CC BY

https://xkcd.com/676

Ponto Flutuante - Experimentos

Paulo Ricardo Lisboa de Almeida

DSBD

UFPR

Para começar

Foi disponibilizado um programa C que calcula o valor absoluto de valores em ponto flutuante.

Seu objetivo é utilizar o que aprendeu sobre representação de ponto flutuante para gerar uma versão mais eficiente do cálculo.

Modifique esse trecho.

```c
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define M 1073741824  // 4GB

int main() {
    float* fval = malloc(M * sizeof(float));
    if (fval == NULL) {
        fprintf(stderr, "Falha de alocação");
        exit(1);
    }

    // valores para testar
    for (size_t i = 0; i < M; ++i)
        fval[i] = (int)(i + 1) * -1;

    printf("Antes: %f\n%f\n", fval[0], fval[M - 1]);

    clock_t ticks;
    ticks = clock();
    for (size_t i = 0; i < M; ++i)
        if (fval[i] < 0)
            fval[i] *= -1;

    ticks = clock() - ticks;
    double segundos = ((double)ticks) / CLOCKS_PER_SEC;

    // imprimindo o primeiro e último valores para testar
    printf("Depois: %f\n%f\n", fval[0], fval[M - 1]);
    printf("Segundos: %f\n", segundos);

    free(fval);
    return 0;
}
```

Possível Solução

```c
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define M 1073741824  // 4GB

int main() {
    float* fval = malloc(M * sizeof(float));
    if (fval == NULL) {
        fprintf(stderr, "Falha de alocação");
        exit(1);
    }
    // valores para testar
    for (size_t i = 0; i < M; ++i)
        fval[i] = (int)(i + 1) * -1;

    printf("Antes: %f\n%f\n", fval[0], fval[M - 1]);

    clock_t ticks;
    ticks = clock();
    uint32_t* bits = (uint32_t*)fval;
    for (size_t i = 0; i < M; ++i, ++bits)
        *bits &= 0x7FFFFFFF;  // 0111 1111 1111 1111 1111 1111 1111 1111

    ticks = clock() - ticks;
    double segundos = ((double)ticks) / CLOCKS_PER_SEC;

    // imprimindo o primeiro e último valores para testar
    printf("Depois: %f\n%f\n", fval[0], fval[M - 1]);

    printf("Segundos: %f\n", segundos);

    free(fval);
    return 0;
}
```

Troca do bit de magnitude.

Diferença de tempo

Convencional:
Segundos: 1.432395

Operando nos bits do float:
Segundos: 0.902892

Para comparação

Em Python:

```python
import time
import array

M = 1073741824  # 4GB em floats

def main():
    fval = array.array('f', (float((i + 1) * -1) for i in range(M)))

    # Valores antes
    print(f"Antes: {fval[0]}, {fval[-1]}")

    start = time.time()
    for i in range(M):
        if fval[i] < 0:
            fval[i] *= -1
    end = time.time()

    print(f"Depois: {fval[0]}, {fval[-1]}")
    print(f"Segundos: {end - start:.6f}")

if __name__ == "__main__":
    main()
```

Para comparação

Em Python:

Segundos: 89.276262
98x o tempo do C!!!

```python
import time
import array

M = 1073741824  # 4GB em floats

def main():
    fval = array.array('f', (float((i + 1) * -1) for i in range(M)))

    # Valores antes
    print(f"Antes: {fval[0]}, {fval[-1]}")

    start = time.time()
    for i in range(M):
        if fval[i] < 0:
            fval[i] *= -1
    end = time.time()

    print(f"Depois: {fval[0]}, {fval[-1]}")
    print(f"Segundos: {end - start:.6f}")

if __name__ == "__main__":
    main()
```

Coprocessadores

Ao adicionar a capacidade de processar pontos flutuantes, é necessário decidir quais registradores usar.

Podemos usar os mesmos registradores de “aritmética convencional”.

Coprocessadores

Ao adicionar a capacidade de processar pontos flutuantes, é necessário decidir quais registradores usar.

Podemos usar os mesmos registradores de “aritmética convencional”.

Muitos projetistas optam por usar um conjunto separado de registradores, específico para ponto flutuante.

Coprocessadores

Ao adicionar a capacidade de processar pontos flutuantes, é necessário decidir quais registradores usar.

Podemos usar os mesmos registradores de “aritmética convencional”.

Muitos projetistas optam por usar um conjunto separado de registradores, específico para ponto flutuante.

