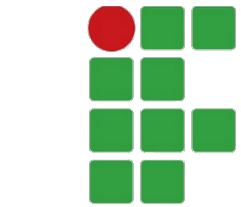
Matemática Computacional

Representação Numérica e Aritmética de Máquina

Felipe Augusto Lima Reis

felipe.reis@ifmg.edu.br



INSTITUTO FEDERAL
Minas Gerais

Sumário

Sistemas de Numeração

Sistemas de Numeração

- No dia a dia utilizamos o sistema decimal de numeração;
- Esse sistema é posicional, assim a posição do dígito indica a potência de 10 deste [Justo et al., 2020]
  - Ex.: o número 732, pode ser reescrito como

$$\begin{aligned} 732 &= (7 \times 10^2) + (3 \times 10^1) + (2 \times 10^0) \\ &= 700 + 30 + 2 \\ &= 732 \end{aligned}$$

Sistemas de Numeração

- Consideremos o número 57 em decimal:

$$57 = (5 \times 10^1) + (7 \times 10^0)$$

- Utilizando o mesmo sistema podemos representar valores como potência de qualquer número (*base = 2, 3, 4, ..., n*);
- Na base binária, esse mesmo número é definido como 111001

$$111001 = (1\times2^5)+(1\times2^4)+(1\times2^3)+(0\times2^2)+(0\times2^1)+(1\times2^0)$$

$$= 32 + 16 + 8 + 0 + 0 + 1$$

$$= 48 + 9$$

$$= 57$$

Sistemas de Numeração

INSTITUTO FEDERAL
Minas Gerais

- Número 57 (decimal) em binário (111001)

$$111001 = (1\times2^5)+(1\times2^4)+(1\times2^3)+(0\times2^2)+(0\times2^1)+(1\times2^0)$$

$$= 32 + 16 + 8 + 0 + 0 + 1$$

$$= 57$$

- Representação unicamente em base binária

$$111001 = (1\times2^5)+(1\times2^4)+(1\times2^3)+(0\times2^2)+(0\times2^1)+(1\times2^0)$$

$$= 100000 + 010000 + 001000 + 000000 + 000000 + 000001$$

$$= 111001$$

Sistemas de Numeração

- Nas bases octal e hexadecimal, comumente utilizadas em computação, temos o mesmo princípio...
- 57 (decimal) = 71 (octal) = 39 (hexadecimal)

$$71 = (7 \times 8^1) + (1 \times 8^0)$$

$$= 56 + 1$$

$$39 = (3 \times 16^1) + (9 \times 16^0)$$

$$= 48 + 9$$

Sistemas de Numeração

- Consideremos β a base de um sistema de numeração, onde $\beta > 1$;
- Um dado número $x \in \mathbb{R}$, na base β pode ser representado por:

$$x = \left(a_m a_{(m-1)} a_{(m-2)} \ldots a_2 a_1 a_0 \ , \ d_1 d_2 \ldots d_n\right)_\beta$$

- Em sua forma polinomial, o número pode ser representado por:

$$x = a_n\beta^m + a_{n-1}\beta^{m-1} + \ldots + a_1\beta^1 + a_0\beta^0 \ , \ d_1\beta^{-1} + d_2\beta^{-2} + \ldots + d_n\beta^{-n}$$

Sistemas de Numeração

- De acordo com a base, são possíveis os seguintes algarismos:
  - Decimal: $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
  - Binário: $\{0, 1\}$
  - Octal: $\{0, 1, 2, 3, 4, 5, 6, 7\}$
  - Hexadecimal: $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$
- De forma geral, uma base β permite os seguintes algarismos:

$$\{0, ..., \beta - 1\}$$

- Para bases $\beta \geq 10$ utilizam-se letras para representação dos algarismos subsequentes.

