Aritmética de computador

Representação de números em diferentes bases

Números inteiros vs. Números reais

- É comum para a maioria dos computadores, o uso de uma base numérica distinta da base decimal
- Em geral, os números são armazenados na
  - **base 2** (**binária**) – mais comum
  - **base 8** (**octal**)
  - **base 16** (**hexadecimal**)
- Um número pode ser representado usando
  - o sistema posicional
  - a forma polinomial

Números inteiros vs. Números reais

- A representação de **números inteiros** é ligeiramente **distinta** da representação de **números reais**
- Representação de números inteiros segue um formato
  - de Sinal-Magnitude
- Representação de números reais seguem dois formatos
  - de ponto fixo
  - de ponto flutuante

Números inteiros

- Um número inteiro **N** é representado, numa **base b**, por uma sequência de dígitos $\mathbf{a_i}$ em que
  - $\mathbf{a_i} \in \{ 0, 1, \ldots, b-1 \}$
  - **i** assume um intervalo de valores que depende da base em uso
- Bases mais utilizadas

base	$\mathbf{a_i}$
2	0, 1
8	0, 1, 2, 3, 4, 5, 6, 7
10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Números inteiros

- Representação em formato de **Sinal-Magnitude**
  - no sistema posicional
  - na forma polinomial
- No **sistema posicional**
  - os dígitos são **agrupados** numa **sequência**
  - a dimensão da contribuição de cada dígito no número depende da posição que ocupa
  - um número **N** é escrito com o seguinte formato:

$$\mathbf{N = (a_n\ a_{n-1}\ \ldots\ a_1\ a_0)_b}$$

- Na **forma polinomial**
  - fica claramente explicitada a contribuição de cada dígito para o valor de N
  - um número N é escrito com o seguinte formato:

$$\mathbf{N = a_n\ b^n + a_{n-1}\ b^{n-1} + \ldots + a_1\ b + a_0}$$

Números reais

- Um número pode ser representado usando dois formatos
  - **de ponto fixo** (por exemplo, 12.34)
  - **de ponto flutuante** (por exemplo, 0.4273×10^2)
- Representação em formato **de ponto fixo**
  - um número real **X** é composto por duas partes:
    - a parte inteira $\mathbf{X_i}$ (número inteiro)
    - a parte fracionaria $\mathbf{X_f}$ (número real), em que $\mathbf{X_f} = X - \mathbf{X_i}$
  - exemplo:

$X = 12.34$

$X_i = 12$

$X_f = 0.34$

Números reais

- Representação em formato **de ponto flutuante**
  - a notação é semelhante à notação científica:

$\pm .d_1 d_2 d_3 \ldots d_p \times b^e$,

em que

- $\mathbf{d_k}$ (k = 1,..., p) são os dígitos da parte fracionária com
  - $d_k \in \{ 0, \ldots, b-1 \}$
  - $d_1 \neq 0$ (se o número for normalizado)
- **b** é o valor da base (geralmente 2, 8, 10 ou 16)
- **p** é o número de dígitos da parte fracionária
- **e** é um expoente inteiro

Números reais

- Representação em formato **de ponto flutuante**
  - um **número real** é composto por três partes:
    - o **sinal**
    - a **parte fracionária** (**significando** ou **mantissa**)
    - o **expoente**
  - as três partes tem um comprimento total **fixo** que **depende**
    - do **computador** e
    - do **tipo de número** (precisão **simples**, **dupla** ou **estendida**)

Números reais

- Representação em formato **de ponto flutuante**
  - exemplo (na base decimal):

X = 0.4273×10^3

- representado por:

  $d_1\ d_2\ d_3\ d_4 = 4273$

  b = 10

  p = 4

  e = 3

- composto por:

  sinal: + (omitido)

  mantissa: 4273

  expoente: 3

Números reais

- Representação em computador
  - a representação de um número pode ser diferente entre os fabricantes do computadores
  - ou seja, o **mesmo programa** implementado em computadores que utilizam **formatos diferentes** pode fornecer **resultados diferentes**

Números reais

- Representação em computador
  - formato utilizado pela maioria dos computadores é padrão IEEE 754 (para binários)

