Representação dos Números Reais

- Algumas das propriedades básicas da aritmética real não valem mais quando executadas no computador, pois, enquanto na matemática alguns números são representados por infinitos dígitos, no computador isso não é possível, pois uma palavra de memória é finita e a própria memória também.
- Exemplos:

$$\sqrt{2} \qquad \sqrt{3} \qquad \pi \qquad \frac{1}{3}$$

Representação dos Números Reais

- Se desejássemos calcular a área de uma circunferência de raio 100m, podemos obter os seguintes resultados:

$$A = \pi * r^2$$

  a) A = 31400 m^2

  b) A = 31416 m^2

  c) A = 31415.92654 m^2

- **Como justificar as diferenças entre os resultados?**
- **É possível obter o valor exato desta área?**

Representação dos Números Reais

- Os erros ocorridos dependem da representação dos números na máquina utilizada.
- A representação de um número depende
  - da base escolhida ou disponível na máquina em uso
  - e do número máximo de dígitos usados na sua representação.

Representação dos Números Reais

- O número π , por exemplo, não pode ser representado através de um número finito de dígitos decimais.
- No exemplo mostrado acima, o número π foi escrito como 3.14, 3.1416 e 3.141592654 respectivamente nos casos (a), (b) e (c).
- Em cada um deles foi obtido um resultado diferente, e o erro neste caso depende exclusivamente da aproximação escolhida para π .
- Qualquer que seja a circunferência, a sua área nunca será obtida exatamente, uma vez que π é um número irracional.

Representação dos Números Reais

- Como neste exemplo, qualquer cálculo que envolva números que não podem ser representados através de um número finito de dígitos não fornecerá como resultado um valor exato.
- Quanto maior o número de dígitos utilizados, maior será a precisão obtida.
- Por isso, a melhor aproximação para o valor da área da circunferência é aquela obtida no caso (c).

Representação dos Números Reais

- Além disso, um número pode ter representação finita em uma base e não-finita em outras bases.
- A base decimal é a que mais empregamos atualmente.
- Um computador opera normalmente no sistema binário.

Representação dos Números Reais

- Observe o que acontece na interação entre o usuário (ou dados do programa) e o computador:
  - os dados de entrada são enviados ao computador pelo usuário no sistema decimal;
  - toda esta informação é convertida para o sistema binário, e as operações todas serão efetuadas neste sistema.
  - os resultados finais serão convertidos para o sistema decimal e, finalmente, serão transmitidos ao usuário.
- Todo este processo de conversão é uma fonte de erros que afetam o resultado final dos cálculos.

Sistema de Numeração

- Existem vários sistemas numéricos, dentre os quais destacam-se o sistema decimal (base 10), o octal (base 8) e o hexadecimal (base 16) e o binário (base 2).
- Em um sistema numérico com base β , existem β dígitos e o maior é β – 1.
- Por exemplo, em sistema decimal maior dígito é 9.
- De um modo geral, um número na base β , $(a_j\ a_{j-1}\ ...a_2\ a_1\ a_0)_{\beta}$,

  $0 \leq a_k \leq (\beta - 1)$, k = 1, 2, ..., j , pode ser escrito na forma polinomial:

$$a_j \beta^j + a_{j-1} \beta^{j-1} + \ldots + a_2 \beta^2 + a_1 \beta^1 + a_0 \beta^0$$

- Com esta representação, podemos facilmente converter um número representado em qualquer sistema para o sistema decimal.

Sistema de Numeração Decimal

- No sistema de numeração usual, o sistema decimal, usamos dez dígitos 0, 1, ..., 9.
- Por exemplo, número 2153:

$$2153 = 2*10^3 + 1*10^2 + 5*10^1 + 3*10^0 = 2000 + 100 + 50 + 3 = 2153$$

- Um número é expresso como uma soma de potências de 10 multiplicadas por coeficientes apropriados.
- No sistema decimal, 10 é a base do sistema.
- Existem 10 dígitos, o maior sendo 9.
- Em um sistema numérico com base β , existem β dígitos e o maior é β -1.

