição possui essa estrutura em qualquer base. Por outro lado, as operações de soma e multiplicação de inteiros na base binária envolvem uma número menor de algarismos:

$+$	0	1
0	0	1
1	1	10

$\times$	0	1
0	0	0
1	0	1

Se considerarmos apenas o resultado do primeiro dígito nas operações de soma:

XOR	F	V
F	F	V
V	V	F

AND	F	V
F	F	F
V	F	V

Note que o segundo dígito na operação de soma corresponde aos valores da tabela AND.

Portas lógicas

As operações lógicas XOR e AND são duas das 16 possíveis operações lógicas binárias: se consideramos dois operandos $R1$ e $R2$ teremos quatro possíveis configurações,

Portas lógicas

As operações lógicas XOR e AND são duas das 16 possíveis operações lógicas binárias: se consideramos dois operandos $R1$ e $R2$ teremos quatro possíveis configurações,

$R1$	$R2$
V	V
V	F
F	V
F	F

Portas lógicas

As operações lógicas XOR e AND são duas das 16 possíveis operações lógicas binárias: se consideramos dois operandos $R1$ e $R2$ teremos quatro possíveis configurações,

$R1$	$R2$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
V	V	F	F	F	F	F	F	F	F	V	V	V	V	V	V	V	V
V	F	F	F	F	F	V	V	V	V	F	F	F	F	V	V	V	V
F	V	F	F	V	V	F	F	V	V	F	F	V	V	F	F	V	V
F	F	F	V	F	V	F	V	F	V	F	V	F	V	F	V	F	V

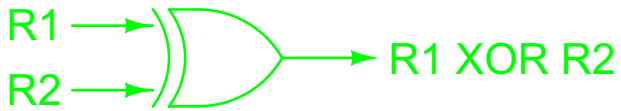
e consequentemente 16 (2^4 possíveis resultados).

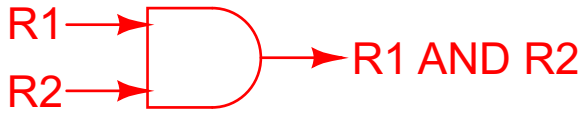
Portas lógicas

As operações lógicas XOR e AND são duas das 16 possíveis operações lógicas binárias: se consideramos dois operandos $R1$ e $R2$ teremos quatro possíveis configurações,

$R1$	$R2$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
V	V	F	F	F	F	F	F	F	F	V	V	V	V	V	V	V	V
V	F	F	F	F	F	V	V	V	V	F	F	F	F	V	V	V	V
F	V	F	F	V	V	F	F	V	V	F	F	V	V	F	F	V	V
F	F	F	V	F	V	F	V	F	V	F	V	F	V	F	V	F	V

As operações 6 e 8 correspondem às operações XOR e AND, respectivamente. Algumas dessas operações são determinadas "portas lógicas" e possuem representação diagramática. É o caso das operações XOR e AND:



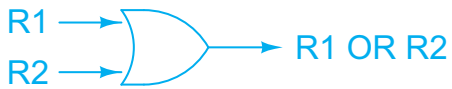


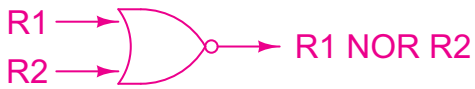
Portas lógicas

As operações lógicas XOR e AND são duas das 16 possíveis operações lógicas binárias: se consideramos dois operandos $R1$ e $R2$ teremos quatro possíveis configurações,

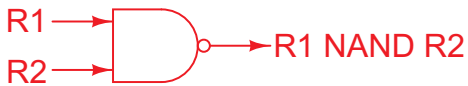
$R1$	$R2$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
V	V	F	F	F	F	F	F	F	F	V	V	V	V	V	V	V	V
V	F	F	F	F	F	V	V	V	V	F	F	F	F	V	V	V	V
F	V	F	F	V	V	F	F	V	V	F	F	V	V	F	F	V	V
F	F	F	V	F	V	F	V	F	V	F	V	F	V	F	V	F	V

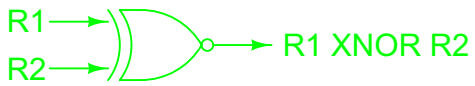
Outras portas lógicas comuns são o "ou lógico" e sua negação (OR e NOR respectivamente), as operações de negação (NOT) e as negações dos operadores AND e XOR:

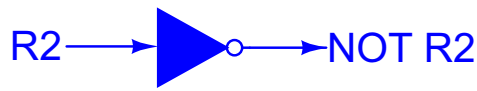
op.14: 

op. 1: 

op. 3:

op. 7: 