Propriedades	Precisão		
	Simples	**Dupla**	**Estendida**
Comprimento total	32	64	80
bits na mantissa	23	52	64
bits no expoente	8	11	15
sinal	1	1	1
expoente máximo	127	1023	16383
expoente mínimo	-126	-1022	-16382
maior número (absoluto)	$\approx 3.40 \times 10^{38}$	$\approx 1.80 \times 10^{308}$	$\approx 1.19 \times 10^{4932}$
menor número (absoluto)	$\approx 1.18 \times 10^{-38}$	$\approx 2.123 \times 10^{-308}$	$\approx 3.36 \times 10^{-4932}$
dígitos decimais (precisão)	7	16	19

Aritmética de ponto flutuante

Operações com números

- **OVERFLOW**
  - ocorre quando uma operação aritmética resultar num **número** que seja **maior**, em **valor absoluto**, que o **maior número representável**
- **UNDERFLOW**
  - ocorre quando uma operação aritmética resultar num **número** que seja **menor**, em **valor absoluto**, que o **menor número representável diferente de zero**

Operações com números em decimal

- Seja um hipotético computador com
  - dois dígitos da mantissa (p = 2)
  - base b = 10
  - expoente e $\in$ {-5, …, 5}
  - formato: $\pm .\mathbf{d_1 d_2 \times 10^e}$
- Quando dois números são **somados** ou **subtraídos**, o processo é o que se segue
  - os dígitos do número de menor expoente são deslocados para alinhar as casas decimais
  - o resultado é então normalizado (o expoente é ajustado de forma a normalizar a mantissa, $d_1 \neq 0$)
  - por fim, o resultado é arredondado para dois dígitos para caber na mantissa (p = 2).

Operações com números em decimal

- Exemplo 1: **4.32 + 0.064**
  - os números são armazenados no formato especificado,
  - as casas decimais são alinhadas,
  - a operação de adição é efetuada;
  - por fim, o resultado é então normalizado e arredondado para dois dígitos:

$$
\begin{aligned}
4.32 + 0.064 = .43 \times 10^{1} + .64 \times 10^{-1} &= .43 \times 10^{1} \\
&+ .0064 \times 10^{1} \\
&= .4364 \times 10^{1} \\
&\rightarrow .44 \times 10^{1}
\end{aligned}
$$

  - o resultado da adição é **4.4** em vez de **4.384**

Operações com números em decimal

- Exemplo 2: **372 – 371**
  - os números são armazenados no formato especificado
  - as casas decimais são alinhadas
  - a operação de adição é efetuada
  - o resultado é então normalizado e arredondado para dois dígitos:

$$
\begin{aligned}
372 - 371 = .37 \times 10^3 - .37 \times 10^3 &= .37 \times 10^3 \\
&\underline{\quad - \quad .37 \times 10^3} \\
&= .00 \times 10^3 \\
&\rightarrow .00 \times 10^0
\end{aligned}
$$

  - o resultado da subtração é **0** em vez de **1**

Operações com números em decimal

- Exemplo 3: **1234 x 0.016**
  - os números são armazenados no formato especificado,
  - a multiplicação é efetuada utilizando 2p = 4 dígitos na mantissa;
  - o resultado é então normalizado e arredondado para dois dígitos:

$$
\begin{aligned}
1234 \times 0.016 = .12 \times 10^{4} \times .16 \times 10^{-1} &= .12 \times 10^{4} \\
&\quad \times .16 \times 10^{-1} \\
&= .0192 \times 10^{3} \\
&\rightarrow .19 \times 10^{2}
\end{aligned}
$$

  - o resultado da multiplicação é **19** em vez de **19.744**

Operações com números em decimal

- A perda de precisão quando dois números aproximadamente iguais são subtraídos é das maiores fontes de erro nas operações de ponto flutuante
- Uma das causas de se cometer erros quando se usa um computador deve-se à conversão de base, pois um número
  - é fornecido ao computador em decimal (base 10), e
  - é armazenado em binário (base 2).
- Enquanto que
  - para um número inteiro, a representação é exata; por exemplo, $44_{10} = 101100_2$
  - para um número real com parte fracionária, a representação a parte fracionária tem que ser arredondada para ser armazenada em formato de ponto flutuante

Representação de números em computadores digitais

Representação de inteiros e reais

- As **representações** apresentadas antes **não são suficientes**, pois é necessário distinguir-se, por exemplo, o **sinal do número**
- Para se representar o sinal **+** ou **-** na memória de um computador, acrescenta-se um **bit** ao número para representar o sinal — o denominado de **bit de sinal**
- Representação de inteiros
  - em Sinal-Módulo (ou Sinal-Magnitude)
  - em Complemento à base (a **b** e a **b-1**)
- Representação de números reais
  - em formato de ponto flutuante normalizado