Mudança de Base

Mudança de Base - Método Prático

INSTITUTO FEDERAL
Minas Gerais

- Ex. 1: Conversão do número 732_{10} para a base binária
  1. Montar a tabela abaixo;
  2. Encontrar o primeiro valor maior que o valor a ser convertido;
  3. Colocar o valor zero na coluna correspondente a esse valor e nas colunas com valores superiores;

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1024	512	256	128	64	32	16	8	4	2	1
0										

Mudança de Base - Método Prático

- Ex. 1: Conversão do número 732_{10} para a base binária
  1. Montar a tabela abaixo;
  2. Encontrar o primeiro valor maior que o valor a ser convertido;
  3. Colocar o valor zero na coluna correspondente a esse valor e nas colunas com valores superiores;

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1024	512	256	128	64	32	16	8	4	2	1
0										

Mudança de Base - Método Prático

- Ex. 1: Conversão do número 732_{10} para a base binária
  4. Adicionar o valor 1 na coluna seguinte;
  5. Efetuar a subtração do número a ser convertido pela coluna atual ($732_{10} - 512_{10} = 220_{10}$; $220_{10} - 128_{10} = 92_{10}$, ...);
  6. Voltar ao item 2 e repetir até que a conversão do número.

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1024	512	256	128	64	32	16	8	4	2	1
0	1	0	1	1	0	1	1	1	0	0

Mudança de Base - Método Prático

- Ex. 1: Conversão do número 732_{10} para a base binária
  4. Adicionar o valor 1 na coluna seguinte;
  5. Efetuar a subtração do número a ser convertido pela coluna atual ($732_{10} - 512_{10} = 220_{10}$; $220_{10} - 128_{10} = 92_{10}$, ...);
  6. Voltar ao item 2 e repetir até que a conversão do número.

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1024	512	256	128	64	32	16	8	4	2	1
0	**1**	0	1	1	0	1	1	1	0	0

Mudança de Base - Método Prático

- Ex. 1: Conversão do número 732_{10} para a base binária
  4. Adicionar o valor 1 na coluna seguinte;
  5. Efetuar a subtração do número a ser convertido pela coluna atual ($732_{10} - 512_{10} = 220_{10}$; $220_{10} - 128_{10} = 92_{10}$, ...);
  6. Voltar ao item 2 e repetir até que a conversão do número.

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1024	512	256	128	64	32	16	8	4	2	1
0	**1**	**0**	1	1	0	1	1	1	0	0

Mudança de Base - Método Prático

INSTITUTO FEDERAL
Minas Gerais

- Ex. 1: Conversão do número 732_{10} para a base binária
  4. Adicionar o valor 1 na coluna seguinte;
  5. Efetuar a subtração do número a ser convertido pela coluna atual ($732_{10} - 512_{10} = 220_{10}$; $220_{10} - 128_{10} = 92_{10}$, ...);
  6. Voltar ao item 2 e repetir até que a conversão do número.

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1024	512	256	128	64	32	16	8	4	2	1
0	**1**	**0**	**1**	1	0	1	1	1	0	0

Mudança de Base - Método Prático

- Ex. 1: Conversão do número 732_{10} para a base binária
  4. Adicionar o valor 1 na coluna seguinte;
  5. Efetuar a subtração do número a ser convertido pela coluna atual ($732_{10} - 512_{10} = 220_{10}$; $220_{10} - 128_{10} = 92_{10}$, ...);
  6. Voltar ao item 2 e repetir até que a conversão do número.

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1024	512	256	128	64	32	16	8	4	2	1
0	**1**	**0**	**1**	**1**	0	1	1	1	0	0

Mudança de Base - Método Prático

- Ex. 1: Conversão do número 732_{10} para a base binária
  4. Adicionar o valor 1 na coluna seguinte;
  5. Efetuar a subtração do número a ser convertido pela coluna atual ($732_{10} - 512_{10} = 220_{10}$; $220_{10} - 128_{10} = 92_{10}$, ...);
  6. Voltar ao item 2 e repetir até que a conversão do número.

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1024	512	256	128	64	32	16	8	4	2	1
0	**1**	**0**	**1**	**1**	**0**	1	1	1	0	0

Mudança de Base - Método Prático

- Ex. 1: Conversão do número 732_{10} para a base binária
  4. Adicionar o valor 1 na coluna seguinte;
  5. Efetuar a subtração do número a ser convertido pela coluna atual ($732_{10} - 512_{10} = 220_{10}$; $220_{10} - 128_{10} = 92_{10}$, ...);
  6. Voltar ao item 2 e repetir até que a conversão do número.