Exemplos:

- RISC-V
- x86-64
- ARM
- MIPS

Coprocessadores

Conjunto separado de registradores, específico para ponto flutuante.

Vantagens? Desvantagens?

Coprocessadores

Conjunto separado de registradores, específico para ponto flutuante.

+ Mais registradores para trabalhar.
+ Pode liberar a ALU para trabalhar com os registradores “normais” enquanto uma ALU dedicada trabalha com os registradores de ponto flutuante.

- São necessárias instruções específicas para lidar com os novos registradores.
- Muitas vezes é necessário mover os dados entre os bancos de registradores convencionais e de PF.

Coprocessadores

É necessária uma **ALU** especializada para tratar os dados em ponto flutuante.

Essa ALU é comumente chamada de Floating-point Unit - **FPU**.

Um pouco de história

Processadores anteriores à década de 90 comumente não tinham uma miniaturização boa o suficiente para comportar a unidade de ponto flutuante.

Essa unidade era comumente vendida como um chip separado, chamado de coprocessador de ponto flutuante.

Um pouco de história

Processadores anteriores à década de 90 comumente não tinham uma miniaturização boa o suficiente para comportar a unidade de ponto flutuante.

Essa unidade era comumente vendida como um chip separado, chamado de coprocessador de ponto flutuante.

Na maioria dos processadores atuais, esse coprocessador fica no mesmo chip da CPU, mas o nome persiste.

Exemplo: a instrução do MIPS de mnemônico lwc1 significa “Load Word to Coprocessor 1”.

Um pouco de história

Na família de processadores x86, o primeiro “coprocessador matemático” era o 8087.

Dessa forma, a FPU do x86 ficou conhecida por x87.

Até nos dias de hoje, é comum encontrar nos manuais as instruções do x86-64 as instruções de ponto flutuante como *Conjunto de Instruções x87*.

RISC-V

O RISC-V adiciona instruções de ponto flutuante através das extensões RV32F, RV64D.

https://msyksphinz-self.github.io/riscv-isadoc/html/rvfd.html#

Registradores f0 até f31 são utilizados para operações de ponto flutuante.

https://msyksphinz-self.github.io/riscv-isadoc/html/regs.html?highlight=floating%20point

Aritmética de Ponto Flutuante

Como visto na aula passada, dada a representação, cálculos podem gerar erros de precisão.

Devido ao arredondamento, a quantidade de ciclos necessários para se efetuar o cálculo também pode variar.

Detalhes sobre os erros e como as operações são feitas são dados em:

- Patterson e Hennessy (2020).
- Null e Lobur (2009).

Em Ruggiero e Lopes (1998) é discutido o erro numérico causado pela aritmética de ponto flutuante.

- Tratado ainda na disciplina *Introdução à Computação Científica*.

Aritmética de Ponto Flutuante

Os erros numéricos causados pela aritmética de ponto flutuante não serão tratados nessa disciplina.

Mas vale a pena realizar alguns experimentos e discutir os problemas, mesmo que de maneira informal.

Falha Catastrófica

25 Fev. 1991, Guerra do Golfo.

Sistema antimísseis Patriot.

O sistema falhou ao tentar interceptar um míssil Scud saudita lançado contra uma base americana.

28 Pessoas morreram e 100 ficaram feridas.

Falha Catastrófica

Causa: sistemas de radar dependem fortemente do tempo (de relógio).

O tempo era computado a cada 0.1 segundos, e armazenado utilizando 24 bits.

Aprendemos que isso não pode ser representado perfeitamente em binário.

O erro se acumulou durante 24 horas, até que não foi mais possível calcular com precisão a rota dos mísseis.

https://www-users.cse.umn.edu/~arnold/disasters/patriot.html
http://www.cs.unc.edu/~smp/COMP205/LECTURES/ERROR/lec23/node4.html
https://www-users.cse.umn.edu/~arnold/disasters/Patriot-dharan-skeel-siam.pdf

Dica de leitura

Parker, M. Humble Pi: A Comedy of Maths Errors. Reino Unido: Penguin Books Limited. 2019.

Vamos perguntar aos especialistas

Vamos perguntar aos “especialistas” da internet.

Dois Exemplos:

- www.guj.com.br/t/erro-de-precisao-float-e-double/39142
- computingat40s.wordpress.com/java-float-and-double-primitive-types-are-evil-dont-use-them

O que está terrivelmente errado?