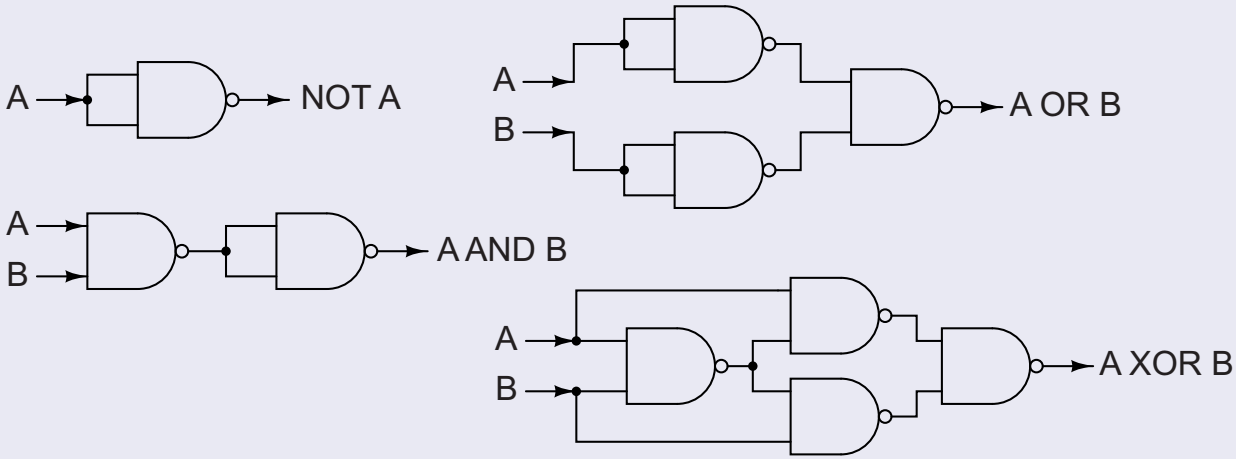
op. 9: 

op. 5: 

Portas lógicas

Observação

As operações NAND e NOR são "funcionalmente completas": cada uma delas é capaz de gerar todas as demais operações lógicas binárias. Como exemplo:



Adição

A soma de dois registros de 1 bit $R1$ e $R2$, $R1 + R2 = S$, equivale às seguintes operações lógicas:

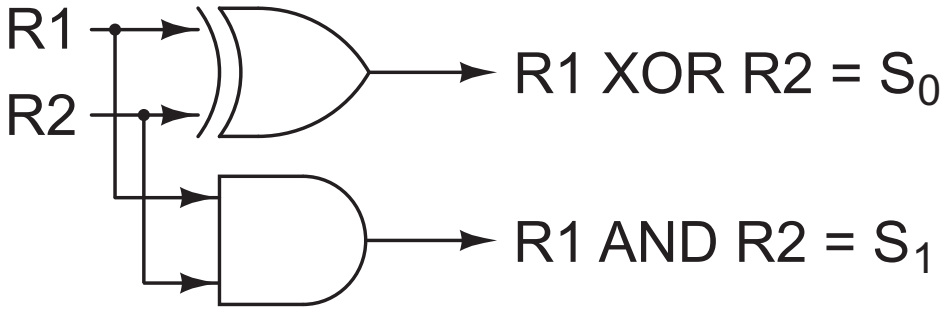
- 1º dígito de S é dado por $R1$ XOR $R2$.

Adição

A soma de dois registros de 1 bit $R1$ e $R2$, $R1 + R2 = S$, equivale às seguintes operações lógicas:

- 1º dígito de S é dado por $R1$ XOR $R2$.

- 2º dígito de S é dado por $R1$ AND $R2$.



Para que não haja perda de informação, o resultado deve ser guardado em um registro com dois bits. Se, os registros contiverem mais de um bit, será necessário incluir as operações lógicas para adicionar os carregamentos.

Adição

Vamos considerar a soma de dois registros com um número arbitrário de bits.
Sejam S e C os registros com informação sobre a soma e os bits que devem ser carregados ao longo da mesma.
Em cada posição do registro, devemos realizar a operação $[R1]_i + [R2]_i + [C]_i$ e do resultado devemos separar o bit que corresponde ao valor do registro na posição, $[S]_i$ e o bit que será carregado para a soma na $(i+1)$-ésima posição, $[C]_{i+1}$.