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1024	512	256	128	64	32	16	8	4	2	1
0	**1**	**0**	**1**	**1**	**0**	**1**	1	1	0	0

Mudança de Base - Método Prático

- Ex. 1: Conversão do número 732_{10} para a base binária
  4. Adicionar o valor 1 na coluna seguinte;
  5. Efetuar a subtração do número a ser convertido pela coluna atual ($732_{10} - 512_{10} = 220_{10}$; $220_{10} - 128_{10} = 92_{10}$, ...);
  6. Voltar ao item 2 e repetir até que a conversão do número.

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1024	512	256	128	64	32	16	8	4	2	1
0	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**0**

Mudança de Base - Método Prático

INSTITUTO FEDERAL
Minas Gerais

- Ex. 1: Conversão do número 732_{10} para a base binária
  4. Adicionar o valor 1 na coluna seguinte;
  5. Efetuar a subtração do número a ser convertido pela coluna atual ($732_{10} - 512_{10} = 220_{10}$; $220_{10} - 128_{10} = 92_{10}$, ...);
  6. Voltar ao item 2 e repetir até que a conversão do número.

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1024	512	256	128	64	32	16	8	4	2	1
0	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**1**	0	0

Mudança de Base - Método Prático

INSTITUTO FEDERAL
Minas Gerais

- Ex. 1: Conversão do número 732_{10} para a base binária
  4. Adicionar o valor 1 na coluna seguinte;
  5. Efetuar a subtração do número a ser convertido pela coluna atual ($732_{10} - 512_{10} = 220_{10}$; $220_{10} - 128_{10} = 92_{10}$, ...);
  6. Voltar ao item 2 e repetir até que a conversão do número.

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1024	512	256	128	64	32	16	8	4	2	1
0	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**1**	**0**	0

Mudança de Base - Método Prático

- Ex. 1: Conversão do número 732_{10} para a base binária
  4. Adicionar o valor 1 na coluna seguinte;
  5. Efetuar a subtração do número a ser convertido pela coluna atual ($732_{10} - 512_{10} = 220_{10}$; $220_{10} - 128_{10} = 92_{10}$, ...);
  6. Voltar ao item 2 e repetir até que a conversão do número.

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1024	512	256	128	64	32	16	8	4	2	1
0	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**0**

Mudança de Base - Método Prático

- Ex. 2: Conversão do número $(57,75)_{10}$ para a base binária
  1. Executar o algoritmo anterior separadamente para as partes decimais e fracionárias:

2^6	2^5	2^4	2^3	2^2	2^1	2^0	,	2^{-1}	2^{-2}	2^{-3}
64	32	16	8	4	2	1	,	0.5	0.25	0.125
0	1	1	1	0	0	1	,	1	1	0

Mudança de Base - Método Prático

- Ex. 2: Conversão do número $(57,75)_{10}$ para a base binária
  1. Executar o algoritmo anterior separadamente para as partes decimais e fracionárias:

2^6	2^5	2^4	2^3	2^2	2^1	2^0	,	2^{-1}	2^{-2}	2^{-3}
64	32	16	8	4	2	1	,	0.5	0.25	0.125
0	1	1	1	0	0	1	,	1	1	0

Mudança de Base - Método Prático

- Ex. 2: Conversão do número $(57,75)_{10}$ para a base binária
  1. Executar o algoritmo anterior separadamente para as partes decimais e fracionárias:

2^6	2^5	2^4	2^3	2^2	2^1	2^0	,	2^{-1}	2^{-2}	2^{-3}
64	32	16	8	4	2	1	,	0.5	0.25	0.125
0	**1**	1	1	0	0	1	,	1	1	0

Mudança de Base - Método Prático

- Ex. 2: Conversão do número $(57,75)_{10}$ para a base binária
  1. Executar o algoritmo anterior separadamente para as partes decimais e fracionárias:

2^6	2^5	2^4	2^3	2^2	2^1	2^0	,	2^{-1}	2^{-2}	2^{-3}
64	32	16	8	4	2	1	,	0.5	0.25	0.125
0	**1**	**1**	1	0	0	1	,	1	1	0

Mudança de Base - Método Prático

- Ex. 2: Conversão do número $(57,75)_{10}$ para a base binária
  1. Executar o algoritmo anterior separadamente para as partes decimais e fracionárias:

2^6	2^5	2^4	2^3	2^2	2^1	2^0	,	2^{-1}	2^{-2}	2^{-3}
64	32	16	8	4	2	1	,	0.5	0.25	0.125
0	**1**	**1**	**1**	0	0	1	,	1	1	0

