Representação Binária no Computador

INF1608 – Análise Numérica

Departamento de Informática, PUC-Rio

Problema

Considere o trecho de código C abaixo

```c
#include <stdio.h>

int main (void)
{
  double a = 1.2 - 1.0 - 0.2;
  if (a == 0.0)
    printf("OK\n");
  else
    printf("ERRO\n");
  return 0;
}
```

- Qual é a mensagem exibida?

Problema

Considere o trecho de código C abaixo

```c
#include <stdio.h>

int main (void)
{
  double a = 1.2 - 1.0 - 0.2;
  if (a == 0.0)
    printf("OK\n");
  else
    printf("ERRO\n");
  return 0;
}
```

- Qual é a mensagem exibida?

  ```
  ERRO
  ```

  - De fato:

    ```c
    printf("%.16g\n",a);
    ```

    - Saída: -5.551115123125783e-17

Representação binária

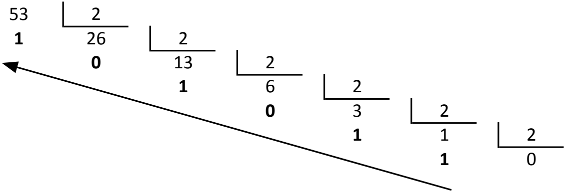
Conversão de decimal para binário

- $(53)_{10}$ em representação binária?

Representação binária

Conversão de decimal para binário

- $(53)_{10}$ em representação binária?



Então:

$$(53)_{10} = (110101)_2$$

Representação binária

$(0.7)_{10}$?

Representação binária

$(0.7)_{10}$?

$0.7 \times 2 = 0.4 + 1$

$0.4 \times 2 = 0.8 + 0 \leftarrow$

$0.8 \times 2 = 0.6 + 1$

$0.6 \times 2 = 0.2 + 1$

$0.2 \times 2 = 0.4 + 0$

$0.4 \times 2 = 0.8 + 0 \leftarrow$ *repete*

Então:

$$(0.7)_{10} = (0.1011001100110...)_2 = (0.1\overline{0110})_2$$

Logo:

$$(53.7)_{10} = (110101.1011001100110...)_2 = (110101.1\overline{0110})_2$$

Representação binária

Exemplo exato: $(0.25)_{10}$?

Representação binária

Exemplo exato: $(0.25)_{10}$?

$$\begin{aligned} 0.25 \times 2 &= 0.5 + 0 \\ 0.5 \times 2 &= 0.0 + 1 \end{aligned}$$

Logo:

$$(0.25)_{10} = (0.01)_2$$

Representação binária

Conversão binário para decimal

- Parte inteira:

$$(10101)_2 \ ?$$

$$1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (21)_{10}$$

- Parte fracionária:

$$(0.1011)_2 \ ?$$

$$\frac{1}{2^1} + \frac{0}{2^2} + \frac{1}{2^3} + \frac{1}{2^4} =$$
$$= \frac{1}{2} + \frac{0}{4} + \frac{1}{8} + \frac{1}{16} = \left(\frac{11}{16}\right)_{10}$$

Representação binária

Caso de "dízimas" binárias: [1]

$$x = (0.\overline{1011})_2$$

$$\begin{aligned} 2^4 x &= 1011.\overline{1011} \\ x &= 0000.\overline{1011} \end{aligned}$$

$$(2^4 - 1)x = (1011)_2 = (11)_{10}$$

Logo:

$$x = \frac{11}{2^4 - 1} = \left(\frac{11}{15}\right)_{10}$$

Representação binária

Caso de "dízimas" binárias: [2]

$$x = (0.10\overline{101})_2$$

$$y = 2^2 x = (10.\overline{101})_2 = (10)_2 + (0.\overline{101})_2 = (10)_2 + z$$
$$z = (0.\overline{101})_2$$

Procedimento:

- Achar $z : \frac{5}{7}$
- Achar $y : (10)_2 + (0.\overline{101})_2 = 2 + z = 2 + \frac{5}{7} = \frac{19}{7}$
- Achar $x : \frac{y}{2^2} = \frac{19}{28}$

Representação no computador

Notação científica para números binários:

- Exemplo: $(1010.01)_2 = (1.01001 \times 2^3)_2$

Representação:

$$\pm 1.bbbb... \times 2^{eee...}$$

- Forma "Normalizada"
- A parte inteira é sempre 1 (implícito e não armazenado)
- A base é 2 (também implícito)
- *bbbb*... representa a mantissa
- *eee*... representa o expoente