- Segue do resultado sobre o 1º dígito em uma soma.

$$[S]_i = ([R1]_i \text{XOR} [R2]_i) \text{XOR} [C]_i$$

- Haverá valor a ser carregado se os dois registros forem 1 ou se o bit carregado da soma anterior e o 1º dígito da soma dos registros forem 1.

$$[C]_{i+1} = ([R1]_i \text{AND} [R2]_i) \text{OR} ([C]_i \text{AND} ([R1]_i \text{XOR} [R2]_i))$$

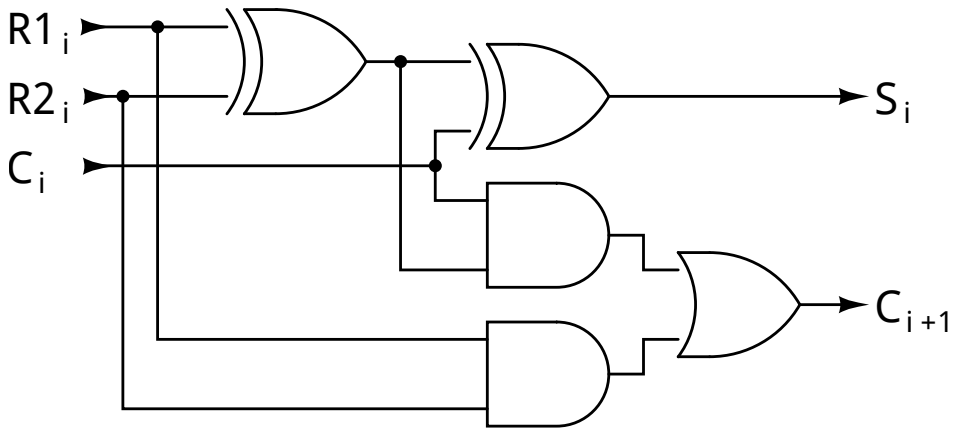
Adição

- Segue do resultado sobre o 1º dígito em uma soma.

$$[S]_i = ([R1]_i \text{XOR} [R2]_i) \text{XOR} [C]_i$$

- Haverá valor a ser carregado se os dois registros forem 1 ou se o bit carregado da soma anterior e o 1º dígito da soma dos registros forem 1.

$$[C]_{i+1} = ([R1]_i \text{AND} [R2]_i) \text{OR} ([C]_i \text{AND} ([R1]_i \text{XOR} [R2]_i))$$



Representação de números inteiros

Representação de números inteiros

Tipicamente, um número inteiro é armazenado em um processador como uma sequência de dígitos binários de comprimento fixo denominada registro.
Se todo o registro for utilizado para representar um inteiro não negativo a representação é única: um registro de n bits da forma

d_{n-1}	d_{n-2}	d_{n-3}	$\ldots$	d_2	d_1	d_0

representa o número $(d_{n-1}d_{n-2}d_{n-3}\ldots d_2d_1d_0)_2$. Assim, é possível representar os números inteiros entre 0 , representado por $000\ldots000_2$ até o inteiro representado por $111\ldots111_2$:

$$111\ldots111_2 = 2^{n-1}+2^{n-2}+2^{n-3}+\cdots+2^2+2^1+2^0 = 2^n-1.$$

Quanto aos inteiros com sinal, existem diferentes maneiras de representá-los através de um registro binário. Veremos três das maneiras mais comuns: representação com bit de sinal, representação complemento de um e representação complemento de dois.

Representação de números inteiros

Bit de sinal: essa forma de representação foi utilizada nos primeiros computadores digitais binários produzidos comercialmente. Consiste na utilização de um dos bits para o sinal. Geralmente, o bit mais significativo no registro (o primeiro à esquerda) é utilizado para esse fim.
De acordo com essa representação, um registro de máquina formado por uma sequência de n bits :

s	d_{n-2}	d_{n-3}	$\ldots$	d_2	d_1	d_0

é utilizado para representar o número binário $((-1)^s\, d_{n-2}\, d_{n-3} \ldots d_2\, d_1\, d_0)_2$. Ou seja, a sequência binária de n dígitos (ou bits) $s\, d_{n-2}\, d_{n-3} \ldots d_2\, d_1\, d_0$ é capaz de representar todos os inteiros entre $-2^{n-1}+1$ e $2^{n-1}-1$.
Nesse caso o maior inteiro representável é dado numeral $I_{max} = 011\ldots11_2$. Portanto

$$I_{max} = (-1)^0 \left(1 + 2^1 + 2^2 + \ldots + 2^{n-2}\right) = 2^{n-1} - 1.$$

Essa técnica de registro com n bits permite armazenar em uma máquina todos os $2^n - 1$ números inteiros entre $-\left(2^{n-1}-1\right)$ e $2^{n-1}-1$.