Mudança de Base - Método Prático

- Ex. 2: Conversão do número $(57,75)_{10}$ para a base binária
  1. Executar o algoritmo anterior separadamente para as partes decimais e fracionárias:

2^6	2^5	2^4	2^3	2^2	2^1	2^0	,	2^{-1}	2^{-2}	2^{-3}
64	32	16	8	4	2	1	,	0.5	0.25	0.125
0	**1**	**1**	**1**	**0**	0	1	,	1	1	0

Mudança de Base - Método Prático

- Ex. 2: Conversão do número $(57,75)_{10}$ para a base binária
  1. Executar o algoritmo anterior separadamente para as partes decimais e fracionárias:

2^6	2^5	2^4	2^3	2^2	2^1	2^0	,	2^{-1}	2^{-2}	2^{-3}
64	32	16	8	4	2	1	,	0.5	0.25	0.125
0	**1**	**1**	**1**	**0**	**0**	1	,	1	1	0

Mudança de Base - Método Prático

- Ex. 2: Conversão do número $(57,75)_{10}$ para a base binária
  1. Executar o algoritmo anterior separadamente para as partes decimais e fracionárias:

2^6	2^5	2^4	2^3	2^2	2^1	2^0	,	2^{-1}	2^{-2}	2^{-3}
64	32	16	8	4	2	1	,	0.5	0.25	0.125
0	**1**	**1**	**1**	**0**	**0**	**1**	,	1	1	0

Mudança de Base - Método Prático

- Ex. 2: Conversão do número $(57,75)_{10}$ para a base binária
  1. Executar o algoritmo anterior separadamente para as partes decimais e fracionárias:

2^6	2^5	2^4	2^3	2^2	2^1	2^0	,	2^{-1}	2^{-2}	2^{-3}
64	32	16	8	4	2	1	,	0.5	0.25	0.125
0	**1**	**1**	**1**	**0**	**0**	**1**	,	**1**	1	0

Mudança de Base - Método Prático

- Ex. 2: Conversão do número $(57,75)_{10}$ para a base binária
  1. Executar o algoritmo anterior separadamente para as partes decimais e fracionárias:

2^6	2^5	2^4	2^3	2^2	2^1	2^0	,	2^{-1}	2^{-2}	2^{-3}
64	32	16	8	4	2	1	,	0.5	0.25	0.125
0	**1**	**1**	**1**	**0**	**0**	**1**	,	**1**	**1**	0

Mudança de Base - Método Prático

- Ex. 2: Conversão do número $(57,75)_{10}$ para a base binária
  1. Executar o algoritmo anterior separadamente para as partes decimais e fracionárias:

2^6	2^5	2^4	2^3	2^2	2^1	2^0	,	2^{-1}	2^{-2}	2^{-3}
64	32	16	8	4	2	1	,	0.5	0.25	0.125
0	**1**	**1**	**1**	**0**	**0**	**1**	,	**1**	**1**	**0**

Mudança de Base - Problemas

- Ex. 3: Comparação da conversão dos números $(57,75)_{10}$ e $(57,76)_{10}$ para a base binária

$$57,75 = 111001,11$$
$$57,76 = 111001,110000101$$

- Observe que o número $(57,76)_{10}$ precisa ser arredondado para ser representado em binário

$$57,76 = (57),(0,5 + 0,25 + 0,0078125 + 0,001953125)$$
$$57,76 \neq 57,759765625$$

Mudança de Base - Problemas

- Ex. 3: Comparação da conversão dos números $(57,75)_{10}$ e $(57,76)_{10}$ para a base binária

$$57,75 = 111001,11$$
$$57,76 = 111001,110000101$$

- Observe que o número $(57,76)_{10}$ precisa ser arredondado para ser representado em binário

$$57,76 = (57),(0,5 + 0,25 + 0,0078125 + 0,001953125)$$
$$57,76 \neq 57,759765625$$

Mudança de Base - Problemas

- Computadores são dispositivos discretos, com limitação de tamanho de palavras (*bits*);
- Alguns números decimais não tem representação finita em base binária;
  - Esses números estão sujeitos a limitação da arquitetura dos computadores (*bits* disponíveis para armazenamento);
  - Alguns números, então, são armazenados de forma aproximada;
  - A aproximação pode causar erros inaceitáveis, quando diversas operações são executadas em sequência.

