
**UNIVERSIDADE FEDERAL DE SANTA CATARINA**
**CAMPUS FLORIANÓPOLIS**
**INE-DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA**
**INE5411 - ORGANIZAÇÃO DE COMPUTADORES I**

**PEDRO TAGLIALENHA (22203674)**
**VITOR PRAXEDES CALEGARI (22200379)**

# RELATÓRIO LABORATÓRIO 4

**FLORIANÓPOLIS**
**2023**

## 1- Introdução

Este relatório descreve a atividade realizada no Laboratório 4, na qual foi desenvolvido um código em assembly para a arquitetura MIPS com o objetivo de calcular a média aritmética (μ) de uma série de números. Além disso, aborda a segunda questão do laboratório, que propôs alterações no procedimento de cálculo da média, visando aprimorar o desempenho em termos de utilização estatística de instruções.

## 2- Códigos desenvolvidos

### 2.1- Questão 1

Código 1: Código que calcula a média de dois vetores

```
.data
    PROMPT_INIT: .asciiz "\nEntre com o tamanho dos vetores A e B: "
    PROMPT_END_A: .asciiz "\nMedia de A: "
    PROMPT_END_B: .asciiz "\nMedia de B: "
    .align 2
    A: .float 0.11 0.34 1.23 5.34 0.76 0.65 0.34 0.12 0.87 0.56
    .align 2
    B: .float 7.89 6.87 9.89 7.12 6.23 8.76 8.21 7.32 7.32 8.22

.text
main:
    la    $s0, A                # Salva endereco de A em $s0
    la    $s1, B                # Salva endereco de B em $s1

    li    $v0, 4                # Escreve no console a string
    la    $a0, PROMPT_INIT    # armazenada em "PROMPT_INIT"
    syscall                     #

    li     $v0, 5               # Lê um valor inteiro do console
    syscall                     # e armazena no registrador $s2
    move    $s2, $v0      #

    move    $a0, $s2      # Passa o tamanho dos vetores para $a0
                                # (parametro de "Calcula_media"


    move    $a1, $s0      # Passa o endereco de A para $a1
                                # (parametro de "Calcula_media")

    jal     Calcula_media         # Chama o procedimento "Calcula_media" para o vetor A

    li    $v0, 4                # Escreve no console a string
    la    $a0, PROMPT_END_A    # armazenada em "PROMPT_END_A"
    syscall                     #

    mtc1     $v1, $f12          # Move o retorno de "Calcula_media" para $f12
    li    $v0, 2                # para podermos imprimir float no terminal

    syscall                     # Realiza a chamada de sistema

    move    $a0, $s2      # Passa o tamanho dos vetores para $a0
                                # (parametro de "Calcula_media"

    move    $a1, $s1      # Passa o endereco de B para $a1
```

```
                                # (parametro de "Calcula_media")

    jal    Calcula_media          # Chama o procedimento "Calcula_media" para o vetor B

    li    $v0, 4                  # Escreve no console a string
    la    $a0, PROMPT_END_B    # armazenada em "PROMPT_END_B"
    syscall                       #

    mtc1    $v1, $f12     # Move o retorno de "Calcula_media" para $f12
    li    $v0, 2                  # para podermos imprimir float no terminal

    syscall                       # Realiza a chamada de sistema

    li    $v0, 10                 # Termina o programa por meio de uma
    syscall                       # chamada de sistema

Calcula_media:

    #=================================
    # Calcula a media de um dado vetor
    #=================================


    # Recebe o tamanho do vetor por meio de $a0
    # Recebe endereco do vetor por meio de $a1
    # Retorna a media em $v1
    #
    # $s0 = tamanho do vetor
    # $s1 = endereco do vetor
    # $t0 = indice atual do calculo
    # $t1 = endereco atual do vetor

    addi    $sp, $sp, -12         # Aumentra pilha e salva $s0, $s1 e $s2
    sw    $s0, 4($sp)     #
    sw    $s1, 8($sp)     #
    sw    $s2, 12($sp)            #

    move     $s0, $a0     # Recebe os argumentos do procedimento e
    move    $s1, $a1      # salva nos devidos registradores

    li    $t0, 0                  # Inicializa $t0 e $f1 com zero
    mtc1    $t0, $f1      #
    cvt.s.w    $f1, $f1           #
loop_media:

    mul    $t1, $t0, 4            # Incrementa o endereco atual em 4 para acessar
    add    $t1, $s1, $t1          # a proxima posicao do vetor

    lwc1    $f0, 0($t1)           # Carrega $f0 com um elemento do vetor na posicao $t1

    add.s    $f1, $f1, $f0        # Soma $f0 ao somatorio

    addi    $t0, $t0, 1           # Incrementa o indice atual do calculo

    bne    $t0, $s0, loop_media    # Checa se o vetor ja acabou, caso tenha acabado continua
                                  # caso nao, pula para "loop_media"

    mtc1    $s0, $f0      # Carrega o registrador $f0 com o tamanho do vetor
    cvt.s.w $f0, $f0

    div.s    $f2, $f1, $f0        # Calcula (somatorio_dos_elementos/tamanho_do_vetor)

    mfc1    $v1, $f2      # Retorna em $v1 o resultado
```

```
    lw    $s0, 4($sp)     # Retorna $s0, $s1 e $s2 aos seus valores originais e
    lw    $s1, 8($sp)     # retorna a pilha para sua posicao anterior
    lw    $s2, 12($sp)            #
    addi    $sp, $sp, 12          #

    jr    $ra
```