Representação de números inteiros

Complemento de um: consiste na utilização de um registro de n bits da seguinte forma: o primeiro bit representa um termo aditivo $-\left(2^{n-1}-1\right)$ e o restante dos bits representam um inteiro não negativo. Assim um registro n bits da forma

d_{n-1}	d_{n-2}	d_{n-3}	$\ldots$	d_2	d_1	d_0

representa o inteiro $(d_{n-2}d_{n-3}\ldots d_2 d_1 d_0)_2 - d_{n-1}\left(2^{n-1}-1\right)$.
Essa forma permite representar os inteiros entre $-\left(2^{n-1}-1\right)$ e $2^{n-1}-1$ e o zero possui também duas representações: $111\ldots11$ e $000\ldots00_2$.

Representação de números inteiros

Complemento de dois: é a representação mais utilizada pelo processadores atuais. Consiste na utilização de um registro de n bits da seguinte forma: o primeiro bit representa um termo aditivo -2^{n-1} e o restante dos bits representam um inteiro não negativo. Assim um registro n bits da forma

d_{n-1}	d_{n-2}	d_{n-3}	$\ldots$	d_2	d_1	d_0

representa o inteiro $(d_{n-2}d_{n-3}\ldots d_2 d_1 d_0)_2 - d_{n-1}2^{n-1}$.
Essa forma permite representar os inteiros entre -2^{n-1} e $2^{n-1}-1$ e possui a vantagem de possuir uma única representação para o zero: $000\ldots00_2$.

Representação de ponto fixo

É possível estender a técnica utilizada na representação dos inteiros para representar com precisão finita, números que possuam parte fracionária.
Representação ponto-fixo: dados dois inteiros positivos p e q, interpretamos um registro de $n = p + q + 1$ bits como a divisão por 2^q do inteiro de n bits na representação complemento de dois. Vamos simbolizar tal registro como $R(p, q)$.
Exemplo:

d_{31}	d_{30}	d_{29}	$\ldots$	d_2	d_1	d_0

corresponde ao registro $R(15, 16)$ e representa o número

$$\left((d_{30}d_{29}\ldots d_2 d_1 d_0)_2 - d_{31}2^{31}\right)2^{-16}$$

$$= \underbrace{\left((d_{30}d_{29}\ldots d_{17}d_{16})_2 - d_{31}2^{15}\right)}_{\text{parte inteira}} + \underbrace{(0, d_{15}d_{14}\ldots d_1 d_0)_2}_{\text{parte fracionária}}$$

O registro $R(p, q)$ é capaz de representar os números fracionários no intervalo $[-2^p, 2^p - 2^{-q}]$ em intervalos uniformemente espaçados de 2^{-q}.

Representação de ponto fixo

O intervalo $(-2^{-q}, 2^{-q})$ é denominada *região de underflow.*
A região $(-\infty, -2^{p}) \cup (2^{p} - 2^{-q}, +\infty)$ é denominada *região de overflow.*

Representação de ponto fixo

O intervalo $(-2^{-q}, 2^{-q})$ é denominada *região de underflow.*
A região $(-\infty, -2^{p}) \cup (2^{p} - 2^{-q}, +\infty)$ é denominada *região de overflow.*

Observações

Vantagens: oferecer uma representação para números com parte fracionária que podem ser trabalhados dentro de ALUs
Desvantagens:representar números distintos com precisão diferente.

Exemplo:
Devido à familiaridade, vamos considerar os registros com 4 dígitos na parte inteira e 4 dígitos na parte fracionária em base 10, $9999{,}\overline{1234}$ e $0{,}0012\overline{113}$ e um registro.

Pontos flutuantes

Definição: Ponto flutuante

A representação x de um número real é denominada ponto flutuante normalizado na base $b \in \mathbb{N}$, $b \geq 2$, se forem satisfeitas as propriedades

1. $x = m\, b^{\varepsilon}$, onde
2. $m = \pm d_1, d_2 \ldots d_n \qquad n \in \mathbb{N}$,
3. $1 \leq d_1 \leq b-1 \quad$ e $\quad 0 \leq d_i \leq b-1 \qquad$ para $i = 2, 3, \ldots n$,
4. $e_1 \leq \varepsilon \leq e_2, \quad$ onde $\varepsilon, e_1, e_2 \in \mathbb{Z}$.

m é denominada significando, ε expoente e n o número de dígitos de precisão.