Notações numéricas

Notações numéricas

- Um número x na base β é representado
  - No sistema posicional:

$$x = \left(a_m a_{(m-1)} \ldots a_1 a_0 \ , \ a_{(-1)} a_{(-2)} a_{(-3)} \ldots\right)_\beta$$

  - Em sua forma polinomial:

$$x = a_n\beta^m + a_{n-1}\beta^{m-1} + \ldots + a_1\beta^1 + a_0\beta^0 \ , \ a_1\beta^{-1} + a_2\beta^{-1} + \ldots + a_n\beta^{-n}$$

Notações numéricas

- Um número x na base β é representado
  - Em notação científica:

$$x = \pm(M)_\beta \times \beta^E$$

onde $(M)_\beta = \left(a_m a_{(m-1)} \ldots a_1 a_0 \ , \ a_{(-1)} a_{(-2)} a_{(-3)} \ldots\right)_\beta$ é chamado de **mantissa** e $E \in \mathbb{Z}$[1] é chamado de **expoente** de x.

  - Em notação (científica) normalizada:

$$x = (-1)^s \times (M)_\beta \times \beta^E$$

onde $(M)_\beta = \left(a_0 \ , \ a_{(-1)} a_{(-2)} a_{(-3)} \ldots\right)_\beta$, com $a_0 \neq 0$, $s = 0$ para positivo e $s = 1$ para negativo.

[1] Na literatura é comumente utilizado o símbolo e para representação do expoente

Representação Numérica

Representação Inteira

- Um número inteiro é armazenado como uma sequência de dígitos binários de tamanho fixo, denominado registro [Justo et al., 2020].
  - Representação sem *bit* de sinal

a_{n-1}	a_{n-2}	...	a_1	a_0

  - Representação com um *bit* de sinal
    - O bit mais significativo (à esquerda) representa o sinal: por convenção, 0 significa positivo e 1 significa negativo.

s	a_{n-1}	a_{n-2}	...	a_1	a_0

Complemento de 2

- Um número inteiro pode ser armazenado com o bit mais significativo representando o coeficiente de -2^{n-1} [Justo et al., 2020].

$a_{n-1} \times (-2^{n-1})$	a_{m-2}	...	a_1	a_0

  - Todo número iniciado em 1 representa um número negativo
  - Ex. 1: Número $(01000011)_2$

$$-0(2^7) + (1000011)_2 = 2^6 + 2^1 + 2^0 = 67$$

  - Ex. 2: Número $(10111101)_2$

$$-1(2^7) + (0111101)_2 = -2^7 + 2^5 + 2^4 + 2^3 + 2^2 + 2^0 = -67$$

Complemento de 2

- Vantagens:
  - As operações aritméticas de soma, substração e multiplicação, usando esse protocolo, são idênticas às operações com números sem *bit* de sinal;
  - Casos de *overflow* são facilmente descartados;
  - Facilidade na implementação de sistemas, especialmente para números de alta precisão.

Sistema de Ponto Fixo

INSTITUTO FEDERAL
Minas Gerais

- No sistema de ponto fixo, as partes inteira e fracionária são representadas com uma quantidade fixa de dígitos.

Sistema de Ponto Flutuante

INSTITUTO FEDERAL
Minas Gerais

- O sistema de ponto flutuante não possui quantidade fixa de dígitos para as partes inteira e fracionária do número
  - Possuem notação (científica) normalizada: a mantissa (ou significando) é um valor entre 0 e 1;

Sistema de Ponto Flutuante

- É dividido em 3 partes: sinal, mantissa e expoente

$$\boxed{x = (-1)^s \times (M) \times \beta^E}$$

  - Mantissa:

$$M = (d_1 d_2 d_3 ... d_t) \quad \text{ou} \quad M = \left[\frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \frac{d_3}{\beta^3} + ... + \frac{d_t}{\beta^t} \right]$$

  - d_i: i-ésimo dígito da mantissa, onde $0 \leq d_i \leq (\beta - 1)$;
  - t: número de dígitos (depende do tamanho da palavra);
  - E: expoente inteiro