Representação de números inteiros em Sinal-Módulo

- A representação mais direta é a em **Sinal-Módulo**, onde
  - o valor absoluto do número é obtido diretamente a partir de algoritmos próprios, e
  - o sinal é representado por um dígito adicional colocado à esquerda do número
- Na **representação binária** o **bit de sinal** é o bit mais significativo (posicionado mais à esquerda na sequência)
- Supondo que o computador dispõe de **q** dígitos para a representação,
  - um inteiro em binário é representado no computador através da seguinte sequência de dígitos (**palavra**):

$\mathbf{a_{q-1}\ a_{q-2}\ \ldots\ a_1\ a_0}$

em que

$a_0, a_1, \ldots, a_{q-2} \in \{ 0, 1 \}$ e

$a_{q-1} \in \{ 0, 1 \}$ representa o **sinal do número** (0 para "+" e 1 para "−")

Representação de números inteiros em Sinal-Módulo

- A conversão para o sistema decimal de um número em binário representado internamente por $\mathbf{a_{q-1}\ a_{q-2}\ \ldots a_1\ a_0}$, é feita através de uma fórmula semelhante à forma polinomial

$$N = (-1)^{a_{q-1}} \times \sum_{k=0}^{q-2} \left(a_k \times 2^k \right),$$

em que,

N o número inteiro na base decimal

q-2 é o índice do dígito mais à esquerda que representa o valor absoluto de **N**

$\mathbf{a_k}$ um dígito válido na representação ($a_k \in \{ 0, 1 \}$), k = 0, 1, …, q-2

$\mathbf{a_{q-1}} \in \{ 0, 1 \}$ e representa o bit de sinal

- Os valores para as quantidades expressas dependem da arquitetura e do compilador utilizado

Representação de números inteiros em Sinal-Módulo

- Por exemplo, um dado compilador possui 4 modelos de representação de inteiros com **1, 2, 4 e 8 bytes**, também denominados de **espécies**
- Para o caso binário, os maiores números em valor absoluto que podem ser representados internamente para cada espécie $\mathbf{N_{max}}(p)$ (p = 1, 2, 4, 8) são

$$N_{max}(p) = 2^0 + 2^1 + 2^2 + \ldots + 2^{8p-2} = 2^{8p-1} - 1 ,$$

p = 1 (1 byte = 8 bites), em que os últimos 7 bites são 7 **1**'s (**1111111**)

$\mathbf{N_{max}} = \mathbf{1111111}_2 = \mathbf{127}_{10}$ ($\mathbf{1}x2^0 + \mathbf{1}x2^1 + \mathbf{1}x2^2 + \mathbf{1}x2^3 + \mathbf{1}x2^4 + \mathbf{1}x2^5 + \mathbf{1}x2^6$)

p = 2 (2 bytes = 16 bites), em que os últimos 15 bites são 15 **1**'s

$\mathbf{N_{max}} = \mathbf{1\ldots1}_2 = \mathbf{32767}_{10}$ ($\mathbf{1}x2^0 + \mathbf{1}x2^1 + \mathbf{1}x2^2 + \cdots + \mathbf{1}x2^{12} + \mathbf{1}x2^{13} + \mathbf{1}x2^{14}$)

p = 4 (4 bytes = 32 bites), em que os últimos 31 bites são 31 **1**'s

$\mathbf{N_{max}} = \mathbf{1\ldots1}_2 = \mathbf{2147483647}_{10}$ ($\mathbf{1}x2^0 + \mathbf{1}x2^1 + \mathbf{1}x2^2 + \cdots + \mathbf{1}x2^{29} + \mathbf{1}x2^{30}$)

p = 8 (8 bytes = 64 bites), em que os últimos 63 bites são 63 **1**'s

$\mathbf{N_{max}} = \mathbf{1\ldots1}_2 = \mathbf{9223372036854775807}_{10}$ ($\mathbf{1}x2^0 + \mathbf{1}x2^1 + \cdots + \mathbf{1}x2^{61} + \mathbf{1}x2^{62}$)

Representação de números reais

- Representação de números reais em computadores denomina-se por
  - representação em formato de ponto flutuante normalizado
- Nesta representação um número é representado internamente através de uma notação científica