Vamos perguntar aos especialistas

Vamos perguntar aos “especialistas” da internet.

Dois Exemplos:

www.guj.com.br/t/erro-de-precisao-float-e-double/39142

computingat40s.wordpress.com/java-float-and-double-primitive-types-are-evil-dont-use-them

O que está terrivelmente errado?

Tudo! (ps.: procurei por “problemas com float” no Google).

Se encontrar mais pérolas, me envie por e-mail.

Ponto Fixo

Quando precisamos garantir que os valores são representados de maneira exata, podemos utilizar bibliotecas para ponto fixo, ou criar a nossa própria biblioteca.

Ideia: Representar os valores antes de depois da vírgula de maneira separada, utilizando múltiplos bytes, se necessário.

Ponto Fixo

Quando precisamos garantir que os valores são representados de maneira exata, podemos utilizar bibliotecas para ponto fixo, ou criar a nossa própria biblioteca

Ideia: Representar os valores antes de depois da vírgula de maneira separada, utilizando múltiplos bytes, se necessário.

Pelo menos 1 byte para armazenar o 3

Exemplo 3,402823466385288598117041834845169254 40

Pelo menos 16 bytes para armazenar a parte decimal como um “inteiro”.

Ponto Fixo

Quais são as vantagens e desvantagens do ponto fixo?

Ponto Fixo

Quais são as vantagens e desvantagens do ponto fixo?

+ Podemos armazenar valores com qualquer precisão (desde que a memória seja o suficiente).
+ Os resultados obtidos são fiéis aos que um humano (que trabalha no mundo decimal) espera.
- Custo de memória.
- Custo de processamento.

Ponto Fixo E Precisão Arbitrária

Existem bibliotecas prontas nas mais variadas linguagens para lidar com valores em ponto fixo e precisão arbitrária (Você pode especificar quantas casas decimais precisa).

Exemplos:

C++

Boost.Multiprecision -› www.boost.org/doc/libs/1_66_0/libs/multiprecision/doc/html/index.html
GNU Multiple Precision Arithmetic Library -› https://gmplib.org

Java

BigDecimal

C#

Decimal

Veja uma lista -› en.wikipedia.org/wiki/List_of_arbitrary-precision_arithmetic_software

Quando usar

Utilizar ou não o IEEE 754 depende do problema que você tem em mãos.

Na grande maioria dos problemas, os pequenos erros numéricos introduzidos pelo ponto flutuante são irrelevantes.

Em alguns problemas podemos precisar utilizar bibliotecas para ponto fixo ou de precisão arbitrária.

Quando usar

Utilizar ou não o IEEE 754 depende do problema que você tem em mãos.

Na grande maioria dos problemas, os pequenos erros numéricos introduzidos pelo ponto flutuante são irrelevantes.

Em alguns problemas podemos precisar utilizar bibliotecas para ponto fixo ou de precisão arbitrária.

Muitos “especialistas” da internet e afins dizem para sempre usar essas bibliotecas.

Péssima ideia.

Custa muito caro e pode introduzir ainda mais erros numéricos se não tomarmos cuidado.

Teste Você Mesmo

Os exemplos a seguir estão em Java.

Compilar

javac MainDouble.java

Será gerado um arquivo de classe jamado MainDouble.class

Executar

time java MainDouble

O comando time estima o tempo de execução do programa.

Veja a saída “real” do comando.

Teste Você Mesmo - Double

```
public class MainDouble{
   public static void main(String args[]){
       //teste com doubles
       double teste = 0.0;
       for(int i=0; i < 1000000; i++){//fazendo 1M vezes para poder medir o tempo
           teste = 0.1;
           for(int j=0; j<9; j++)
               teste+=0.1;
       }
       System.out.println("Resultado com double: " + teste);

       System.out.println("Memória Utilizada (MB): " + (Runtime.getRuntime().totalMemory() -
Runtime.getRuntime().freeMemory())/1024/1024);
   }
}
```