Sistema de Ponto Flutuante

INSTITUTO FEDERAL
Minas Gerais

- Sistema de ponto flutuante também pode ser definido como:

$$F(\beta, |M|, |E|, BIAS) \quad \text{ou} \quad F(\beta, |M|, E_{MIN}, E_{MAX})$$

  - β: base;
  - M: mantissa (ou significando);
  - E: expoente inteiro;
  - $BIAS$: valor de deslocamento do expoente (dependente da precisão).
  - E_{MIN}: menor expoente;
  - E_{MAX}: maior inteiro;

Sistema de Ponto Flutuante

- O número normal pode ser representado por

$$\boxed{x = (-1)^s \times M \times 2^{c-BIAS}}$$

  - M: mantissa, definida por

$$M = (1\ .\ m_1 m_2 ... m_N)_2$$

  - c: característica, definida por

$$c = (c_n c_{n-1} ... c_1 c_0)^2 = c_n 2^n + c_{n-1} 2^{n-1} + ... + c_1 2^1 + c_0 2^0$$

  - N: corresponde ao número de dígitos disponíveis da mantissa;
  - n: corresponde ao número de dígitos disponíveis do expoente.

Sistema de Ponto Flutuante

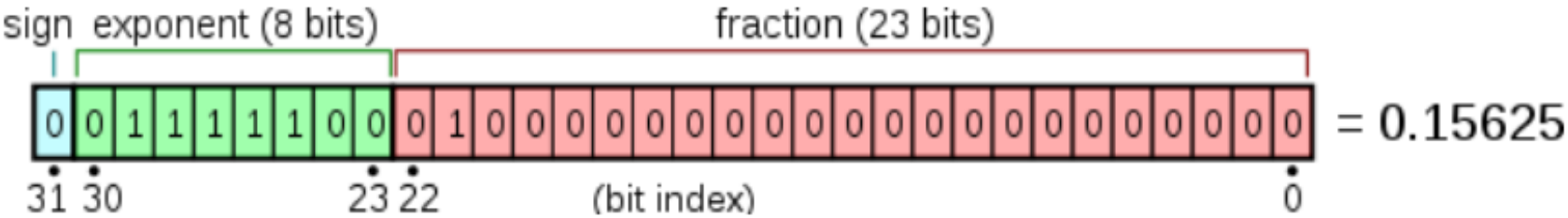
INSTITUTO FEDERAL
Minas Gerais

- O número de dígitos N, da mantissa, e n, do expoente, devem ser obtidos na tabela, de acordo com a precisão desejada;
- O valor de $BIAS$ também deve ser recuperado da tabela.

Tipo	Sinal	Exponente	Mantissa	Total bits		Expoente bias	Bits precisão
Half	1	5	10	16		15	11
Single	1	8	23	32		127	24
Double	1	11	52	64		1023	53
x86 ext. precision	1	15	64	80		16383	64
Quad	1	15	112	128		16383	113

Fonte: [Wikipedia contributors, 2020]

Sistema de Ponto Flutuante



Fonte: [Wikipedia contributors, 2020]

- Expoente (8 *bits*):

$$2^6 + 2^5 + 2^4 + 2^3 + 2^2$$

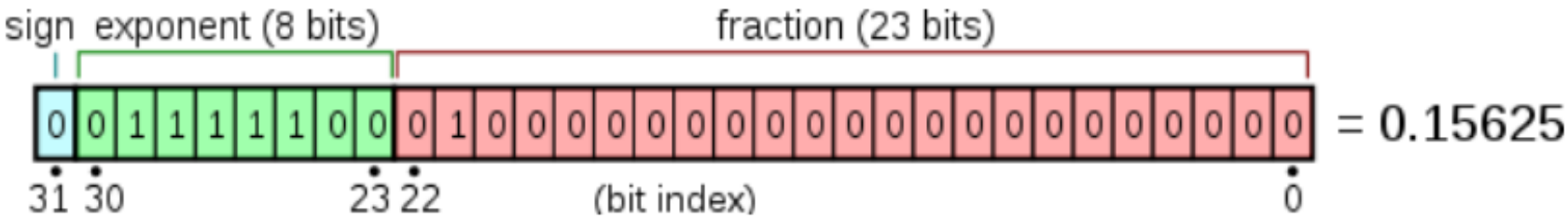
$$64 + 32 + 16 + 8 + 4 = 124$$

- Mantissa (23 *bits*):

