Resolução de equações quadráticas em assembly
Organização de Computadores 1

Carolina Adilino, Ana Julia Botega

Departamento de Informática e Estatística – Universidade Federal de Santa Catarina
Caixa Postal 5094 – 88035-972 – Florianópolis – SC – Brasil

Resumo. *Esse trabalho visa apresentar o programa desenvolvido pelos autores, cujo objetivo é a resolução de equações quadráticas através de operações de alto-nível em Assembly do MIPS, para ser executado no simulador MARS. Tal resultado é obtido através da realização das operações aritméticas simples de modo a compor a equação de Bhaskara. Seguindo as diretrizes do projeto, o programa recebe 3 números inteiros, os coeficientes 'a', 'b' e 'c' da equação quadrática cujas raízes se desejam calcular. Como saída, são apresentados dois resultados, denominados x1 e x2, sendo essas as corretas raízes da respectiva equação. Durante a execução do código, os comandos e operações foram realizados corretamente, resultando sempre em um resultado satisfatório. Assim, consideramos que o programa foi bem-sucedido.*

Palavras-chave: *Assembly, MIPS, Equação quadrática, Fórmula de Bhaskara*

(colocar aqui que é a atividade 1)

1. Parte Introdutória

A linguagem assembly é usada para a programação em baixo nível, devido à sua forte correspondência entre as instruções usadas no código e as instruções usadas no código da máquina. Embora mais complicada que linguagens de alto-nível, seu aprendizado é essencial no entendimento de um programador sobre sistemas operacionais e design de compiladores.

Esse relatório tem como objetivo detalhar o processo e resultado do programa desenvolvido pelos autores. Através desse código e de sua documentação, é possível realizar as operações necessárias para chegar ao resultado desejado, sendo esse as raízes de uma equação

quadrática. Também é importante reforçar que o trabalho segue as diretrizes exigidas, sendo elas, para o primeiro exercício: As variáveis (a, b, c) devem estar previamente definidas e armazenadas na memória de dados, assim como a variável x que deverá armazenar na memória o resultado final; Os conteúdos destas variáveis armazenadas na memória podem ser quaisquer valores; O programa deverá trabalhar apenas com números inteiros; E as diretrizes do segundo exercício sendo: O resultado final deve ser apresentado no terminal e também armazenado na variável x, na memória de dados; Utilize chamadas de sistema (syscall) para realizar a entrada (teclado) e saída (tela) de dados; O programa deverá trabalhar apenas com números inteiros;

1. Objetivo

Escrever um código em assembly que realize operações de aritmética simples de modo em que seja capaz de calcular as raízes de uma equação quadrática através da fórmula de Bhaskara.

2. Materiais e métodos

O código assembly foi desenvolvido para arquitetura MIPS e executado na plataforma MARS (MIPS Assembler and Runtime Simulator).

A primeira versão do código tem como objetivo realizar o cálculo com valores previamente definidos e armazenados na memória de dados. Para nossa execução, foram definidos os coeficientes como a = 1, b = -5, e c = 6; Essas constantes foram definidas no segmento de dados do código, denominado ".data". O resto do código se encontra no segmento ".text", em sua única seção, "main".

O primeiro passo consiste no carregamento das variáveis a, b e c nos registradores $t1, $t0 e $t2, respectivamente. Em seguida, esses registradores foram usados para calcular o conteúdo de dentro da raiz quadrada, $b^2 - 4ac$. Foi realizada uma instrução por vez, seguindo a ordem de prioridade definida pela matemática, como ilustrada na figura 1. Começando pela multiplicação de $t0 por ele mesmo, através da instrução mul, e guardando esse resultado no registrador $t4. Em seguida, 4ab foi calculado. Para isso, primeiro foi usada a instrução sll para realizar um shift de duas casas em $t1, assim multiplicando *a* por 4 e armazenando seu resultado em $t5. A multiplicação desse valor por *c* ocorreu através da instrução "mul $t6, $t2, $t5". Com os resultados de b^2 e 4ac armazenados nos registradores $t4 e $t6, respectivamente, faltava

apenas a subtração desses valores. Tal operação foi realizada com o comando sub, e teve seu resultado armazenado em $t7.

```
15      #Calculando delta
16        mul       $t4, $t0, $t0    #b²
17        sll       $t5,$t1, 2       #4a
18        mul       $t6, $t2, $t5    #4a . c
19        sub       $t7, $t4, $t6    #b² - 4ac
20
```

Fig. 1 - Cálculo do delta

O próximo passo na equação quadrática é calcular a raiz quadrada do delta. Para isso, mostrou-se necessário realizar uma conversão de tipo, considerando que o comando sqrt.s aceita apenas números float. Essa conversão foi realizada com o comando mtc1, que converteu o valor armazenado em $t7 para float e o armazenou em $f1. Possibilitada a instrução, através de "sqrt.s $f2, $f1" agora temos o valor da raiz quadrada de delta armazenada em $f2. Considerando as diretrizes do trabalho, nos é exigido converter o resultado de volta para um número inteiro. Isso, então, é realizado com comando "mfc1 $t8, $f3". Assim, agora temos o resultado da raiz quadrada de delta como um inteiro no registrador $t8.

