



Fundamentos de Arquitetura de Computadores

Trabalho 03

Prof. Tiago Alves

Programação em Linguagem de Montagem MIPS: Aritmética em Ponto Flutuante

Introdução

A disciplina de Fundamentos de Arquitetura de Computadores trata de diversos tópicos que nos ajudam a compreender como sistemas eletrônicos de computação são construídos. Esse tipo de conhecimento ajudará profissionais de áreas afetas a tecnologias de informação e comunicação a aplicarem, adequadamente, um computador digital na realização de tarefas que, devido à sua natureza, serão melhores conduzidas por um sistema automatizado.

Além de identificar a conveniência da aplicação dos computadores digitais, a disciplina ajudará a desenvolver competências necessárias para a solução de problemas em sistemas computacionais em operação, principalmente problemas decorrentes de análise de desempenho.

Para construir ou adicionar funcionalidades a esses sistemas computacionais, é necessário conhecimento de linguagens de programação e ferramentas de desenvolvimento. Em nosso curso, o domínio de linguagens de montagem é um pré-requisito para o devido acompanhamento das atividades da disciplina.

Objetivos

- 1) Exercitar conceitos da linguagem de montagem para arquitetura MIPS, especialmente aqueles referentes à implementação de solução de problemas em aritmética de ponto flutuante.
- 2) Interagir com ferramentas de desenvolvimento para criação, gerenciamento, depuração e testes de projeto de aplicações.

Referências Teóricas

Mitchell, Mark, Jeffrey Oldham, and Alex Samuel. Advanced linux programming. New Riders, 2001.

Material Necessário

- Computador com sistema operacional programável
- Ferramentas de desenvolvimento GNU/Linux ou similares: MARS ou SPIM.



Roteiro

- 1) Revisão de técnicas e ferramentas de desenvolvimento usando linguagem de montagem MIPS.

Colete o material acompanhante do roteiro do trabalho a partir do Moodle da disciplina e estude os princípios e técnicas de desenvolvimento de aplicações usando linguagem de montagem MIPS

- 2) Realizar as implementações solicitadas no questionário do trabalho.

Implementações e Questões para Estudo

- 1) Escreva um programa em linguagem de montagem para MIPS usando, preferencialmente, o simulador MARS como plataforma de desenvolvimento e validação. A sua aplicação deverá calcular a **raiz quadrada (sqrt)** de um número inteiro positivo ($X > 1$). Seguem os requisitos de implementação:
 - Sua aplicação deverá receber em entrada em console dois números inteiros positivos.
 - Espera-se que o algoritmo calcule a raiz quadrada do primeiro número (X) de forma que o erro de aproximação seja menor que $1,0 \cdot 10^{(-E)}$, onde E ($E > 0$) é o segundo inteiro apresentado na entrada. E representa a ordem de grandeza do erro de aproximação. Uma forma de entender o E é considerá-lo como a quantidade de casas decimais corretas da aproximação. Lembre-se que $\text{sqrt}(X)$ pode ser um número irracional.
 - Como saída, o programa deverá imprimir dois números: o número real com o valor aproximado da raiz quadrada de X ($\text{sqrt}(X)$) e um número inteiro N , que informa quantas iterações foram necessárias para o algoritmo da bissecção atender o requisito de erro de aproximação menor que $1,0 \cdot 10^{(-E)}$.
 - O algoritmo a ser utilizado no cálculo da raiz quadrada deverá ser baseado no seguinte teorema: *Se $x = a \times b$, então \sqrt{x} está entre a e b .*
 - A ideia é usar o fato descrito no teorema repetidamente, com diversos pares de números a e b , como $(a_1, b_1), (a_2, b_2), \dots$, de tal maneira que os pares vão ficando cada vez mais próximos da raiz procurada e, assim, os decimais da raiz vão sendo descobertos.
 - Para um exemplo didático de aplicação do teorema, vejamos cálculo de $\text{sqrt}(130)$:
 - Os dois inteiros $a_0 = 11$ e $b_0 = 12$ são tais que o seus quadrados são, respectivamente, menor e maior que 130: $11 \cdot 11 = 121 < 130$ e $12 \cdot 12 = 144 > 130$.
 - É possível afirmar que a raiz de 130 é um número real maior que 11 e menor que 12. Passemos ao refinamento das casas decimais.
 - Tomemos $a_1 = a_0 = 11$. Usando o teorema acima, b_1 tem que ser tal que $a_1 \times b_1 = 130$. Façamos $b_1 = \frac{130}{a_1} = \frac{130}{11} = 11.818181\dots$ Perceba que $\text{sqrt}(130)$ está realmente entre a_1 e b_1 .
 - Para usar o teorema acima em toda a sua plenitude, conforme faziam os Sumérios 3400 anos atrás, o próximo passo é bastante simples. O segundo par (a_2, b_2) deve ser formado da seguinte forma:
 - a_2 deve ser feito igual à média de a_1 e b_1 :