$$1 + 2^{-2}$$

$$1 + 0.25 = 1.25$$

Sistema de Ponto Flutuante



Fonte: [Wikipedia contributors, 2020]

$$x = (-1)^s \times M \times 2^{c-BIAS}$$

$$x = (-1)^0 \times 1.25 \times 2^{124-127}$$

$$x = 1 \times 1.25 \times 2^{-3}$$

$$x = 1.25 \times 0.125$$

$$x = 0.15625$$

Casos Especiais

- Expoentes reservados são usados para casos especiais:

$$[0|\mathbf{00000000}|00000000000000000000000]$$

  - $c = [X|00000000|XX...]$ é usado para representar o zero, se $m = 0$, e números subnormais, se $m \neq 0$;
    - Números subnormais (ou denormais) são aqueles que possuem zero na mantissa, não podendo ser representados por expoentes;
  - $c = [X|11111111|XX...]$ é usado para representar o infinito, se $m = 0$, e NaN, se $m \neq 0$.

Casos Especiais

- Um número subnormal deve utilizar E_{MIN}, que varia de acordo com a precisão desejada;
  - Para 16 bits, $E_{MIN} = -14$
  - Para 32 bits, $E_{MIN} = -126$
  - Para 64 bits, $E_{MIN} = -1023$
- Nesses casos, a mantissa é definida por

$$M = (0 \ . \ m_1 m_2 ... m_N)_2$$

IEEE 754

INSTITUTO FEDERAL
Minas Gerais

- Padrão estabelecido em 1985 para definição da representação de números em ponto flutuante;
- Criado devido a variedade de implementações, o que dificultava a confiabilidade e portabilidade.

Propriedade	Prec. Simples	Prec. Dupla	Prec. Estendida
Comprimento	32	64	80
Mantissa (bits)	23	52	64
Expoente (bits)	8	11	15
Base	Binária	Binária	Binária
Dígitos Decimais	7	16	19
Maior Expoente	+127	+1023	+16383
Menor Expoente	-126	-1022	-16382
Maior Número	$\approx 3.40 \times 10^{+38}$	$\approx 1.80 \times 10^{+308}$	$\approx 1.19 \times 10^{+4932}$
Menor Número	$\approx 1.18 \times 10^{-38}$	$\approx 2.23 \times 10^{-308}$	$\approx 3.36 \times 10^{-4932}$
C	`float`	`double`	`long double`
Pascal	`real, single`	`double`	`extended`

Fonte: [da Silva, 2020]

Épsilon (ϵ)

- A precisão p de uma máquina é o número de dígitos significativos usado para representar um número [Justo et al., 2020];
- Devido a arredondamentos, dois números extremamente pequenos podem ser tratados como iguais;
- O número épsilon, ϵ, é o menor número $\epsilon > 0$, em que

$$\boxed{(1 + \epsilon) \neq 1}$$

- A multiplicação por épsilon evita que operações em sequência sejam zeradas.

Overflow e Underflow

- Cálculos podem resultar em condições no qual o resultado é muito grande ou muito pequeno para ser representado pelo computador
  - Underflow: número muito pequeno, próximo a zero;
  - Overflow: número muito grande, resultando em infinito;
- Essa situação pode ocasionar erros, principalmente quando há uma sequência de operações.

Erros

Tipos de Erros

- Em aproximações numéricas, as fontes mais comuns de erros são:
  - Modelagem incorreta;
  - Incerteza de dados;
  - Erros de arredondamento;
  - Erros de truncamento;
- As medidas de erros mais comuns são:
  - Erro relativo;
  - Erro absoluto;

Modelagem Incorreta

- O modelo adotado não reflete o fenômeno com perfeição;
- Simplificações necessárias à confecção do modelo causam exclusão de informações preciosas, transformando o modelo em uma versão aproximada;

Incerteza dos dados

- Incerteza de dados está relacionado a erros causados devido aos dados de entrada;
  - Em modelagem de problemas físicos, a imprecisão é inerente aos equipamentos de medição;
  - A acurácia de um instrumento é sempre finita, por mais preciso que esse seja;
  - Não é possível evitar que erros de medição ocorram;