Representação no computador

Tipos ponto flutuante no computador

- Diferem no número de bits

	sinal	mantissa	expoente
float	1	23	8
double	1	52	11

- *float* $\approx \pm 3.4028235 \times 10^{38}$
- *double* $\approx \pm 1.7976931 \times 10^{308}$

Representação no computador

Precisão simples (float):

- Quantos dígitos decimais de precisão?
- Pelos bits da mantissa em binário

$$2^{23} = 10^x$$

$$\log_2 2^{23} = \log_2 10^x$$
$$23 = x \log_2 10$$

- Como $\log_2 10 \approx 3.322$, temos:

$$x = 6.92$$

Precisão dupla (double):

$$x = \frac{52}{\log_2 10}$$
$$x = 15.65$$

De fato, note:

```
printf("%.16g\n",0.3)
--> 0.3
printf("%.17g\n",0.3)
--> 0.29999999999999999
```

Representação no computador

Precisão dupla (double)

- Representação na máquina

$$\underbrace{se_1e_2 \ldots e_{11}b_1b_2 \ldots b_{52}}_{64 \text{ bits}}$$

- Representação do número 1:

$$+1.000\ldots000 \times 2^0$$

  - Menor número representável maior que 1 (*nextafter*):

$$+1.000\ldots001 \times 2^0 = 1 + 2^{-52}$$

Epsilon da máquina

$$\epsilon_{mach} = 2^{-52} \approx 2.22 \times 10^{-16}$$

Arredondamento no sistema binário

De acordo com a *IEEE*:

- Se bit 53 for 0: descarta-se bits 53 em diante
- Se bit 53 for 1 e existir bit > 53 com valor 1: soma-se 2^{-52}, descarta-se bits 53 em diante
- Se bit 53 for 1 e bits > 53 for 0
  - Se bit 52 for 0: descarta-se bits 53 em diante
  - Se bit 52 for 1: soma-se 2^{-52}, descarta-se demais
  - Ao final, bit 52 fica com valor 0

Representação no computador

Representação do expoente

- 11 bits (valores positivos): 0 a 2047
- Valores especiais: 0 e 2047
- Valor do expoente: $e_{xp} \in [-1022, 1023]$
  - Avaliação dos bits com valores: $x_b \in [1, 2046]$

$$e_{xp} = x_b - 1023$$

- Exemplos
  - Se $x_b = 1$: $e_{xp} = -1022$
  - Se $x_b = 2046$: $e_{xp} = 1023$
  - Se $x_b = 1023$: $e_{xp} = 0$

Em resumo:

- Para armazenar (achar x_b): soma-se 1023 de e_{xp}
- Para interpretar (achar e_{xp}): subtrai-se 1023 de x_b

Representação no computador

Números especiais

- Expoente $x = (11111111111)_2 = (2047)_{10}$
  - Se mantissa diferente de zero: *NaN*
  - Se mantissa for zero: $\pm Inf$

$$+Inf : 011111111111000...000$$

$$-Inf : 1\,\underbrace{11111111111}_{11 bits}\,\underbrace{000...000}_{52 bits}$$

- Expoente: $(00000000000)_2 = (0)_{10}$
  - Números sub-normais (não normalizados)

$$\pm 0.b_1 b_2 ... b_{52} \times 2^{-1022}$$

Logo:

- Menor número diferente de 0 representável

$$2^{-52} \times 2^{-1022} = 2^{-1074}$$

$$\underbrace{0}\ \underbrace{00000000000}\ \underbrace{000000...0000001}$$

Representação no computador

Observações:

- Números menores que 2^{-1074} não podem ser representados
- Existem números representáveis ($< \epsilon_{mach}$) que, se somados a 1, não alteram seu valor
  - $1 \times 2^0 + 2^{-52} \times 2^{-1022} = 1$
  - $1 \times 2^0 + 1 \times 2^{-53} = 1$
- Números sub-normais incluem o zero: ± 0
  - -0 e $+0$ são tratados como iguais

Representação no computador

Exercício:

- Se $fl(x)$ indica a representação ponto flutuante do valor x no computador, podemos afirmar que $fl(0.2) < 0.2$?