Teste Você Mesmo - BigDecimal

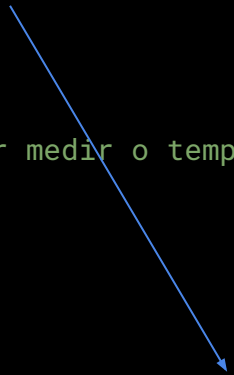
```
import java.math.BigDecimal;

public class MainBigDecimal{
   public static void main(String args[]){
       BigDecimal zeroUm = new BigDecimal("0.1");
       BigDecimal teste = null;
       for(int i=0; i < 10000000; i++){//fazendo 1M vezes para poder medir o tempo
           teste = zeroUm;
           for(int j=0; j<9; j++)
               teste=teste.add(zeroUm);
       }

       System.out.println("Resultado com BigDecimal: " + teste);
       System.out.println("Memória Utilizada (MB): " + (Runtime.getRuntime().totalMemory() -
Runtime.getRuntime().freeMemory())/1024/1024);
   }
}
```

Teste Você Mesmo - BigDecimal

Atenção: Essa é apenas uma estimativa da memória utilizada. O coletor de lixo pode passar no meio do caminho. A quantidade de memória realmente utilizada pode ser **muito** maior.



```
import java.math.BigDecimal;

public class MainBigDecimal{
   public static void main(String args[]){
       BigDecimal zeroUm = new BigDecimal("0.1");
       BigDecimal teste = null;
       for(int i=0; i < 10000000; i++){//fazendo 1M vezes para poder medir o tempo
           teste = zeroUm;
           for(int j=0; j<9; j++)
               teste=teste.add(zeroUm);
       }

       System.out.println("Resultado com BigDecimal: " + teste);
       System.out.println("Memória Utilizada (MB): " + (Runtime.getRuntime().totalMemory() -
Runtime.getRuntime().freeMemory())/1024/1024);
   }
}
```

Resultados

Com Double:
Resultado com double: 0.9999999999999999
Memória Utilizada (MB): 2

real 0m0,068s

Com BigDecimal
Resultado com BigDecimal: 1.0
Memória Utilizada (MB): 146

real 0m0,653s

Gasto energético desnecessário (da máquina executando, refrigeração, ...)

Venus

O Simulador Venus possui um suporte bastante limitado ao IEEE 754.

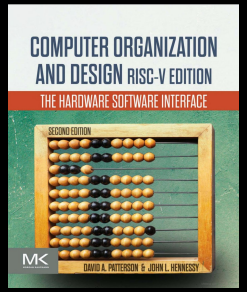
Cuidado!

Exercícios

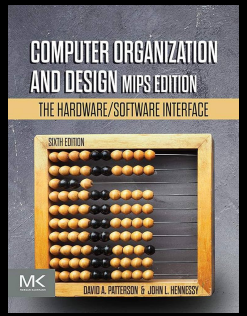
1. Assista a esse vídeo: https://www.youtube.com/watch?v=PZRI1IfStY0
2. Some dois valores em ponto flutuante quaisquer utilizando assembly do RISC-V. Depois, armazene o resultado em um registrador de inteiros. Exiba o resultado (inteiro) na tela.
3. É comum a utilização de bibliotecas de ponto fixo ou de precisão arbitrária para representar valores monetários. Você consegue pensar numa solução melhor e mais eficiente do que isso para, por exemplo, representar o salário de uma pessoa?

Referências

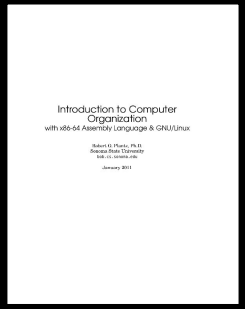
Patterson, Hennessy. Computer Organization and Design RISC-V Edition: The Hardware Software Interface. 2020.



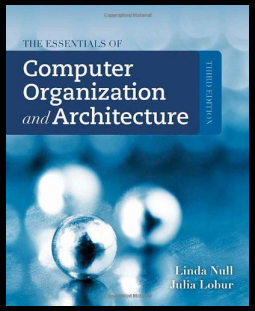
Patterson, Hennessy. Computer Organization and Design MIPS Edition: The Hardware/Software Interface. 2020.



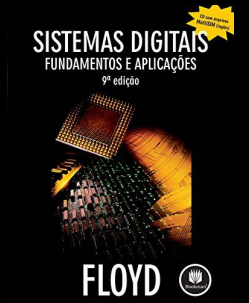
Plantz. Introduction to Computer Organization with x86-64 Assembly Language & GNU/Linux. 2011.



Null, Lobur. The Essentials of Computer Organization and Architecture. 2014.



Floyd. Sistemas Digitais: Fundamentos e Aplicações. 2009.



Licença

Esta obra está licenciada com uma Licença Creative Commons Atribuição 4.0 Internacional.