Pontos flutuantes

Definição: Ponto flutuante

A representação x de um número real é denominada ponto flutuante normalizado na base $b \in \mathbb{N}$, $b \geq 2$, se forem satisfeitas as propriedades

1. $x = m\, b^{\varepsilon}$, onde
2. $m = \pm d_1, d_2 \ldots d_n \qquad n \in \mathbb{N}$,
3. $1 \leq d_1 \leq b-1 \quad$ e $\quad 0 \leq d_i \leq b-1 \qquad$ para $i = 2, 3, \ldots n$,
4. $e_1 \leq \varepsilon \leq e_2, \quad$ onde $\varepsilon, e_1, e_2 \in \mathbb{Z}$.

m é denominada significando, ε expoente e n o número de dígitos de precisão.

Exemplos:

- O número $9999,\overline{1234}$ em representação de ponto flutuante em base 10 com 8 dígitos de precisão é $9{,}9991234 \cdot 10^3$.
- Utilizando essa mesma prescrição, o número $0{,}0012\overline{113}$ é representado como $1{,}2113131 \cdot 10^{-3}$.

Pontos flutuantes

Observações

- Em uma representação de ponto flutuante normalizado, o primeiro algarismo antes da vírgula é necessariamente maior ou igual a 1.
- O número 0 é incluído em um conjunto denominado "sistema de ponto flutuante" onde possui a representação $0{,}00\ldots0\,b^e$.
- O **sistema de ponto flutuante** $F(b, n, e_1, e_2)$ é definido como o conjunto de números que inclui o zero e os pontos flutuantes em base b com n dígitos de precisão e expoente que pode variar entre e_1 e e_2 inclusive.
- As operações aritméticas $\oplus$, $\ominus$, $\otimes$ e $\oslash$ são definidas sobre $F(b, n, e_1, e_2) \times F(b, n, e_1, e_2)$ a partir das operações aritméticas dos reais mas o resultado deve ser um elemento do $F(b, n, e_1, e_2)$. Isto implica a realização de um arredondamento.
- Algumas funções, como $\sqrt{}$ também devem ter esse comportamento. Para diferenciá-las das funções originais, utilizamos a notação fl antes da função. No exemplo considerado, $fl\sqrt{}$.

Pontos flutuantes

Ao contrário do que ocorre com os números representados por um esquema de ponto fixo, os elementos sucessivos do conjunto $F(b, n, e_1, e_2)$ não são igualmente espaçados.

Exemplo:

Consideremos o sistema $F(10, 2, -10, 10)$.
O elemento positivo mais próximo de zero é o numeral $1{,}0 \cdot 10^{-10}$, em seguida vem $1{,}1 \cdot 10^{-10}$, $1{,}2 \cdot 10^{-10}$, $\ldots$, $9{,}9 \cdot 10^{-10}$: o espaçamento é $0{,}1 \cdot 10^{-10}$.
O elemento seguinte é $1{,}0 \cdot 10^{-9}$, em seguida vem $1{,}1 \cdot 10^{-9}$, $1{,}2 \cdot 10^{-9}$, $\ldots$, $9{,}9 \cdot 10^{-9}$: o espaçamento é $0{,}1 \cdot 10^{-9}$.
E assim por diante até os últimos elementos positivos cujo espaçamento é $0{,}1 \cdot 10^{10}$.

Pontos flutuantes

Um outra propriedade importante dos pontos flutuantes diz respeito às propriedades algébricas que ao contrário dos reais, racionais e inteiros, em geral não são válidas.
Exemplo:
Sejam $x = 1 \cdot 10^{-3}$ e $y = z = 1 \cdot 10^{10}$, elementos do conjunto $F(10, 3, -10, 10)$.
Então

$$\begin{aligned} x \oplus (y \ominus z) &= 1{,}00 \cdot 10^{-3} \oplus (1{,}00 \cdot 10^{10} \ominus 1{,}00 \cdot 10^{10}) \\ &= 1{,}00 \cdot 10^{-3} \oplus 0 \\ &= 1{,}00 \cdot 10^{-3}, \end{aligned}$$

por outro lado

$$\begin{aligned} (x \oplus y) \ominus z &= (1{,}00 \cdot 10^{-3} \oplus 1{,}00 \cdot 10^{10}) \ominus 1{,}00 \cdot 10^{10} \\ &= 1{,}00 \cdot 10^{10} \ominus 1{,}00 \cdot 10^{10} \\ &= 0. \end{aligned}$$