- um bit de sinal **s** (interpretado como positivo ou negativo)
- um expoente inteiro exato **e**
- uma mantissa inteira positiva **M**

sendo que apenas um número limitado de dígitos é permitido para **e** e **M**

Representação de números reais

- Tomando todas estas quantidades juntas, estas representam o número

$$x = s \times (0.d_1 d_2 \ldots d_n) \times b^e,$$

em que,

- **s** é o sinal do número,
- os dígitos **d_1, d_2, ..., d_n**
  - formam a mantissa **M** = $0.d_1d_2...d_n$ e
  - são limitados pela base **b** ($0 \leq d_i \leq b-1$, i = 1, ..., n e $d_1 \neq 0$),
- o expoente **e** é limitado ao conjunto { e_{min}, ..., e_{max} },
- $n \geq 1$
  - denomina-se de número de dígitos do sistema e
  - define o tamanho da mantissa M

- O valor zero não pode ser normalizado e tem representação especial
  - com mantissa nula (todos dígitos iguais a zero) e
  - expoente o menor possível

Representação de números reais

- O conjunto formado por zero e todos os números em formato de ponto flutuante
  - chamado-se **Sistema de Ponto Flutuante na base b** com **n** algarismos significativos
  - denota-se por **F(b, n, e_{min}, e_{max})**
- Um computador só pode representar os valores de **e** e **M** com dígitos da base **b**
- Um computador digital (b = 2) dispõe sempre de um tamanho de palavra finito
- Para um dado tipo de números reais, é sempre fixo o **número total de bits** que podem ser utilizados para representar
  - o sinal **s** (1 bit)
  - o expoente
  - a mantissa
- Exemplo de um número real de precisão simples que é representado por 32 bits
  - **1 bit** é utilizado para representar o **sinal**
  - **8 bits** são utilizados para representar o **expoente**
  - os restantes **23 bits** para representar a **mantissa**

Representação de números reais

- Um número real será representado na memória do computador como

$x = s\, e_7 e_6 \ldots e_1 e_0\, d_1 d_2 \ldots d_{22} d_{23}$

em que

$s, e_0, \ldots, e_7, d_1, \ldots, d_{23} \in \{0, 1\}$.

- Exemplo

Considere dois números binários com 8 algarismos significativos em F(2, 8, -3, 4):

$n_1 = 0\ 010\ 11100110_2 \Rightarrow (-1)^0 \times 2^2 \times (0.11100110) = 3.59375_{10}$

$n_2 = 0\ 010\ 11100111_2 \Rightarrow (-1)^0 \times 2^2 \times (0.11100111) = 3.609375_{10}$

- observe-se que, no sistema de representação utilizado,
  - n_1 e n_2 são dois números consecutivos (não há representação de número entre eles)
- portanto, por exemplo, o número 3.60000
  - não tem representação exata neste sistema, sendo representada por n_1 ou n_2
  - o que vai gerar um erro, denominado **Erro de Arredondamento**

Representação de números reais

- Portanto
  - os números reais podem ser representados por uma reta contínua
  - em formato de ponto flutuante só se podem representar pontos discretos da reta real
- Conversão de um número **x** representado na base **b** para a base decimal

$$x = (-1)^s \times b^e \times \sum_{k=1}^{n} \left(d_k \times b^{-k} \right)$$

- A tabela mostra alguns números para um computador que usa o padrão IEEE 754

Espécie	REAL (4)	REAL (8)	REAL (10)
n	23	52	64
e_{min}	-126	-1022	-16382
e_{max}	127	1023	16383
X_{min}	$1.1754944 \times 10^{-38}$	$2.225073858507201 \times 10^{-308}$	$3.3621031431120935 06... \times 10^{-4932}$
X_{max}	3.4028235×10^{38}	$1.797693134862316 \times 10^{308}$	$1.1897314953572317 65... \times 10^{4932}$
X_{eps}	1.1920929×10^{-7}	$2.220446049250313 \times 10^{-16}$	$1.9259299443872358 53... \times 10^{-34}$

Representação de números reais no sistema normalizado

- No **sistema binário** (base = 2) **normalizado** ($d_1 \neq 0$)
  - o primeiro dígito da mantissa no sistema normalizado é sempre igual a 1, e por esta razão não é armazenado (é o denominado **bit escondido**)
  - esta normalização permite um ganho na precisão, pois pode-se considerar que a mantissa é armazenada em 24 bits