Fonte: Elaborado por autores(2023)

Nós desenvolvemos um código em assembly para calcular a média aritmética de duas séries de números, A e B. Vamos explicar as partes importantes deste código:

**Inicialização dos vetores A e B:**

Os vetores A e B são inicializados com valores em ponto flutuante.

**Entrada de Dados:**

A mensagem "Entre com o tamanho dos vetores A e B" é exibida no console.

O tamanho dos vetores é lido do usuário e armazenado em $s2.

**Chamada da Função "Calcula_media" para o Vetor A:**

O tamanho e o endereço do vetor A são passados como parâmetros para a função "Calcula_media".

O resultado da média é armazenado em $v1 e posteriormente impresso no console.

**Chamada da Função "Calcula_media" para o Vetor B:**

O tamanho e o endereço do vetor B são passados como parâmetros para a função "Calcula_media".

O resultado da média é armazenado em $v1 e posteriormente impresso no console.

**Função "Calcula_media":**

Esta função recebe o tamanho e o endereço de um vetor como argumentos.

Ela calcula a média aritmética dos elementos do vetor e retorna o resultado em $v1.

O loop "loop_media" percorre o vetor, somando seus elementos.

O resultado é dividido pelo tamanho do vetor para obter a média.

No final, os registradores $s0, $s1 e $s2 são restaurados, e a função retorna.

O código faz uso de chamadas de função, manipulação de pilha, conversões entre inteiros e pontos flutuantes e operações de loop para calcular as médias dos vetores A e B.