Arredondamento

O operação de arredondamento consiste em encontrar um representação $\tilde{x}$ para um número x com uma determinada precisão.
Não há uma única forma de realizar o arredondamento e, de acordo com a aplicação, existem regras mais convenientes. Vamos discutir as cinco mais comuns e previstas pelo padrão da IEEE.
O resultado depende exclusivamente dos dígitos que compõe o número; a sua magnitude não desempenha papel algum. Dado um x,

$$x = \pm(d_0, d_{-1} \ldots)_b,$$

uma operação de arredondamento do número x resulta em um número $\tilde{x}$ com k dígitos de precisão

$$\tilde{x} = \pm(\tilde{d}_0, \tilde{d}_{-1} \ldots \tilde{d}_{-k+1})_b$$

onde os valores dos dígitos $\tilde{d}_0$, $\tilde{d}_{-1}$, $\ldots$, $\tilde{d}_{-k+1}$ são determinados a partir dos valores de d_{-k+1}, d_{-k}, d_{-k-1}, $\ldots$ e da escolha do tipo de arredondamento.

Arredondamento

Em comum, as operações de arredondamento possuem a característica de produzir uma resposta da forma

$$\bar{x} := \pm(d_0, d_{-1} \dots d_{-k+1})_b,$$

determinada "truncamento com k dígitos", ou da forma

$$\bar{x} \pm b^{-k+1} = \pm((d_0, d_{-1} \dots d_{-k+1})_b + (0,0 \dots 1)_b),$$

que consiste em adicionar uma unidade ao dígito d_{-k+1} na representação $\bar{x}$.
A seguir estão enumerados os cinco tipos de arredondamento mais comuns.
Os dois primeiros são arredondamentos para representante mais próximo, os demais são denominados “arredondamentos dirigidos”.

Arredondamento

1. Arredondamento para o mais próximo, desempate par: é o número com k dígitos mais próximo de x. Em caso de empate, isto é, se os dígitos sobressalentes $(0,d_{-k}d_{-k-1}\ldots)_b$ representarem a fração $\frac{1}{2}$, o dígito d_{-k+1} é arredondado para o algarismo seguinte se d_{-k+1} for ímpar e mantido se par.
2. Arredondamento para o mais próximo, desempate no sentido oposto ao zero: é o número com k dígitos mais próximo de x. Em caso de empate, o dígito d_{-k+1} é arredondado para o algarismo seguinte.
3. Arredondamento por truncamento ou no sentido do zero: os dígitos sobressalentes são descartados.
4. Arredondamento no sentido de $+\infty$: o dígito d_{-k+1} é arredondado para o algarismo seguinte se o numeral for positivo e os dígitos sobressalentes forem não nulos. Se o numeral for negativo e os dígito sobressalentes forem não nulos, o dígito d_{-k} é mantido.
5. Arredondamento no sentido de $-\infty$: o penúltimo dígito é mantido se o numeral for positivo e os dígitos sobressalentes forem não nulos. Se o numeral for negativo e os dígitos sobressalentes forem não nulos, o dígito d_{-k} é arredondado para o algarismo seguinte.

Arredondamento

O quadro abaixo apresenta o resultado das operações de arredondamento para três dígitos.

x	mais próximo, desempate par	mais próximo, desempate $\leftarrow 0 \rightarrow$	truncamento	sentido $+\infty$	sentido $-\infty$
13.140	13.100	13.100	13.100	13.200	13.100
0,01875	0,0188	0,0188	0,0187	0,0188	0,0187
-3,3196	-3,32	-3,32	-3,31	-3,31	-3,32
-2.145	-2.140	-2.150	-2.140	-2140	-2.150

Representação de pontos flutuante (IEEE754)

O padrão IEEE754 (a sigla se refere ao Institute of Electrical and Electronics Engineers), apresentado em 1985, foi desenvolvido com o objetivo de unificar as diversas implementações em máquina de registros e operações em ponto flutuante. A maioria dos processadores atuais possuem unidades especializadas, denominadas FPUs (Unidades de Ponto Flutuante) que suportam o padrão IEEE754 ou pelo menos suportam um subconjunto obrigatório das definições.
Além dos numerais em ponto flutuante, o padrão prevê que o registro pode conter informação sobre $+\infty$, $-\infty$, $+0$, -0, numerais subnormais (menores, em valor absoluto, que os elementos de $F(2, n, e_2, e_1)$) e os *NaN* ("not a number" reservado para operações ilegais como $\infty - \infty$, $0 \times \infty$, $0/0$ e $\sqrt{-1}$).