Sistema de Numeração Binário

- No sistema binário existem apenas 2 dígitos: **0** e **1**
- Como os circuitos eletrônicos usados no computador apresentam 2 estados possíveis, convencionou-se chamar o estado desligado (tensão baixa) de 0 e o estado ligado (tensão alta) de 1.
- Cada dígito de um número representado no sistema binário é denominado bit (**BI**nary digi**T**)
- O 8 bits corresponde a 1 byte.

Sistema de Numeração Binário

- A representação dos números binários num computador é feita com um número finito de bits.
- Esse tamanho finito de bits é chamado de **palavra de computador (word)**
- O tamanho da palavra do computador depende de características internas à arquitetura do mesmo.
- Em geral, os microcomputadores padrão PC tem tamanho de palavra de **16** e **32** bits.
- Computadores modernos tem palavras de **64** bits ou mais.
- Quanto maior o tamanho da palavra do computador mais veloz e mais preciso será o computador.

Sistema de Numeração Binário

- Um computador não consegue representar números infinitamente grandes e/ou infinitamente pequenos.
- Os limites são impostos pela arquitetura e pela definição dos tipos de dados.
- Se esses limites não foram respeitados podemos ter erros do tipo:
  - **Overflow**: número a representar é maior que maior número possível de ser representado
  - **Underflow**: número a representar é menor que menor número possível de ser representado

Exemplo

```c
#include <stdio.h>
#include <limits.h>

 int main()
 {
    printf("\t\tUsing <limits.h> library definitions...\n");

    int n_min;
    int n_max;

    printf("\n INT \n");
    printf(" Storage size (bytes): %lu \n", sizeof(int));
    printf("\n --------------------------------\n");

    printf("\n Signed int min: %d\n", INT_MIN);
    n_min = INT_MIN;
    n_min = n_min - 100;
    printf("\n n_min - 100 = %i", n_min);
    printf("\n Underflow \n");

    printf("\n --------------------------------\n");

    printf("\n Signed int max: %d\n", INT_MAX);
    n_max = INT_MAX;
    n_max = n_max + 100;
    printf("\n n_max + 100 = %i", n_max);
    printf("\n Overflow \n");

    return 0;
}
```

Conversão do Sistema Binário para Decimal

- **Números inteiros**
- Dado um número N, binário, para expressá-lo em decimal, deve-se escrever cada número que o compõe (bit), multiplicado pela base do sistema (base = 2), elevado à posição que ocupa.
- Exemplo: **1001**(binário)

$\mathbf{1} \times 2^{\mathbf{3}} + \mathbf{0} \times 2^{\mathbf{2}} + \mathbf{0} \times 2^{\mathbf{1}} + \mathbf{1} \times 2^{\mathbf{0}} = 9$

Portanto, 1001 é 9 em decimal

Conversão do Sistema Binário para Decimal

- Para facilitar a conversão podemos usar a tabela

	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
N	**2048**	**1024**	**512**	**256**	**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**
									1	**0**	**1**	**1**

$(\mathbf{1011})_2 = 1 * 8 + 0 * 4 + 1 * 2 + 1 * 1 = 11$

Conversão do Sistema Binário para Decimal

- **Exercício 1:** Converter para sistema decimal

1) $(101)_2 =$

2) $(1100)_2 =$

3) $(110100)_2 =$

4) $(1101001)_2 =$

5) $(11010100)_2 =$

	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
N	**2048**	**1024**	**512**	**256**	**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**

Conversão do Sistema Binário para Decimal

- Números reais
- Nesse caso as potencias de 2 que correspondem a parte depois do ponta são 2^{-1}, 2^{-2} e etc.
- Lembrando da regra

$$a^{-n}=\frac{1}{a^n}$$

temos

$$2^{-1}=\frac{1}{2^1}=\frac{1}{2}=0{,}5$$

e assim por diante

2^5	2^4	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}
32	**16**	**8**	**4**	**2**	**1**	**.**	0,5	0,25	0,125	0,0625	0,03125
						.					