Agora, poucas operações nos separam dos resultados finais. Primeiramente, através do comando neg, armazenamos em $t9 o oposto de b. Em seguida, é calculado 2.a, fazendo um shift para a esquerda de uma casa. Agora, apresentamos todos os valores necessários para chegar às respostas. A primeira raiz é calculada através da soma de -b com o resultado da raiz de delta, que está armazenado no registrador $t8. Para isso, é usado o comando add, e seu resultado é armazenado em $t3. O último passo para calcular essa raíz é "div $t5, $t3, $t4", dividindo o valor que acabamos de calcular pelo dobro de a. Para calcular a segunda raíz, todos os passos são repetidos, com a exceção da soma de -b com a raiz de delta. Seguindo a equação, essa instrução é substituída pela subtração desses dois valores.

Em conclusão, todas as operações necessárias são realizadas em sua devida ordem. Nossos resultados se apresentam nos registradores $t5 e $t6.

A próxima atividade é uma adaptação do programa recém explicado, que, ao invés de valores fixos, recebe as variáveis *a*, *b* e *c* do usuário e depois exibe as raízes no terminal.

A primeira diferença entre os códigos se encontra no segmento ".data" de dados do código. Aqui, ao invés da definição de valores dos coeficientes, declaramos as strings que serão mostradas no terminal durante a execução do programa. São chamadas "mem1", "mem2", e "mem3" as strings que pedem, respectivamente, os coeficientes *a*, *b* e *c*.

Seguindo, no segmento de texto ".text", começamos o código carregando ao registrador $v0 o número 4, que corresponde ao comando de escrever string. Assim, o valor de mem1 é escrito no terminal. Em seguida, através do comando de ler inteiro e move, o valor escrito pelo usuário é salvo em $t1. Esse código é, então, repetido com as strings armazenadas em "mem1" e "mem2", recebendo do usuário os valores de *b* e *c,* que serão armazenados em $t0 e $t2, respectivamente. As linhas do código que correspondem à primeira variável sendo lida do teclado podem ser lidas na figura 2.

```
18          li        $v0, 4       #Comando
19          la        $a0, mem1    #Carrega string (endereço).
20          syscall
21
22          li        $v0, 5       #Comando para ler inteiro.
23          syscall
24          move      $t1, $v0
```

Figura 2 - início do segmento .text da segunda atividade

As próximas etapas do código serão idênticas às previamente mencionadas no detalhamento da primeira atividade.

As diferenças entre as duas atividades surgem no final apenas, onde agora nos é exigido que a saída do programa seja apresentada no terminal. Para isso, novamente é utilizado o comando "escrever string". Primeiramente, para a string "Resultado de X1 é", que está armazenada em "memresult1" e, em seguida, para o resultado é usado o comando "escrever inteiro". Essas operações são repetidas para a segunda raiz. Assim, todas as etapas são concluídas, e realizadas com êxito.

3. Resultados e discussões

A execução do programa foi bem-sucedida, com todas as etapas ocorrendo de maneira esperada. Durante a fase de desenvolvimento, o valor de todos os registradores foi acompanhado e os resultados obtidos pelo código foram conferidos.

Uma análise interessante é a discrepância entre a quantidade de linhas de código existentes no programa. Como está evidenciado na tabela 1, tanto no primeiro programa quanto no segundo, a quantidade de linhas do código em Basic é significativamente maior que o código source. Isso se dá devido ao uso de pseudo instruções. Esse termo se refere às instruções que teoricamente não existem, elas são a combinação de mais de uma instrução, ocupando mais de uma linha no montador.

Programa	Basic	Source
1	28	19
2	61	49

Tabela 1- Quantidade de linhas de código Basic e Source dos dois programas

4. Conclusões

Esse trabalho foi desenvolvido com o objetivo de realizar operações aritméticas simples, a fim de calcular as raízes de uma equação quadrática usando a arquitetura de MIPS no programa de desenvolvimento MARS. Tal meta foi alcançada com êxito, de modo que os dois programas foram escritos e executados sem erros.

O desenvolvimento desse código foi um ótimo auxílio para o melhor entendimento de como a arquitetura do MIPS é organizada. Além de uma melhora nas habilidades de uso da plataforma MARS, também facilitou a fluência em assembly.

As habilidades adquiridas com esse simples projeto são uma forte evidência da importância do aprendizado de linguagens de baixo-nível para o desenvolvimento de um programador. Os conhecimentos aqui desenvolvidos serão utilizados em qualquer futuro projeto e desafio que será futuramente enfrentado no curso.