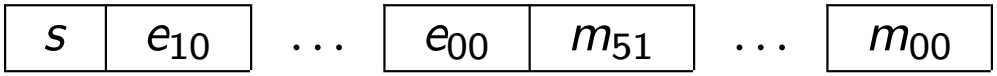
Representação de pontos flutuante (IEEE754)

O padrão prevê quatro tipos de registros:

- registros de 32 bits – precisão simples (***floats***);
- registros de 64 bits – precisão dupla (***doubles***);
- registros de precisão estendida (simples/dupla).

Representação de pontos flutuante (IEEE754)

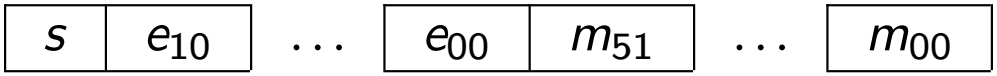
Exemplo: registro de 64 bits – contém todos os elementos de $F(2, 53, -1022, 1023)$.



- o bit s é responsável pelo sinal.
- os bits $e_{10}e_{09}\ldots e_{00}$ representam o expoente (11bits).
- os bits $m_{51}m_{50}\ldots m_{00}$ representam o significando (52 bits).

Representação de pontos flutuante (IEEE754)

Exemplo: registro de 64 bits – contém todos os elementos de $F(2, 53, -1022, 1023)$.



- o bit s é responsável pelo sinal.
- os bits $e_{10}e_{09}\ldots e_{00}$ representam o expoente (11bits).
- os bits $m_{51}m_{50}\ldots m_{00}$ representam o significando (52 bits).

Os bits do expoente podem representar os inteiros entre 1 e 2046 (quando todos os bits são iguais a 0 ou 1, há uma interpretação especial) com um desvio de 1023. Ou seja, os bits do expoente representam situações especiais e os inteiros entre -1022 e 1023.
Os 52 bits restantes são utilizados para representar o significando com 53 dígitos binários (já que o primeiro dígito é sempre igual a 1 em uma base binária, não há necessidade explícita de armazená-lo no registro e com isso ganha-se um bit extra).

Representação de pontos flutuante (IEEE754)

Situações especiais

- zeros: bits do expoente e do significando todos nulos. O bit de sinal pode ser igual a 0 ou 1, ou seja, há uma representação para $+0$ e -0.
- subnormais: bits do expoente todos nulos e os do significando e sinal guardam informação sobre o subnormal.
- infinitos: bits do expoente todos iguais a 1 e os do significando iguais a 0. O bit de sinal pode ser igual a 0 ou 1, ou seja, há uma representação para $+\infty$ e $-\infty$.
- *NaN*: bits do expoente todos iguais a 1 e os demais bits contém informação de diagnóstico.

Representação de pontos flutuante (IEEE754)

Exemplo: vamos encontrar o registro de 64bits equivalente ao numeral 1345,875. O primeiro passo é representar o número na base binária:

$$1345{,}875 = 10101000001{,}111_2.$$

Em seguida vamos reescrevê-lo como um ponto flutuante normalizado: $1{,}0101000001111 \cdot 2^{10}$. Ignoramos o 1 antes da vírgula e adicionamos tantos 0 à direita quantos forem necessários para preencher os 52 bits, dessa forma encontramos os bits do significando:

$$0101\,0000\,0111\,1000\,0000\,0000\,0000\,0000\,0000\,0000\,0000\,0000\,0000_2$$

Representação de pontos flutuante (IEEE754)

Exemplo: vamos encontrar o registro de 64bits equivalente ao numeral 1345,875. O primeiro passo é representar o número na base binária:

$$1345{,}875 = 10101000001{,}111_2.$$

Em seguida vamos reescrevê-lo como um ponto flutuante normalizado: $1{,}0101000001111 \cdot 2^{10}$. Ignoramos o 1 antes da vírgula e adicionamos tantos 0 à direita quantos forem necessários para preencher os 52 bits, dessa forma encontramos os bits do significando:

$$0101\,0000\,0111\,1000\,0000\,0000\,0000\,0000\,0000\,0000\,0000\,0000\,0000_2$$

Note que ainda não acabamos de montar o registro completo e a sequência de bits quase não cabe no slide. Na realidade, essa forma não é a mais conveniente de apresentar o registro. Uma das formas mais utilizadas é apresentar os "bytes" que compõe o registro.