Conversão do Sistema Binário para Decimal

- Por exemplo

$(10011,101)_2 =$

2^5	2^4	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}
32	16	8	4	2	1	.	0,5	0,25	0,125	0,0625	0,03125
	1	0	0	1	1	.	1	0	1		

1*16 + 0*8 + 0*4 +1*2 +1*1 +
+ 1 * 0,5 + 0*0,25 +1 *0,125 = 19,625

Conversão do Sistema Binário para Decimal

- **Exercício 2:** Converter para sistema decimal

1) $(110{,}001)_2$ =

2) $(100{,}1101)_2$ =

3) $(1101{,}0101)_2$ =

4) $(11110{,}10111)_2$ =

5) $(10011{,}01001)_2$ =

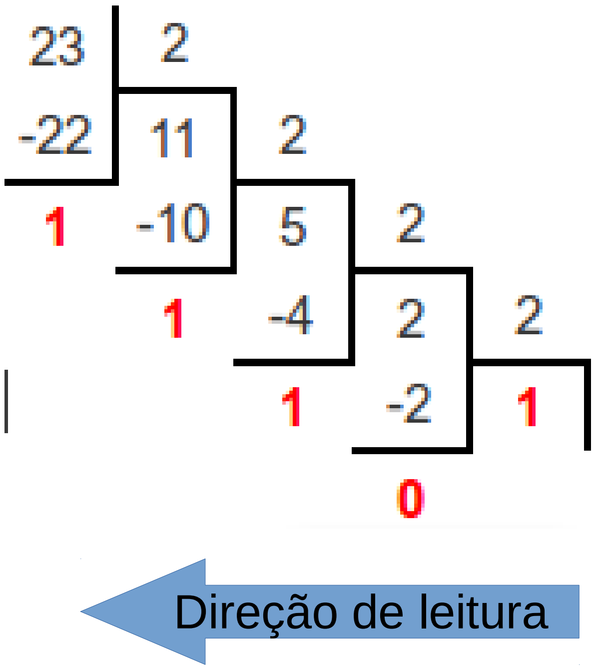
2^5	2^4	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}
32	16	8	4	2	1	.	0,5	0,25	0,125	0,0625	0,03125
						.					

Conversão do Sistema Decimal para Binário

- Uma das formas de conversão de um numero decimal para um numero binário é aplicar um método para a parte inteira (divisões sucessivas).

Exemplo:

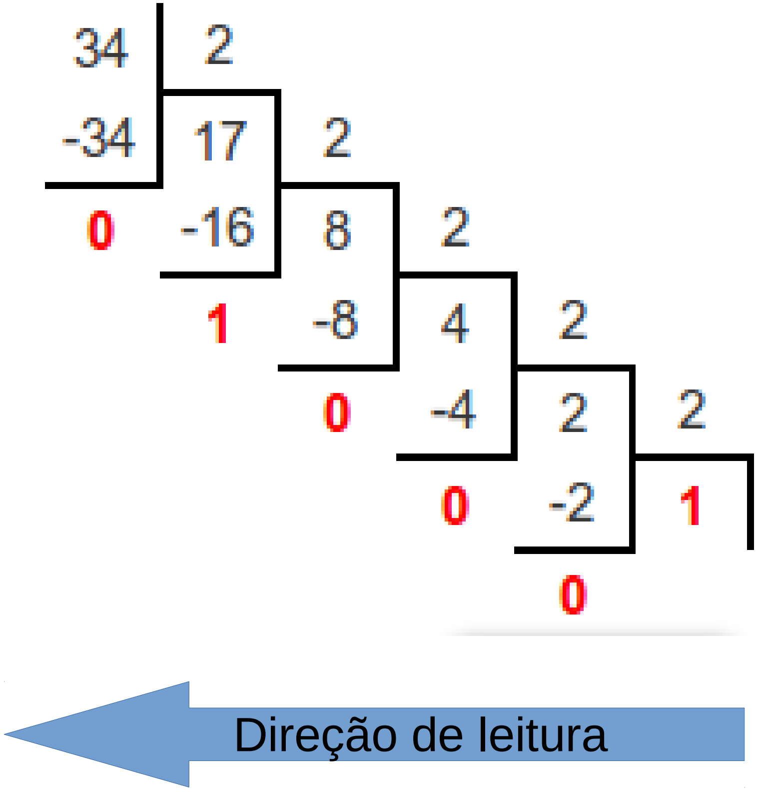
$(23)_{10} = 10111_2$



Conversão do Sistema Decimal para Binário

Exemplo:

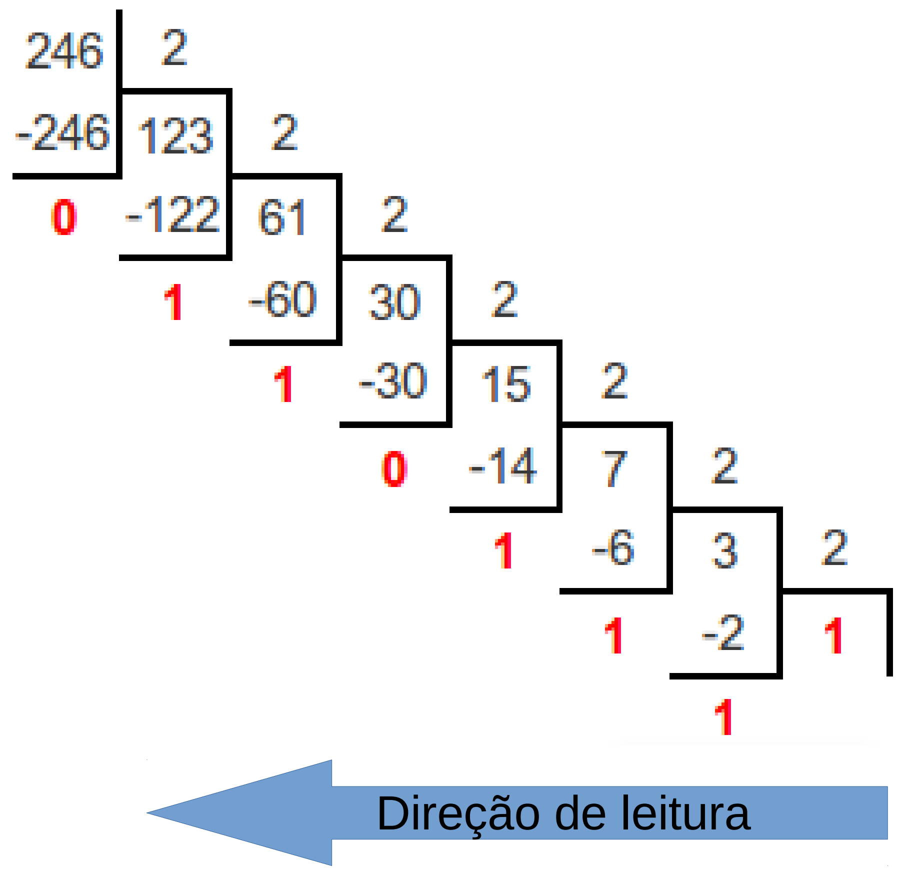
$(34)_{10} = 100010_2$



Conversão do Sistema Decimal para Binário

Exemplo:

$(246)_{10} = 11110110_2$



Conversão do Sistema Decimal para Binário

- Outra forma de conversão é utilizando a tabela

	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
N	**2048**	**1024**	**512**	**256**	**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**
1634		**1**	**1**	**0**								

- O procedimento se inicia do extremo esquerdo, e consiste na verificação de uma possível **subtração não-negativa** (N – potencia de 2)
- 1024<1634. Logo fica "1";

  1634-1024=610
- 512<610. Logo fica "1"

  610-512=98
- 256>98. Logo fica "0"

Conversão do Sistema Decimal para Binário

	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
N	2048	1024	512	256	128	64	32	16	8	4	2	1
1634		**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**1**	**0**

- 256>98. Logo fica "0"
- 128>98. Logo fica "0"
- 64<98. Logo fica "1"

  98-64=34
- 32<34. Logo fica "1"

  34-32=2
- 16>2. Logo fica "0"
- 8>2. Logo fica "0"
- 4>2. Logo fica "0"
- 2=2. Logo fica "1"

  2-2=0
- 1>0. Logo fica "0"

Resultado da conversão: 11001100010

Conversão do Sistema Decimal para Binário

Números Reais

- Para números fracionários utilizamos a regra da multiplicação
- Temos que aplicar sucessivas multiplicações por 2, guardando-se os zeros.
- Exemplo:

$\mathbf{(0{,}125)_{10} = {,}\, a_{-1}\, a_{-2}\, a_{-3}\, a_{-4} \ldots}$

$2 \times (0{,}125)_{10} = 0{,}250$; Não há parte real, resultado **0**;

$2 \times (0{,}250)_{10} = 0{,}50$; Não há parte real, resultado **0**;

