RELATÓRIO TÉCNICO

Programas sobre Entrada de Dados em Assembly MIPS

Diogo Borges Corso

Departamento de Computação

(48) 3721-6950
Campus Jardim das Avenidas, Rodovia Governador Jorge Lacerda, 3201 - Jardim das Avenidas - Araranguá - SC

UNIVERSIDADE FEDERAL DE SANTA CATARINA

SUMÁRIO

Lista de Figuras

1 INTRODUÇÃO

Este relatório tem como objetivo analisar e documentar o funcionamento de dois programas desenvolvidos em linguagem Assembly, utilizando o simulador MARS. Ambos os códigos foram projetados para realizar operações matemáticas a partir de variáveis armazenadas na memória e manipular essas informações com base nas instruções vistas durante os módulos 7 e 8 da disciplina.

O primeiro programa realiza o cálculo da expressão $c = d^3 - (b + 35 + e)$, considerando que os valores de b, d e e já estão previamente definidos na memória. Já o segundo programa realiza a mesma operação, porém solicitando que o usuário insira os valores de entrada pelo teclado durante a execução do código, utilizando chamadas de sistema (syscall) para entrada e saída de dados no console do MARS.

Este relatório abordará os detalhes de implementação de cada código, os testes realizados para validar os resultados obtidos, a contagem e comparação entre as linhas de código nas colunas Basic e Source do MARS, bem como as conclusões obtidas a partir da análise e execução dos programas.

2 OBJETIVOS

Este relatório tem como principais objetivos:

1. Demonstrar o funcionamento correto dos programas desenvolvidos em linguagem Assembly MIPS, utilizando o simulador MARS. Para isso, serão apresentados os códigos, as descrições das operações realizadas e os resultados obtidos durante a execução.
2. Verificar e comparar a contagem de linhas de código de cada programa. Essa contagem será feita com base nas informações fornecidas pelas colunas Basic e Source, disponíveis na aba Execute do MARS. A comparação entre essas colunas permite observar a quantidade real de instruções interpretadas pela máquina (Basic) em contraste com a quantidade de linhas escritas no código-fonte (Source), que pode incluir comentários, diretivas e espaços em branco.
3. Analisar a diferença entre as duas abordagens implementadas: uma versão com valores definidos estaticamente e uma versão interativa, com entrada de dados via teclado. O objetivo é evidenciar como a inserção de interatividade impacta na complexidade e extensão do código.

3 DESENVOLVIMENTO

3.1 Programa com Valores Fixos na Memória

O primeiro programa foi construído com valores previamente definidos na memória, utilizando diretivas da seção .data. Os valores de entrada são:

- b = 20
- d = 3
- e = 7

A lógica do programa executa os seguintes passos:

1. Carrega o valor de b e soma 35 a ele.
2. Eleva o valor de d ao cubo (isto é, d * d * d).
3. Soma o resultado da etapa 1 com o valor de e.
4. Subtrai o resultado da etapa 3 do valor obtido na etapa 2.
5. Armazena o resultado final na variável c.
6. Exibe o valor de c na tela.

Funcionamento

Ao executar o programa no MARS, o resultado exibido é o valor correto de c, validando que o programa funciona conforme esperado. Prints de tela podem ser utilizados no relatório para comprovar essa execução correta, mostrando tanto a saída no console quanto os valores atualizados na memória.

Contagem de Linhas

- Basic: 70 linhas
- Source: 58 linhas

Existe uma diferença entre o número de linhas nas colunas Basic e Source. Isso ocorre porque a coluna *Source* conta apenas as linhas escritas pelo programador, enquanto a coluna *Basic* contabiliza cada instrução de máquina gerada. Algumas diretivas e instruções simples podem ser traduzidas em múltiplas instruções de máquina (por exemplo, chamadas syscall e instruções que envolvem labels ou carregamento de endereços), o que aumenta o total no *Basic*.

