Trabalho - Assembly do Processador MIPS

Implementação da Função Raiz Quadrada usando o Método Newton-Raphson Recursivo

Este documento descreve o trabalho da disciplina de Organização e Arquitetura de Processadores que consiste na compreensão de um problema algorítmico, descrição deste problema em linguagem de alto nível, compreensão da ISA do MIPS e mapeamento do algoritmo no assembly do MIPS considerando todo o ISA (*Instruction Set Architecture*).

A atividade envolve trabalhar com técnicas e fundamentos apreendidos na disciplina de OAP; dentre estes estão (*i*) a programação em linguagem de máquina do MIPS, (*ii*) a implementação algorítmica com o uso de função e, possivelmente macros, (*iii*) o salvamento e recuperação de registradores em pilha ao trabalhar com funções, (*iv*) o emprego de recursividade e (*v*) o interfaceamento do programa com o sistema operacional.

A atividade, que deverá ser realizada em ***grupos de até 4 alunos***, envolve também o emprego do ambiente MARS para descrição e verificação do comportamento do algoritmo, e uma documentação adequada que apresente o desenvolvimento de todas as atividades requisitadas.

1. Especificação Técnica do Trabalho

O grupo deve implementar uma implementação específica do método numérico conhecido como Newton-Raphson, mas com uma particularidade: esta versão deve ser recursiva e tem o objetivo específico de **calcular a raiz quadrada de *X*, um número inteiro e positivo**. O método de Newton-Raphson, fundamentado no princípio matemático da linearização de funções, é uma técnica **iterativa** que visa encontrar as raízes de uma função por meio das aproximações sucessivas.

A essência do método recursivo é que, em vez de usar laços de repetição comuns em implementações iterativas, a função chama a si mesma com novos parâmetros, sendo que a cada nova chamada da função o número de iterações é decrementado. Isso torna o código mais compacto e conceitualmente simples. Dessa forma, a implementação recursiva do método Newton-Raphson para cálculo da raiz quadrada representa uma abordagem elegante e eficaz na solução de um problema comum de cálculo numérico.

A regra de recursividade que implementa o cálculo da raiz quadrada utilizando o método Newton-Raphson é a seguinte:

$$\sqrt{\text{¿}}nr(x,i)=\begin{cases} 1, \wedge i=0 \\ \dfrac{\sqrt{\text{¿}}nr(x,i-1)+\dfrac{x}{\sqrt{\text{¿}}nr(x,i-1)}}{2}, \wedge i>0 \end{cases}$$

Onde *x* é o valor que desejamos encontrar a raiz quadrada, e *i* é o número de iterações que o método irá executar. Por exemplo, se executarmos a função para *x* = 100 e *i* = 3 a função retornará a resposta *sqrt_nr*(100, 3) = 14, pois não houveram iterações suficientes para que o resultado convergisse. Agora, para os valores de *x* = 100 e *i* = 5, temos: *sqrt_nr*(100, 5) = 10, ou seja, o

resultado convergiu adequadamente pois $\sqrt{100} = 10$.

Abaixo segue uma tabela que mostra a evolução do método Newton-Raphson para alguns valores de *x* e *i*.

		x							
		16	**64**	**100**	**256**	**400**	**900**	**1200**	**2500**
i	**1**	8	32	50	128	200	450	600	1250
	2	5	17	26	65	101	226	301	626
	3	4	10	14	34	52	114	152	314
	4	4	8	10	20	29	60	79	160
	5	4	8	10	16	21	37	47	87
	6	4	8	10	16	20	30	36	57
	7	4	8	10	16	20	30	34	50
	8	4	8	10	16	20	30	34	50
	9	4	8	10	16	20	30	34	50
	10	4	8	10	16	20	30	34	50

1.1. Detalhamento da Especificação

O programa deve apresentar o valor da função de Newton-Raphson para um par de inteiros (*x*, *i*) lidos da entrada padrão, sendo que *x* corresponde ao número que se deseja descobrir a raiz quadrada e *i* o número de iterações do método. Exibindo na saída padrão o resultado da função.

O programa deve ser implementado com funções, tendo pelo menos as duas funções descritas a seguir e duas macros quaisquer. Contudo, o grupo pode implementar outras funções e macros, de forma a tornar o programa mais modular e legível. O uso correto de macros e funções será considerado na avaliação do trabalho!

(i) Uma função recursiva para cálculo do valor de Newton-Raphson em relação a *x* e *i*;

(ii) Uma função principal (main).

Dica de Implementação:

Para realizar a divisão de dois números inteiros podemos usar a instrução `div`, tal como exemplificado abaixo:

```
div $t0, $t1
```

Neste exemplo a instrução `div` faz a divisão do conteúdo do registrador `$t0` pelo conteúdo do registrador `$t1` e o resultado da divisão é armazenado em um registrador especial chamado *LO* (`LO = $t0/$t1`). Além disso, o resto da divisão é colocado em outro registrador especial, denominado *HI* (`HI = $t0 % $t1`).

Para obter os conteúdos dos registradores LO e HI existem duas instruções `mflo` e `mfhi`, exemplificadas abaixo:

```
mflo $t2
mfhi $t3
```

Usando as instruções acima, o resultado inteiro da divisão será armazenado em `$t2` e o resto da divisão em `$t3`. Ficando disponível para manipulação pelas instruções que trabalhamos em aula.

Ah! Não se esqueça que a divisões por potências de dois (2, 4, 8, 16, ...) podem ser obtidas

através de deslocamentos (instruções sra ou srl)!

1.2. Detalhamento da Interface com o Usuário

O programa a ser entregue deve conter as seguintes funcionalidades:

1) Iniciar a execução apresentando a seguinte mensagem:

   "Programa de Raiz Quadrada – Newton-Raphson"

   "Desenvolvedores: <Lista de nomes dos alunos>"

2) Entrar em um laço de execução que somente termina quando for pressionado ***um número negativo***.

   "Digite os parâmetros *x* e *i* para calcular *sqrt_nr* (*x, i*) ou -1 para abortar a execução"

   (i) Caso o usuário digitar ***um número negativo***, *seja para* ***x*** *ou para* ***i****, o programa encerra.*

   (ii) Caso o usuário não digitar um número negativo:

      a. O programa deve utilizar os dois inteiros e calcular a função recursiva.

      b. Ao terminar o cálculo da função, o programa deve retornar o resultado em um formato similar ao descrito a seguir:

         "sqrt(500, 8) = 22"

   (iii) Retornar para executar um novo laço.

2. Entregas

As principais atividades a serem realizadas e comprovadas através de uma documentação adequada estão descritas a seguir:

1) Algoritmo descrevendo o programa de alto nível (linguagem Java, C, português estruturado, ...). O programa deve conter as funções especificadas na descrição, considerando a recursividade requisitada;
2) Descrição em linguagem assembly do MIPS equivalente ao programa de alto nível descrito em "1";
3) Captura de telas do MARS mostrando:
   a. A área de código montada;
   b. O estado dos registradores ao término de uma execução;
   c. A área de pilha utilizada para a recursividade; e
   d. Um exemplo de execução do programa.

O grupo deve entregar as atividades descritas acima em um arquivo compactado contendo uma documentação em formato pdf. Adicionalmente, o programa assembly que deve estar no documento, também tem que ser colocado dentro do arquivo compactado para possível verificação de seu funcionamento no ambiente MARS.