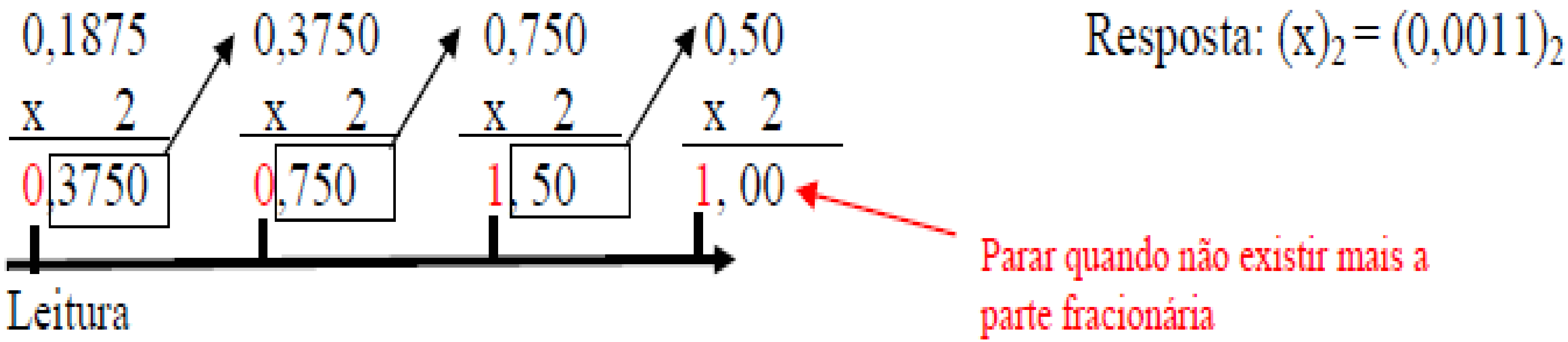
$2 \times (0{,}50)_{10} = 1{,}0$; Há parte real, resultado **1**; Resto 0.

Assim, $(\mathbf{0{,}125})_{10} = (\mathbf{0{,}001})_2$

Conversão do Sistema Decimal para Binário

- Exemplo:

$(0{,}1875)_{10} = (0{,}0011)_2$



Sistema Decimal para Binário

- **Exercício 3:** Converter para sistema binário

1) $(252)_{10} =$

2) $(315)_{10} =$

3) $(1031)_{10} =$

4) $(23,125)_{10} =$

5) $(45,9375)_{10} =$

6) $(121,75)_{10} =$

Representação Finita

- Nem todos os números que são representados com um número finito de algarismos numa base o são nas outras.
- Por exemplo, 0,1 tem representação finita na base 10, mas, na base 2, será que também o é?
- Vejamos

2 × (0,1) 10 = 0, 2 ; Não há parte real, resultado 0;

2 × (0,2) 10 = 0, 4 ; Não há parte real, resultado 0;

2 × (0,4) 10 = 0,8 ; Não há parte real, resultado 0;

2 × (0,8) 10 = 1, 6 ; Há parte real, resultado 1, sobra 0,6;

2 × (0,6) 10 = 1, 2 ; Há parte real, resultado 1, sobra 0,2;

2 × (0,2) 10 = 0, 4 ; Não há parte real, resultado 0;

….

Assim, $(0,1)_{10} = (0,0001100110011...)_2$

Neste caso e em outros casos semelhantes, ficamos sujeitos ao limite físico de representação da máquina que usamos para fazer os cálculos.

Representação Finita

- Nesse caso concluímos que o numero $(0,1)_{10}$ NÃO tem representação binaria finita !!!
- Por mais moderno que seja o computador ele nunca vai saber exatamente o que significa o numero $(0,1)_{10}$ pois sua conversão para binário sempre acarretara numa aproximação (truncamento ou arredondamento)
- Obs. O fato de um numero não ter representação finita no sistema binário pode acarretar a ocorrência de erros aparentemente inexplicáveis nos cálculos dos dispositivos eletrônicos.