Representação de pontos flutuante (IEEE754)

Observação: bits e bytes

Já vimos que o dígito binário é comumente denominado "bit", uma contração do termo em língua inglesa "binary digit", registrado pela primeira vez em um artigo do engenheiro americano Vannevar Bush, publicado em 1938. O termo "byte" foi criado em 1956 pelo cientista da computação Werner Buchholz durante a fase de desenvolvimento do primeiro supercomputador transistorizado (IBM 7030, conhecido como IBM Stretch) e correspondia a quantidade de bits necessária para codificar um caractere. Devido à popularidade da arquitetura IBM System/360 (anos 1960) e da família de microprocessadores de 8 bits lançados nos anos 1970, a quantia universalmente aceita é de oito bits para cada byte (em francês é comum utilizar o termo "octet" para designar um conjunto de oito bits).

Representação de pontos flutuante (IEEE754)

Assim, os registros de 64 bits correspondem a um conjunto de 8 bytes. Por sua vez, cada conjunto de quatro bits (conhecidos como "nibbles") corresponde a um único dígito hexadecimal, ou seja, um algarismo no conjunto

$$0, 1, 2, \ldots 9, A, B, \ldots, F.$$

Representação de pontos flutuante (IEEE754)

Assim, os registros de 64 bits correspondem a um conjunto de 8 bytes. Por sua vez, cada conjunto de quatro bits (conhecidos como "nibbles") corresponde a um único dígito hexadecimal, ou seja, um algarismo no conjunto

$$0, 1, 2, \ldots 9, A, B, \ldots, F.$$

Por exemplo,

$$0110_2 = 6_{16},$$
$$1100_2 = C_{16}$$

Representação de pontos flutuante (IEEE754)

Assim, os registros de 64 bits correspondem a um conjunto de 8 bytes. Por sua vez, cada conjunto de quatro bits (conhecidos como "nibbles") corresponde a um único dígito hexadecimal, ou seja, um algarismo no conjunto

$$0, 1, 2, \ldots 9, A, B, \ldots, F.$$

Por exemplo,

$$0110_2 = 6_{16},$$
$$1100_2 = C_{16}$$

e dessa forma

$$01101100_2 = 6C_{16}.$$

Voltando ao registro do número 1345,875...

Representação de pontos flutuante (IEEE754)

Exemplo: vamos encontrar o registro de 64bits equivalente ao numeral 1345,875. O primeiro passo é representar o número na base binária:

$$1345{,}875 = 10101000001{,}111_2.$$

Em seguida vamos reescrevê-lo como um ponto flutuante normalizado: $1{,}0101000001111 \cdot 2^{10}$. Ignoramos o 1 antes da vírgula e adicionamos tantos 0 à direita quantos forem necessários para preencher os 52 bits, dessa forma encontramos os bits do significando:

$$0101\,0000\,0111\,1000\,0000\,0000\,0000\,0000\,0000\,0000\,0000\,0000\,0000_2$$

O registro desses bits corresponde a seis e meio bytes:

$$50780000000000_{16}.$$

Os bytes restantes são formados pelos bits do sinal e do expoente.

Representação de pontos flutuante (IEEE754)

O expoente vale 10, com o deslocamento de 1023, o inteiro a ser representado pelos bits do expoente é o $1033 = 10000001001_2$. O bit de sinal é igual a 0 pois o número é positivo.
Assim, o registro dessa parte é formado pelos bits

$$0100000001001_2$$

que correspondem a um byte e meio

$$409_{16}.$$

O registro de 64 bits completo é dado por

$$4095078000000000_{16}.$$

Representação de pontos flutuante (IEEE754)

O seguinte código Scilab armazena o registro do número 1345,875 em formato de 64bits (como 8 bytes) no arquivo "double.bin".

```
nomearq=home+'/double.bin' // nome completo do arquivo.

mopen(nomearq,'wb');       // abre o arquivo para gravação

mput(1345.875,'dl');       // grava o número no formato
                           // 64bits little-endian.

mclose();                  // fecha o arquivo
```

O arquivo "double.bin" será salvo na área "home" (no Windows corresponde ao diretório com seu nome de usuário em "C:\Users\", no Linux corresponde ao diretório "~"). Os bytes podem ser visualizados com um editor hexadecimal (por exemplo, o "Frhed" para Windows , disponível em http://frhed.sourceforge.net/ ou o "0xED" para MacOS, disponível em http://www.suavetech.com/0xed/ ou o "ghex" para Linux).