Representação binária de 0.2:

$$
\begin{aligned}
0.2 \times 2 &= 0.4 + 0 \\
0.4 \times 2 &= 0.8 + 0 \\
0.8 \times 2 &= 0.6 + 1 \\
0.6 \times 2 &= 0.2 + 1 \\
0.2 \times 2 &= 0.4 + 0 \\
\ldots
\end{aligned}
$$

Logo: $(0.2)_{10} = (0.\overline{0011})_2$

Representação no computador

Exercício (cont):

$$(0.2)_{10} = (0.\overline{0011})_2 = 0.001100110011...$$

Notação científica:

$$1.1001\overline{1001} \times 2^{-3}$$

- Então, o bit 53 vale 1, com valores diferentes de 0 em seguida
  - Soma-se $2^{-52}2^{-3}$, descarta-se $(0.\overline{1001})_2 \times 2^{-52}2^{-3}$
  - $(0.\overline{1001})_2 = \frac{9}{15}$
  - Isto é:

$$\begin{aligned} fl(0.2) &= 0.2 + 2^{-55} - \frac{9}{15} \times 2^{-55} \\ &= 0.2 + (1 - 0.6) \times 2^{-55} = 0.2 + 0.4 \times 2^{-55} > 0.2 \end{aligned}$$

Representação no computador

Entendendo o exercício do início da aula

```
#include <stdio.h>

int main (void)
{
  double a = 1.2 - 1.0 - 0.2;
  if (a == 0.0)
    printf("OK\n");
  else
    printf("ERRO\n");
  return 0;
}
```

Representação no computador

Avaliação da expressão: $1.2 - 1.0 - 0.2$

$$fl(1.2) = ?$$

$$fl(1.0) = 1.0$$

$$fl(0.2) = 0.2 + 0.4 \times 2^{-55}$$

Temos:

$$(1.2)_{10} = (1.\overline{0011})_2$$

$$fl(1.2) = 1.2 - 0.2 \times 2^{-52}$$

Então:

$$\begin{aligned} fl(1.2 - 1.0 - 0.2) &= 1.2 - 0.2 \times 2^{-52} - 1.0 - (0.2 + 0.4 \times 2^{-55}) \\ &= (-1.6 - 0.4) \times 2^{-55} = -2 \times 2^{-55} = -2^{-54} \\ &= -5.5511151231257827 \times 10^{-17} \end{aligned}$$

Representação no computador

Corrigindo o código

- Use tolerância numérica em comparação de ponto flutuante

```c
#include <stdio.h>
#include <math.h>
#define TOL 1e-15 // 1 * 10^-15

int main (void)
{
  double a = 1.2 - 1.0 - 0.2;
  if (fabs(a) < TOL)
    printf("OK\n");
  else
    printf("ERRO\n");
  return 0;
}
```

Avaliação de erro

Como avaliar o erro de um método/procedimento?

Erro absoluto

$$|x_c - x| < \epsilon$$

onde x_c é o valor computado e x é o valor exato

- Pode falhar se os números forem pequenos

Avaliação de erro

Erro relativo

$$\frac{|x_c - x|}{|x|} < \epsilon$$

- O padrão IEEE garante: $\frac{|fl(x)-x|}{|x|} \leq \frac{1}{2}\epsilon_{mach}$
- Pode falhar quando:
  - x e x_c são zero: $=>$ *NaN*
  - x é zero: $=>$ *Inf*
  - Quando solução está próxima de zero

Fórmula híbrida (absoluto/relativo)

$$\frac{|x_c - x|}{\max(|x|, \delta)} < \epsilon, \quad \text{com} \quad \delta > 0$$

Avaliação de erro

Definição

- Uma solução é correta com precisão de p casas decimais se:

$$erro < 0.5 \times 10^{-p}$$

Exercício proposto

Se $fl(x)$ indica a representação do número x em precisão *double* nos computadores modernos, qual o erro relativo de $fl(0.3)$? O padrão IEEE garante $e \leq \frac{1}{2}\epsilon_{mach}$; verifique.

Verifique: $fl(1.2) > 1.2$?

Qual o valor da expressão $fl(2.3 - 2 - 0.3)$?

Exercício proposto [1]

Se $fl(x)$ indica a representação do número x em precisão *double* nos computadores modernos, qual o erro relativo de $fl(0.3)$? O padrão IEEE garante $e \leq \frac{1}{2}\epsilon_{mach}$; verifique.