Representação dos números no formato Ponto Flutuante

- A principal vantagem da representação em ponto flutuante é que ela pode representar uma grande faixa de números se comparada a representação de ponto fixo
- Por exemplo, se consideramos uma representação com 6 dígitos:
  - Ponto fixo
    - o maior número = 9,99999 ≈ 10
    - o menor número = 0,00001 = 10^{-5}
  - Ponto flutuante: aloca-se dois dos seis dígitos para representar a potência de 10
    - o maior número = 9,999 * 10^{99}
    - o menor número = 0,001 * 10^{-99}

Representação dos números no formato Ponto Flutuante

- A representação em ponto flutuante permite representar uma faixa muito maior de números
- O preço a ser pago é que esta representação tem quatro dígitos de precisão, enquanto a representação por ponto fixo possui seis dígitos de precisão

Representação dos números no formato Ponto Flutuante

- Um computador representa números reais da seguinte forma:

$$N = (\,,a_1\ a_2\ a_3\ a_4\ \ldots\ a_t\,) \times \beta^e$$

- onde $(\,,a_1\ a_2\ a_3\ a_4\ \ldots\ a_t\,)$ é a representação com **t** algarismos do número na base **β** , chamada **mantissa**
- sendo que a **1** deve ser o primeiro algarismo significativo (não nulo)
- **e** é um expoente, cujo valor vai de um limite inferior **-m** até um limite superior **M**, que depende da capacidade da máquina
- Esta forma de representação foi descoberta por Konrad Zuse (1910-1995) para os seus computadores eletromecânicos Z1 e Z3, no início da década de 40

Representação dos números no formato Ponto Flutuante

- Exemplo:

  $x = (34,2)_{10}$

  $x = 0,3420*10^{2}$

  - t (quantidade de algarismos): 4
  - β (base): 10
  - mantissa: 3420

Representação dos números no formato Ponto Flutuante

- Exemplo:

  $x = (0{,}00011001100110011...)_2$

  $x = 0{,}110011001 * 2^{-3}$

  - t (quantidade de algarismos): 9
  - β (base): 2
  - mantissa: 110011001

Representação dos números no formato Ponto Flutuante

Exemplos:

a) $0{,}35 = (3x10^{-1} + 5x10^{-2})x10^{0} = 0{,}35x10^{0}$

b) $-5{,}172 = -(5x10^{-1} + 1x10^{-2} + 7x10^{-3} + 2x10^{-4})\ x\ 10^{1} = -0{,}5172x10^{1}$

c) $0{,}0123 = (1x10^{-1} + 2x10^{-2} + 3x10^{-3})\ x\ 10^{-1} = 0{,}123x10^{-1}$

d) $5391{,}3 = (5x10^{-1} + 3x10^{-2} + 9x10^{-3} + 1x10^{-4} + 3x10^{-5})\ x10^{4} = 0{,}53913x10^{4}$

e) $0{,}0003 = (3x10^{-1})x10^{-3} = 0{,}3x10^{-3}$

Considerando agora que estamos diante de uma máquina que utilize apenas três dígitos significativos e que tenha como limite inferior e superior para o expoente, respectivamente, -2 e 2, como seriam representados nesta máquina os números do exemplo anterior?

a) $0{,}35 = 0{,}350x10^{0}$

b) $-5{,}172 = -0{,}517x10^{1}$

c) $0{,}0123 = 0{,}123x10^{-1}$

d) $5391{,}3 = 0{,}539x10^{4}$ Não pode ser escrito nessa máquina. Erro de *overflow*

e) $0{,}0003 = 0{,}3x10^{-3}$ Não pode ser escrito nessa máquina. Erro de *underflow*

Representação dos números no formato Ponto Flutuante

- O padrão IEEE 754 para ponto (vírgula) flutuante é a representação mais comum para números reais em computadores de hoje, incluindo PC's compatíveis com Intel, Macintosh, e a maioria das plataformas Unix/Linux.
- O padrão (ou norma) IEEE 754 define dois formatos básicos para os números em ponto flutuante:
  - o formato ou precisão simples, com 32 bits; e,
  - o duplo com 64 bits.

	Sinal	Expoente(+/-)	Mantissa
Simples (32bits)	1 [bit31]	8 [bits30-23]	23 [bits22-00]
Dupla (64 bits)	1 [bit63]	11 [bits62-52]	52 [bits51-00]

- **Precisão Dupla**: a variável será representada no sistema de aritmética de ponto flutuante da máquina, mas com aproximadamente o dobro de dígitos disponíveis na mantissa.