Representação de números reais no sistema normalizado

- Os números do sistema F(b, n, e_{min}, e_{max}) têm as seguintes características
  - o menor número positivo que pode ser representado neste sistema é

    $x_{min} = 0.1 \times b^{e_{min}}$

    o que significa que qualquer número real **x** tal que

    $-x_{min} < \mathbf{x} < x_{min}$

    não poderá ser representado pelo computador — ocorre **underflow**
  - os compiladores, quando detetam **underflow**, são instruídos normalmente para
    - terminar o processamento neste ponto, disparando uma mensagem de erro, ou
    - seguir o processamento arredondando **x** para **0**

Representação de números reais

- Os números do sistema F(b, n, e_{min}, e_{max}) têm as seguintes características
  - o maior número positivo que pode ser representado neste sistema é

$$x_{max} = 0.\underbrace{(b-1)(b-1)\ldots(b-1)}_{\text{n vezes}} \times b^{e_{max}} = (b-1) \times \left(\sum_{k=1}^{n} b^{-k} \right) \times b^{e_{max}} = \left(1 - b^{-n}\right) b^{e_{max}}$$

  - isto significa que qualquer número **x** tal que

$$\mathbf{x} < -x_{max} \quad \text{ou} \quad \mathbf{x} > x_{max}$$

    não poderá ser representado pelo computador — ocorre **overflow**
  - os compiladores quando detetam **overflow**, são instruídos normalmente para
    - parar o processamento do programa emitindo uma mensagem de erro, ou
    - continuar o processamento atribuindo a **x** o valor simbólico **-Infinito** ou **Infinito**

Representação de números reais

- Os números do sistema F(b, n, e_{min}, e_{max}) têm as seguintes características
  - o maior número que pode ser somado ou subtraído a **1.0**, em que o resultado permanece inalterado (i.é, a diferença entre **1.0** e o número que lhe sucede em F), é

$$x_{eps} = 0.\underbrace{10\ldots01}_{\text{n vezes}} \times b^1 - 0.\underbrace{10\ldots00}_{\text{n vezes}} \times b^1 = b^{1-n}$$

  em que $\mathbf{x_{eps}}$ é denominada de **epsilon da máquina**, ϵ, ou de **precisão da máquina**
  - o epsilon da máquina, ϵ, também pode ser definido como o menor número em formato de ponto flutuante, tal que:

$$1 + \epsilon > 1$$

  - esta quantidade é da maior **importância** na análise de **erros de arredondamento**

Representação de números reais

- Duma forma mais geral, para um número em formato de ponto flutuante $x \in F$

ulp(x) = $(0.00...01)_b \times b^e = b^{-n} \times b^e = \varepsilon \times b^e$

em que **ulp** é a abreviatura para **unit in the last place**

- se $x > 0$, então **ulp(x)** é a distância entre **x** e o número que lhe sucede em F
- se $x < 0$, então **ulp(x)** é a distância entre **x** e o número que o antecede em F

- Apenas um conjunto finito R_F de números racionais podem ser representados na forma apresentada
  - estes números denominam-se de **números de ponto flutuante**.
- Numa representação normalizada ($d_1 \neq 0$), a quantidade de números racionais que este conjunto contém é precisamente

$2 (b - 1) b^{n-1} (e_{max} - e_{min} + 1) + 1$

Representação de números reais

- Exemplo

Considere-se o sistema de representação numérica F(2, 4, -5, 6)

Menor número positivo possível:

$\mathbf{x_{min}} = (0.1000)_2 \times 2^{-5} = 2^{-5-1} = 2^{-6} = \mathbf{1/64} = \mathbf{0.015625}$

assim, a região de **underflow** consiste no intervalo

$-0.015625 < \mathbf{x} < 0.015625$

Maior número positivo possível:

$\mathbf{x_{max}} = (0.1111)_2 \times 2^{6} = (1 - 2^{-4}) \times 2^{6} = \mathbf{60}$

assim, as regiões de overflow consistem nos intervalos

$\mathbf{x} < -60$ e $\mathbf{x} > 60$

Maior número que pode ser somado/subtraído a **1.0** mantendo o resultado inalterado:

$\mathbf{x_{eps}} = 2^{1-4} = 2^{-3} = \mathbf{1/8} = \mathbf{0.125}$

Número de elementos em R_F:

$2 \times 1 \times (6 + 5 + 1) \times 2^{4-1} + 1 = \mathbf{193}$