$$a_2 = \frac{a_1 + b_1}{2} = 11.4090909090909$$

- b_2 segue de acordo com a primeira aproximação b_1 :

$$b_2 = \frac{130}{a_2} = 11.394422310757$$

- Continuando o processo, o terceiro par seria:

- $a_3 = \frac{a_2 + b_2}{2} = 11.4017566099239$

- $b_3 = \frac{130}{a_3} = 11.4017518920593$

- Repetindo-se o processo, é possível afirmar que $b_n = \sqrt{130} = 11.4017542509911$. Para o nosso exercício, a medida do erro de aproximação será dada pelo valor absoluto de $(b_n - a_n)$. Para $n=4$, é possível afirmar que o erro de aproximação é menor que $1,0 \cdot 10^{(-12)}$, ou seja 12 casas decimais da aproximação estão corretas.
- As mensagens de saída deverão ser mostradas de acordo com os seguintes formatos:
 - Entradas invalidas.
 - A raiz quadrada de X eh ZZ, calculada em N iteracoes.
 - Não foi possível calcular sqrt(X).
- Na sua implementação, esperam-se encontrar as funções:
 - `encontra_inteiros`, que encontrarão os inteiros a_0 e b_0 . Esses inteiros são tais que os seus quadrados a_0^2 e b_0^2 são, respectivamente, menor e maior que X.
 - `calc_raiz`, que calculará a raiz de acordo com o critério de parada que requer erro de aproximação menor que $1,0 \cdot 10^{(-E)}$.
 - `imprime_saida`, função que imprimirá o resultado bem sucedido.
- Caso o algoritmo não convirja, ou seja, mais de 100 iterações tenham sido executadas sem que o critério de parada (erro de aproximação menor que $1,0 \cdot 10^{(-E)}$) tenha sido atendido, uma mensagem de erro deverá ser apresentada. Outras funções poderão ser criadas, ficando a critério da equipe de implementação.
- Exemplos:
 - Exemplo de invocação 1:
78
6
A raiz quadrada de 78 eh 8.831760865699747, calculada em 4 iteracoes.
 - Exemplo de invocação 2:
2
16
Nao foi possivel calcular sqrt(2).
 - Exemplo de invocação 3:
1
5
Entradas invalidas.
 - Exemplo de invocação 4:
16
5
A raiz quadrada de 16 eh 3.999998612268036, calculada em 4 iteracoes.



- Exemplo de invocação 4:
100
-2
Entradas invalidas.

Instruções e Recomendações

A submissão das respostas aos problemas dos trabalhos deverá ser feita através do Moodle da disciplina.

Cada Problema do Trabalho 03 deverá ser entregue em um pacote ZIP. A dupla de alunos deverá nomear o pacote ZIP da seguinte forma: nome_sobrenome_matricula_nome_sobrenome_matricula_**trab03.zip**.

Entre os artefatos esperados, listam-se:

- códigos-fonte C das soluções dos problemas;
- documentação mínima da aplicação:
 - o qual sistema operacional foi usado na construção do sistema;
 - o qual ambiente de desenvolvimento foi usado;
 - o quais são as telas (instruções de uso)
 - o quais são as limitações conhecidas

Não devem ser submetidos executáveis.

Códigos-fonte C com erros de compilação serão desconsiderados (anulados).

Os trabalhos poderão ser realizados em duplas; a identificação de cópia ou plágio irá provocar anulação de todos os artefatos em recorrência.