  O tempo de execução e os requerimentos de memória aumentam significativamente.

- Sinal: 0 = + e 1 = -
- Combinações: Sinal + Expoente + Mantissa

Representação dos números no formato Ponto Flutuante

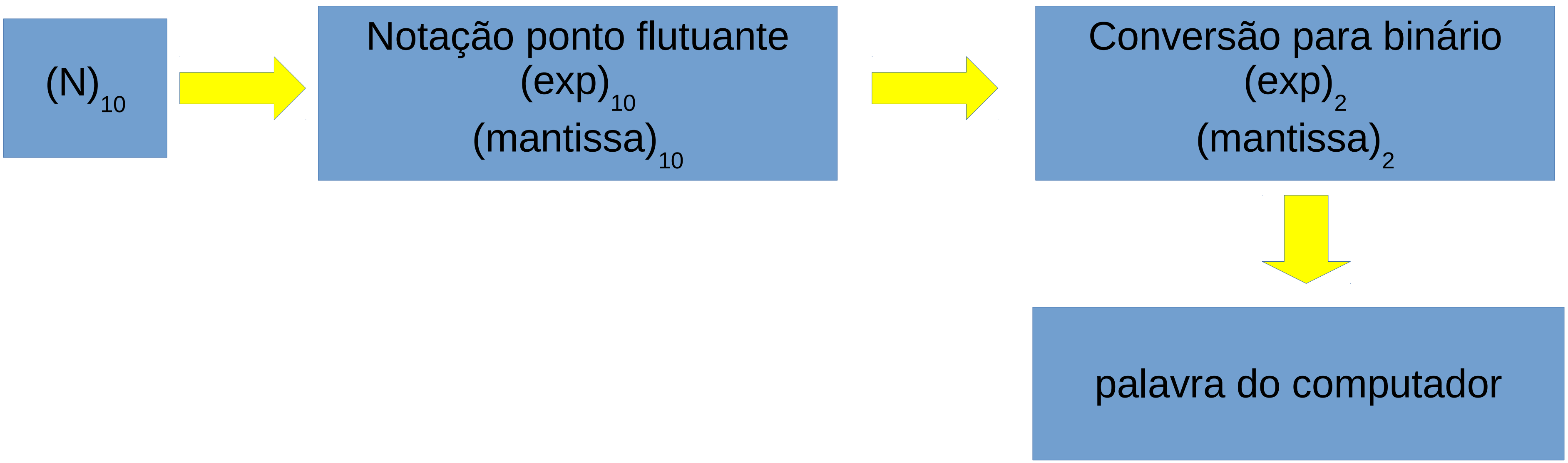
- O conjunto dos números de ponto flutuante é discreto, e não continuo como os números reais.
- Não temos mais o conceito que entre dois números sempre existe um outro.

Aritmética de Ponto Flutuante

- Assim:
  - $24543=0{,}24543 \times 10^5$, se a máquina pode apresentar 5 algarismos na mantissa
  - $24543=0{,}245 \times 10^5$, se a máquina pode apresentar 3 algarismos na mantissa
  - $24543=0{,}25 \times 10^5$, se a máquina pode apresentar 2 algarismos na mantissa.
- Repare que usamos de arredondamento, não truncamento para escolhermos o último algarismo apresentado na mantissa.
- Um número grande demais para ser apresentado por uma máquina provoca o que chamamos de OVERFLOW (um número por demais pequeno causa um UNDERFLOW).
- Nesses casos, dependendo da máquina e da linguagem de programação, os efeitos podem ser os mais variáveis, desde o truncamento involuntário à parada forçada do cálculo, ou a representação simbólica **NAN** (Not a Number)

Aritmética de Ponto Flutuante

Uma máquina digital (que opera em base 2) armazena um número internamente da seguinte forma:



exp						mantissa									
±	0	1	0	1	0	±	0	1	0	1	0	1	0	1	0

Erros na representação dos números

- O conjunto de números reais é infinito, entretanto, a sua representação em um sistema de ponto flutuante é limitada, pois é um sistema finito. Essa limitação tem duas origens:
  - a faixa dos exponentes é limitada $e_{min} \leq e \leq e_{max}$
  - a mantissa representa um número finito de números

Erros na representação dos números

Exemplo:

- Considerando β = 10, t = 2, e_{min}=-5, e_{max}=5 calcular **x * y**, sendo x = 875 e y = 3172
- Primeiro, temos que arredondar os números e armazená-los no formato indicado
- A operação de multiplicação é efetuada usando t=2 dígitos
- $x = 0{,}875 * 10^3 = 0{,}88 * 10^3$
- $y = 0{,}3172 * 10^4 = 0{,}32 * 10^4$
- $x * y = 0{,}2816 * 10^7 = 0{,}28 * 10^7$
- Como exponente é maior do que 5, resulta em **overflow**

Erros na representação dos números

- **Exemplo:**
- Considerando $\beta = 10$, $t = 2$, $e_{min} = -5$, $e_{max} = 5$ calcular **x / y**, sendo $x = 0{,}0064$ e $y = 7312$
- Primeiro, temos que arredondar os números e armazená-los no formato indicado
- A operação de divisão é efetuada usando t=2 dígitos
- $x = 0{,}64 * 10^{-2}$
- $y = 0{,}7312 * 10^{4} = 0{,}73 * 10^{4}$
- $x / y = 0{,}87671 * 10^{-6} = 0{,}88 * 10^{-6}$
- Como exponente é menor do que -5, resulta em **underflow**

Erros na representação dos números

Exemplo:

- Considerando uma máquina que opere com apenas 6 dígitos na mantissa, ou seja, que seja capaz de armazenar números no formato
  $m = \pm\ 0,\ d_1\ d_2\ d_3\ d_4\ d_5\ d_6 * 10^e$
- Como podemos armazenar o número

$(0{,}10)_{10} = (0{,}000111000010100011110101110000101000111101\ \ldots..)_2$

Como esse número não tem representação finita, teremos nesse caso

$(\mathbf{0{,}11})_{10} \rightarrow (0{,}000111)_2 \rightarrow (\mathbf{0{,}109375})_{10}$

Erros na representação dos números

Exemplo:

- Vamos considerar a representação binária de **0,6** e **0,7**
- 0,6 = 0,100110011001.....
- 0,7 = 0,1011001100110......
- Se esses dois números forem representados em sistema binário com 2 dígitos para mantissa e com e_{min}= -1, e_{max}= 2 eles serão representados igualmente por

  $0{,}10 * 2^0$
- Esse número equivale a **0,5** em decimal
- Portanto, tanto o **0,6** quando o **0,7** serão considerados **0,5**

Erros e condicionamento

- Um problema numérico é dito mal condicionado ou instável quando sua solução é muito suscetível aos dados de entrada.
- Por exemplo temos uma simples equação do segundo grau

$\mathbf{x^2 - 100.22\,x + 1.2371 = 0}$

que tem como soluções os valores para **x**

$$x_{1,2} = \frac{b \mp \sqrt{b^2 - 4ac}}{2a}$$

Erros e condicionamento

- Levando em conta que o número de algarismos que podemos usar para efetuar os cálculos é sempre finito (no caso vamos pegar cinco) temos:

$$b^2 = 10044$$

$$b^2 - 4ac = 10039$$

$$\sqrt{b^2 - 4ac} = 100,19$$

- chegamos às soluções:

$x_1 = 100,20$ *e* $x_2 = 0,015$

Erros e condicionamento

- Acontece que se substituirmos essas soluções na equação original, não encontraremos a confirmação de sua validade.
- Se ao invés disso usarmos uma propriedade das soluções,

$$x_2 = \frac{c}{a\,x_1} = 0{,}012346$$

vamos chegar muito mais próximo da solução real

- Neste caso o problema não é matematicamente mal posto.
- O fato de ***b*** ser muito maior que ***4ac*** foi o que induziu a um erro maior já que tivemos no segundo passo acima a subtração de números muito próximos, próximos do limite da representação a nós imposta.

Erros na representação dos números

- **Exercício 4:**
- Realizar os cálculos e analisar os resultados obtidos em sistema definido por

$\beta = 10$, $t = 2$, $e_{min} = -5$, $e_{max} = 5$

a) Calcular **x + y ,** sendo x = 4,32 e y = 0,064

b) Calcular **x - y ,** sendo x = 372 e y = 371

c) Calcular **x + y ,** sendo x = 691 e y = 2,71