Erros de Arredondamento

INSTITUTO FEDERAL
Minas Gerais

- Erros relacionados com as limitações existentes na forma de representar números em máquina;
  - Depende da precisão: simples, dupla, estendida, etc.;

Erros de Arredondamento

- Ex. 1: Consideremos o menor número positivo não zero normal, em uma arquitetura de 16 bits

s	expoente					mantissa									
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

$$x = (-1)^s \times M \times 2^{c-BIAS}$$

$$x = (-1)^0 \times (1+0) \times 2^{1-15}$$

$$x = 1 \times (1+0) \times 2^{-14}$$

$$x = 1.0 \times 0.000061035$$

$$x = 0.000061035$$

$$x = 6.1035 \times 10^{-5}$$

$$x = 6.1035e-05$$

Erros de Arredondamento

- Ex. 2: Consideremos o menor número positivo não zero subnormal, em uma arquitetura de 16 bits
  - Lembrar que o cálculo de números denormais é ligeiramente diferente

s	expoente					mantissa									
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

$$x = (-1)^s \times M_{SUB} \times 2^{E_{MIN}} \quad \text{, onde} \quad M_{SUB} = (0\ .\ m_1 m_2 ... m_N)_2$$

$$x = (-1)^0 \times 2^{-10} \times 2^{1-14}$$

$$x = 1 \times (0 + 2^{-10}) \times 2^{-14}$$

$$x = 0.0009765625 \times 0.000061035$$

$$x = 0.000000059604645$$

$$x = 5.9604645 \times 10^{-8}$$

Erros de Truncamento

- Erros de truncamento ocorrem quando há substituição de um conceito matemático formado por uma sequência infinita de passos por um de procedimento finito [Justo et al., 2020];
- Devido à aproximação causada pelo uso de fórmulas;
  - Ex. 1: A definição de integral é dada por um processo de limite de somas. Numericamente, ela é calculada por uma soma finita;
  - Ex. 2: Cálculo de e, usando a aproximação $f(x) = e^x$, pela Série de Taylor

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ... + \frac{x^n}{n!}$$

- Quanto maior o valor de n, menor o erro de truncamento.

Erro Absoluto

- O erro absoluto EA_x corresponde ao módulo da diferença entre o valor exato do número x e seu valor aproximado $\overline{x}$;

$$\boxed{EA_x = |x - \overline{x}|}$$

- Muitas vezes, o valor exato é desconhecido, existindo somente o valor aproximado.
  - Ex. 1: Consideremos o valor de $\pi \in (3.14, 3.15)$
  - O erro, nesse caso, é dado por $EA_\pi = |\pi - \overline{\pi}| < 0,01$.

Erro Relativo

- O erro relativo ER_x corresponde a razão entre o erro absoluto EA_x e o valor aproximado $\overline{x}$;

$$ER_x = \frac{|x-\overline{x}|}{|x|} \quad , x \neq 0$$

- Frequentemente o erro relativo (adimensional) é expresso em percentual

$$ER_x \times 100\%$$

Erro Absoluto e Relativo

- Ex. 1: Consideremos dois números $\overline{y} = 2112.9$ e $\overline{z} = 5.3$, com respectivos erros $EA_y < 0.1$ e $EA_z < 0.1$
  - A precisão é a mesma, uma vez que o erro absoluto é igual?

$$ER_y < \frac{|0.01|}{|2112.9|} \approx 4.7 \times 10^{-5} \quad \text{e} \quad ER_z < \frac{|0.01|}{|5.3|} \approx 2 \times 10^{-2}$$

  - Como $|ER_y| < |ER_z|$, o número y é representado com maior precisão do que z.

Referências I

da Silva, D. M. (2020).
Cálculo Numérico - Slides de Aula.
IFMG - Instituto Federal de Minas Gerais, Campus Formiga.

Justo, D., Sauter, E., Azevedo, F., Guidi, L., and Konzen, P. H. (2020).
Cálculo Numérico, Um Livro Colaborativo - Versão Python.
UFRGS - Universidade Federal do Rio Grande do Sul.
https://www.ufrgs.br/reamat/CalculoNumerico/livro-py/livro-py.pdf.

Wikipedia contributors (2020).
Floating-point arithmetic.
[Online]; acessado em 16 de Julho de 2020.