Código Fonte:

```
.data
b:      .word 20
d:      .word 3
e:      .word 7
c:      .word 0

msg_b:  .asciiz "Número fixo b: "
msg_d:  .asciiz "Número fixo d: "
msg_e:  .asciiz "Número fixo e: "
msg_res:.asciiz "Resultado: "
newline:.asciiz "\n"

.text
.globl main

main:
    lw $t0, b

    li $t2, 35
    add $t1, $t0, $t2

    lw $t3, d

    mul $t4, $t3, $t3
    mul $t4, $t4, $t3

    lw $t5, e

    add $t6, $t1, $t5

    sub $t7, $t4, $t6

    sw $t7, c

    li $v0, 4
    la $a0, msg_b
    syscall
```

```
    move $a0, $t0
    li $v0, 1
    syscall

    li $v0, 4
    la $a0, newline
    syscall

    li $v0, 4
    la $a0, msg_d
    syscall

    move $a0, $t3
    li $v0, 1
    syscall

    li $v0, 4
    la $a0, newline
    syscall

    li $v0, 4
    la $a0, msg_e
    syscall

    move $a0, $t5
    li $v0, 1
    syscall

    li $v0, 4
    la $a0, newline
    syscall

    li $v0, 4
    la $a0, msg_res
    syscall

    move $a0, $t7
    li $v0, 1
    syscall

    li $v0, 4
    la $a0, newline
    syscall

    li $v0, 10
    syscall
```

3.2 Programa com Entrada do Usuário

O segundo programa realiza a mesma operação matemática do primeiro, mas permite que o usuário insira os valores de b, d e e durante a execução. Essa abordagem torna o código mais flexível e dinâmico.

A lógica segue os mesmos passos do Programa 1, com a adição de:

1. Impressão de mensagens solicitando ao usuário os valores.
2. Recebimento dos valores via chamadas syscall.
3. Armazenamento desses valores nas variáveis correspondentes.

Funcionamento

Durante a execução no MARS, o usuário é solicitado a digitar os valores de entrada. Após o cálculo, o programa exibe corretamente o valor da variável c, conforme esperado. A interação no console e o valor final podem ser evidenciados por meio de capturas de tela (prints), mostrando o funcionamento real do código.

Contagem de Linhas

- Basic: 47 linhas
- Source: 39 linhas

Assim como no Programa 1, existe uma diferença entre as colunas *Basic* e *Source*. No Programa 2, a diferença é mais acentuada, pois há uma maior quantidade de chamadas de sistema (syscall) para entrada e saída, além de mensagens de texto e carregamento de endereços de memória, o que gera mais instruções de máquina do que linhas escritas no código-fonte.

Código Fonte:

```
.data
b:  .word 0
d:  .word 0
e:  .word 0
c:  .word 0

msg_b: .asciiz "Digite o valor de b: "
msg_d: .asciiz "Digite o valor de d: "
msg_e: .asciiz "Digite o valor de e: "
msg_result: .asciiz "Resultado (c) = "

.text
.globl main

main:
    li $v0, 4
    la $a0, msg_b
    syscall

    li $v0, 5
    syscall
    move $t0, $v0
    sw $t0, b

    li $v0, 4
    la $a0, msg_d
    syscall

    li $v0, 5
    syscall
    move $t1, $v0
    sw $t1, d

    li $v0, 4
    la $a0, msg_e
    syscall

    li $v0, 5
    syscall
    move $t2, $v0
    sw $t2, e

    li $t3, 35
    add $t4, $t0, $t3

    mul $t5, $t1, $t1
    mul $t5, $t5, $t1

    add $t6, $t4, $t2
```

```
sub $t7, $t5, $t6

sw $t7, c

li $v0, 4
la $a0, msg_result
syscall

li $v0, 1
move $a0, $t7
syscall

li $v0, 11
li $a0, 10
syscall

li $v0, 10
syscall
```

4 RESULTADOS

4.1 Programa 1 - Valores Fixos

O primeiro programa utiliza valores pré-definidos na memória para realizar o seguinte cálculo:

$$c = d^3 - (b + 35 + e)$$

Substituindo os valores:

$$c = 3^3 - (20 + 35 + 7) = 27 - 62 = -35$$

Imagens da Simulação:

A seguir, estão apresentadas imagens da simulação, ilustrando:

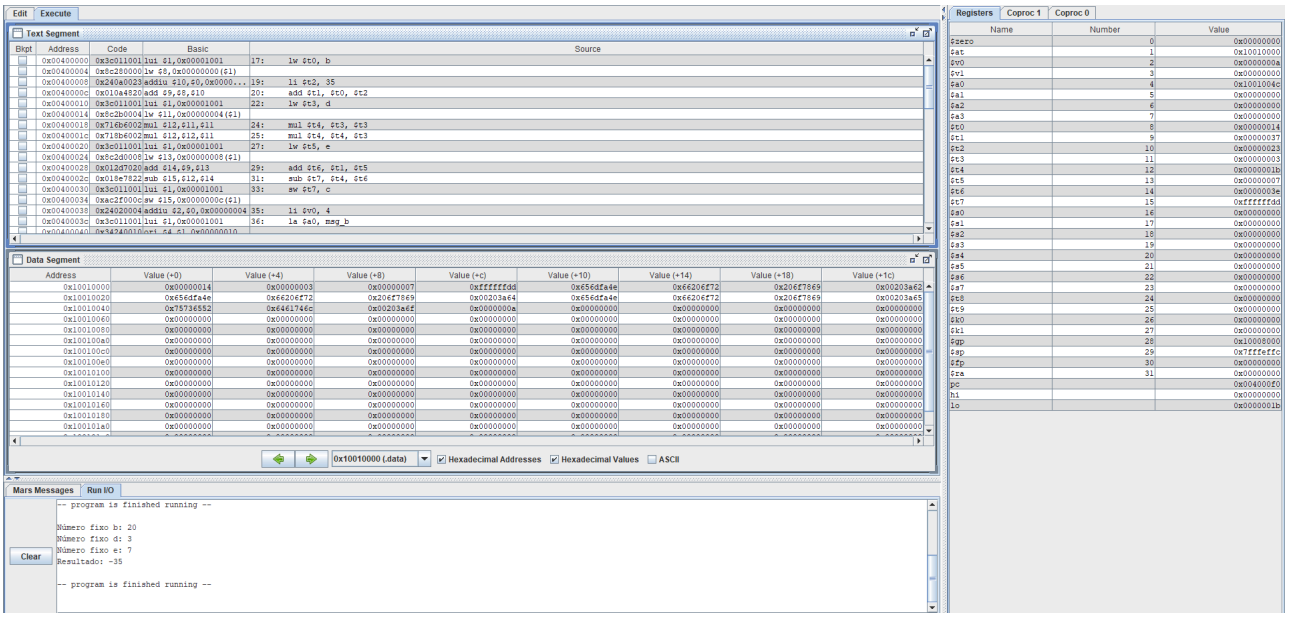

Figura 1 - Execução do programa 1

Fonte: Imagem do autor.

Ao executar o programa no MARS, o valor exibido na tela é -35, exatamente como esperado. A memória também reflete esse valor armazenado corretamente na variável c. A execução ocorre sem erros, validando o funcionamento correto do algoritmo.

4.2 Programa 2 - Entrada do Usuário

No segundo programa, o usuário fornece os valores de b, d e e. Durante os testes, foram inseridos os seguintes valores:

- b = 10
- d = 20
- e = 30

Imagens da Simulação:

A seguir, estão apresentadas imagens da simulação, ilustrando:

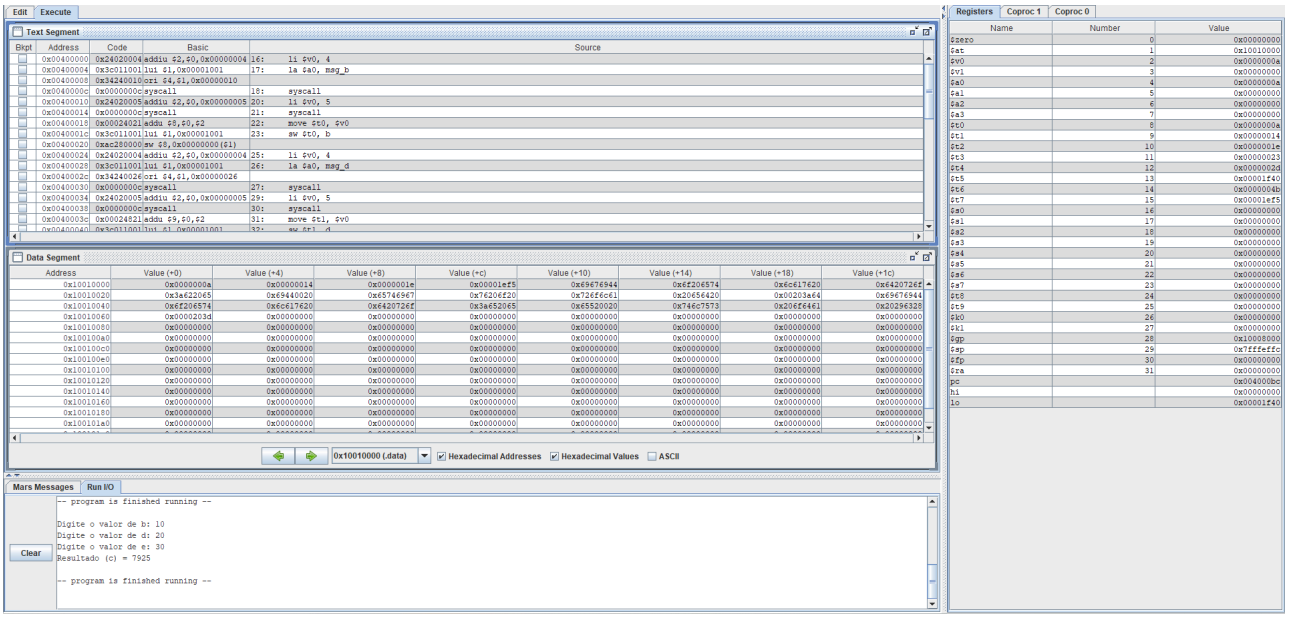

Figura 2 - Execução do programa 2

Fonte: Imagem do autor.

O resultado exibido também foi 7925, confirmando que a lógica do programa está funcionando corretamente com valores fornecidos dinamicamente.

Resumo da contagem de linhas

Programa	Linha Source	Linha Basic	Diferença
Programa 1	58	70	12 linhas
Programa 2	39	47	8 linhas

As diferenças entre as colunas Source e Basic se justificam pelo fato de que instruções simples escritas pelo programador (como syscall ou uso de labels) são traduzidas em múltiplas instruções de máquina, o que aumenta o número de linhas interpretadas pelo MARS na coluna *Basic*.

5 Conclusão

Com a análise e execução dos dois programas desenvolvidos em linguagem Assembly MIPS no simulador MARS, foi possível comprovar o correto funcionamento das instruções implementadas. Ambos os programas realizam a operação aritmética esperada, armazenando e exibindo corretamente o resultado da expressão c = d^3 - (b + 35 + e).

O Programa 1 se mostrou eficiente para testes diretos, pois utiliza valores fixos definidos na memória, permitindo uma execução rápida e automática. Já o Programa 2, com entrada de dados interativa, amplia a aplicabilidade do código ao permitir que diferentes valores sejam testados em tempo de execução, embora isso implique um aumento na complexidade e na quantidade de instruções.

A comparação entre as colunas Basic e Source no MARS evidenciou que a quantidade de instruções de máquina geradas é superior à quantidade de linhas escritas pelo programador. Essa diferença se deve à forma como algumas instruções são convertidas internamente, especialmente chamadas de sistema (syscall) e acessos à memória com la, lw, entre outras.

Por fim, os testes demonstraram que os dois programas cumprem seus objetivos de forma eficiente, reforçando o entendimento sobre operações aritméticas, manipulação de memória, entrada e saída de dados, além de boas práticas na programação em baixo nível com MIPS Assembly.