## 2.2- Questão 2

Código 2: Optimização do código 1

```
.data
    PROMPT_END_A: .asciiz "\nMedia de A: "
    PROMPT_END_B: .asciiz "\nMedia de B: "
    .align 2
    A: .float 0.11 0.34 1.23 5.34 0.76 0.65 0.34 0.12 0.87 0.56
    .align 2
    B: .float 7.89 6.87 9.89 7.12 6.23 8.76 8.21 7.32 7.32 8.22

.text
main:
    la    $s0, A                # Salva endereco de A em $s0
    la    $s1, B                # Salva endereco de B em $s1

    li     $v0, 5               # Lê um valor inteiro do console
    syscall                     # e armazena no registrador $s2

    move    $a1, $v0      # Passa o tamanho dos vetores para $a1
                                # (parametro de "Calcula_media")

    move    $a2, $s0      # Passa o endereco de A para $a1
                                # (parametro de "Calcula_media")

    jal     Calcula_media       # Chama o procedimento "Calcula_media" para o vetor A

    li    $v0, 4                # Escreve no console a string
    la    $a0, PROMPT_END_A    # armazenada em "PROMPT_END_A"
    syscall                     #

    li    $v0, 2                # Imprime float no terminal
    syscall                     #

    move    $a2, $s1      # Passa o endereco de B para $a1
                                # (parametro de "Calcula_media")

    jal    Calcula_media        # Chama o procedimento "Calcula_media" para o vetor B

    li    $v0, 4                # Escreve no console a string
    la    $a0, PROMPT_END_B    # armazenada em "PROMPT_END_B"
    syscall                     #

    li    $v0, 2                # Imprime float no terminal
    syscall                     #

    li    $v0, 10               # Termina o programa por meio de uma
    syscall                     # chamada de sistema

Calcula_media:

    #=================================
    # Calcula a media de um dado vetor
    #=================================


    # Recebe o tamanho do vetor por meio de $a1
    # Recebe endereco do vetor por meio de $a2
```

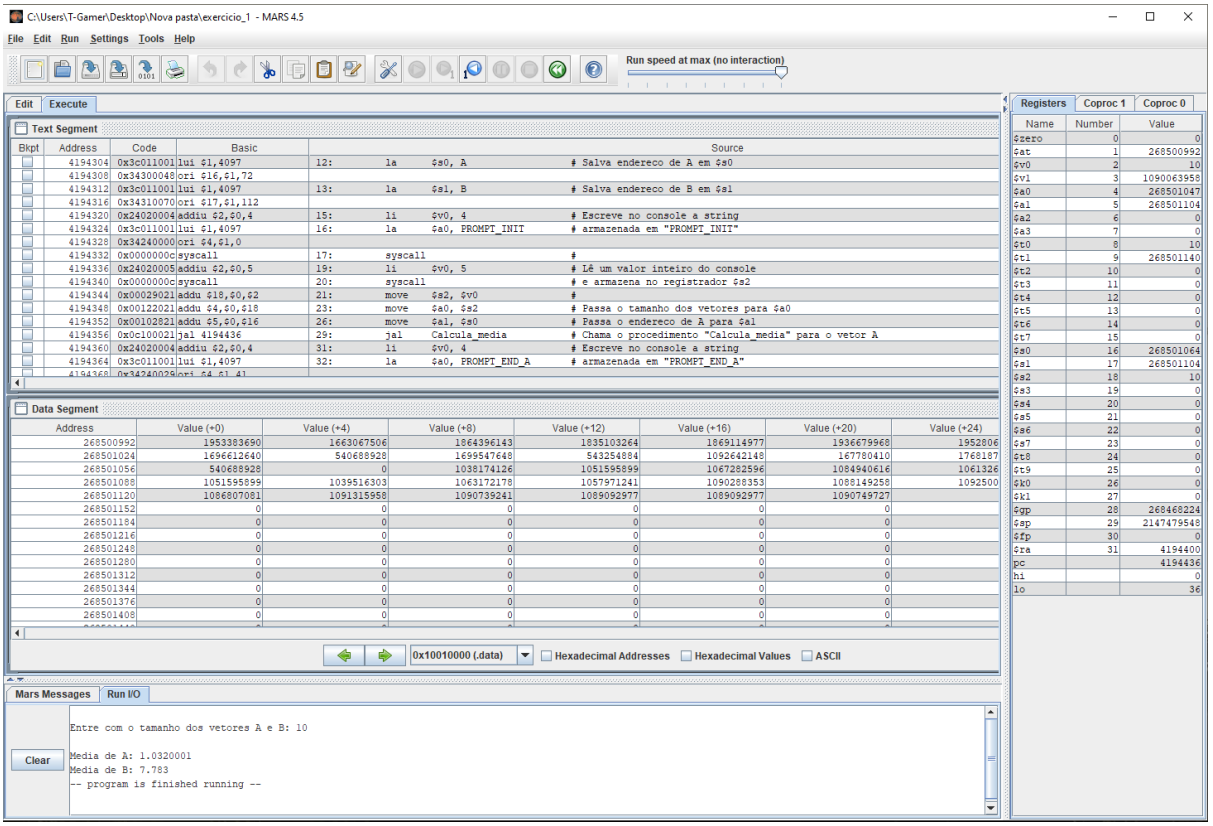
```
    # Retorna a media em $f12
    #
    # $s2 = tamanho do vetor
    # $s3 = endereco do vetor
    # $t0 = indice atual do calculo
    # $t1 = endereco atual do vetor

    move     $s2, $a1     # Recebe os argumentos do procedimento e
    move    $s3, $a2      # salva nos devidos registradores

    li    $t0, 0                  # Inicializa $t0 e $f1 com zero
    mtc1    $t0, $f12     #
    cvt.s.w    $f12, $f12         #
loop_media:

    mul    $t1, $t0, 4            # Incrementa o endereco atual em 4 para acessar
    add    $t1, $s3, $t1          # a proxima posicao do vetor

    lwc1    $f0, 0($t1)           # Carrega $f0 com um elemento do vetor na posicao $t1

    add.s    $f12, $f12, $f0      # Soma $f0 ao somatorio

    addi    $t0, $t0, 1           # Incrementa o indice atual do calculo

    bne    $t0, $s2, loop_media    # Checa se o vetor ja acabou, caso tenha acabado continua
                                   # caso nao, pula para "loop_media"

    mtc1    $s2, $f0      # Carrega o registrador $f0 com o tamanho do vetor
    cvt.s.w $f0, $f0      #

    div.s    $f12, $f12, $f0      # Calcula (somatorio_dos_elementos/tamanho_do_vetor)

    jr    $ra
```