Calcular $fl(0.3) = 0.0\overline{1001}$

- Normalizar: $0.0\overline{1001} = 1.\overline{0011} \times 2^{-2}$
- Arredondar: ...

Bits 53 a 56 continuam a dízima periódica $\overline{0011} = 0.2$.

Bit 53 é 0.

Para descartar os bits a partir de 53, subtraimos $0.\overline{0011} \times 2^{-54}$.

Logo $fl(0.3)$ é: $fl(0.3) = 0.3 - 0.2 \times 2^{-54}$

Exercício proposto [2]

Se $fl(x)$ indica a representação do número x em precisão *double* nos computadores modernos, qual o erro relativo de $fl(0.3)$? O padrão IEEE garante $e \leq \frac{1}{2}\epsilon_{mach}$; verifique.

Sabemos que $fl(0.3) = 0.3 - 0.2 \times 2^{-54}$

Erro relativo: $\frac{|x_c - x|}{|x|} \leq \frac{1}{2}\epsilon_{mach}$

$$\frac{|0.3 - 0.2 \times 2^{-54} - 0.3|}{|0.3|} = \frac{|-0.2 \times 2^{-54}|}{|0.3|} = \frac{|-2 \times 2^{-54}|}{|3|}$$

$$\frac{|-2 \times 2^{-54}|}{|3|} \times \frac{2}{2} \times \frac{2}{2} = \frac{|-2 \times 2^{-52}|}{|12|} = \frac{|-1|}{|6|} \times 2^{-52} \leq \frac{1}{2}\epsilon_{mach}$$

Exercício proposto [2]

Verifique: $fl(1.2) > 1.2$?

$fl(1.2) = 1.\overline{0011}$

- Normalizar: Valor já está normalizado
- Arredondar: ...

Bits 53 a 56 são a continuação da dízima periódica $\overline{0011}$.

Utilizando a conversão de dízimas temos que:

$$0.\overline{0011} = 0.2$$

Para descartar os bits a partir de 53, subtraimos $0.\overline{0011} \times 2^{-52}$.

Logo $fl(1.2)$ é:

$$fl(1.2) = 1.2 - 0.2 \times 2^{-52}$$

$$fl(1.2) < 1.2$$

Exercício proposto [3]

Avaliação da expressão: $2.3 - 2.0 - 0.3$

$$fl(2.3) = ?$$
$$fl(2.0) = 2.0$$
$$fl(0.3) = 0.3 - 0.2 \times 2^{-54}$$

Temos:

$$(2.3)_{10} = (10.0\overline{1001})_2$$

Normalizando:

$$(2.3)_{10} = (1.00\overline{1001})_2 \times 2^1$$

Sabemos que o bit 52 é zero, e o bit 54 é 1 (por causa da dízima). Podemos ou mudar a dízima para alinhar com o bit 53, ou realizar 2 remoções.

Exercício proposto [3]

Avaliação da expressão: $2.3 - 2.0 - 0.3$

$$fl(2.3) = ?$$
$$fl(2.0) = 2.0$$
$$fl(0.3) = 0.3 - 0.2 \times 2^{-54}$$

Realizando 2 remoçoes:

- Remover bit $2^{-54} \times 2^{1} = 2^{-53}$
- Remover dízima $0.\overline{1001} \times 2^{-54} \times 2^{1} = 0.6 \times 2^{-53}$

Logo:

$$\begin{aligned} fl(2.3) &= 2.3 - 2^{-53} - 0.6 \times 2^{-53} \\ &= 2.3 - 1.6 \times 2^{-53} \\ &= 2.3 - 3.2 \times 2^{-54} \end{aligned}$$

Exercício proposto [3]

Avaliação da expressão: $2.3 - 2.0 - 0.3$

$$\begin{aligned} fl(2.3) &= 2.3 - 3.2 \times 2^{-54} \\ fl(2.0) &= 2.0 \\ fl(0.3) &= 0.3 - 0.2 \times 2^{-54} \end{aligned}$$

Então:

$$\begin{aligned} fl(2.3 - 2.0 - 0.3) &= 2.3 - 3.2 \times 2^{-54} - 2.0 - (0.3 - 0.2 \times 2^{-54}) \\ &= -3.2 \times 2^{-54} + 0.2 \times 2^{-54} \\ &= -3 \times 2^{-54} \\ &= -1.6653345 \times 10^{-16} \end{aligned}$$