Fonte: Elaborado por autores(2023)

Foram realizadas algumas otimizações significativas no código 2 em relação ao código 1. Primeiramente, a operação de manipulação de pilha no início e no final da função "Calcula_media" foi eliminada, o que reduziu a complexidade e o consumo de recursos do código. Além disso, a mensagem de prompt inicial que não era estritamente necessária foi removida. Para otimizar ainda mais, os argumentos da função foram colocados em registradores que não seriam modificados posteriormente, reduzindo a necessidade de transferências de dados entre registradores. Por fim, o valor de retorno da função passou a ser diretamente armazenado em $f12, eliminando a etapa adicional de mover o resultado para $v1 antes de ser impresso no console.

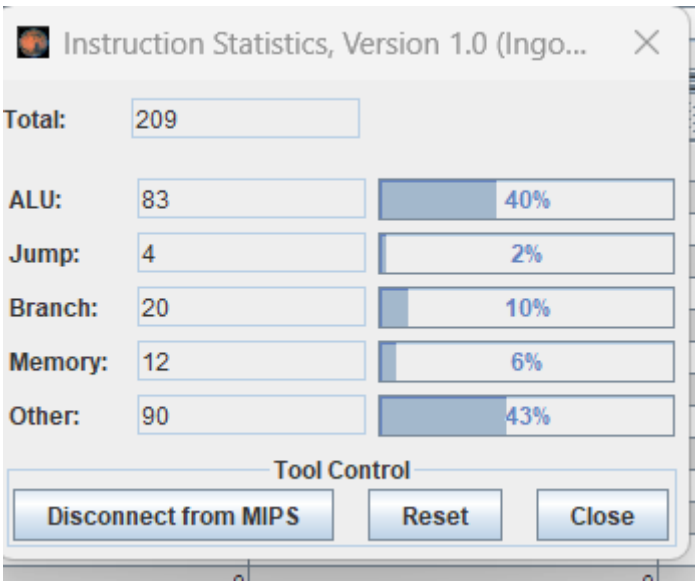
## 3- Saídas das estatísticas de instruções

Figura 1: Saída do código 1



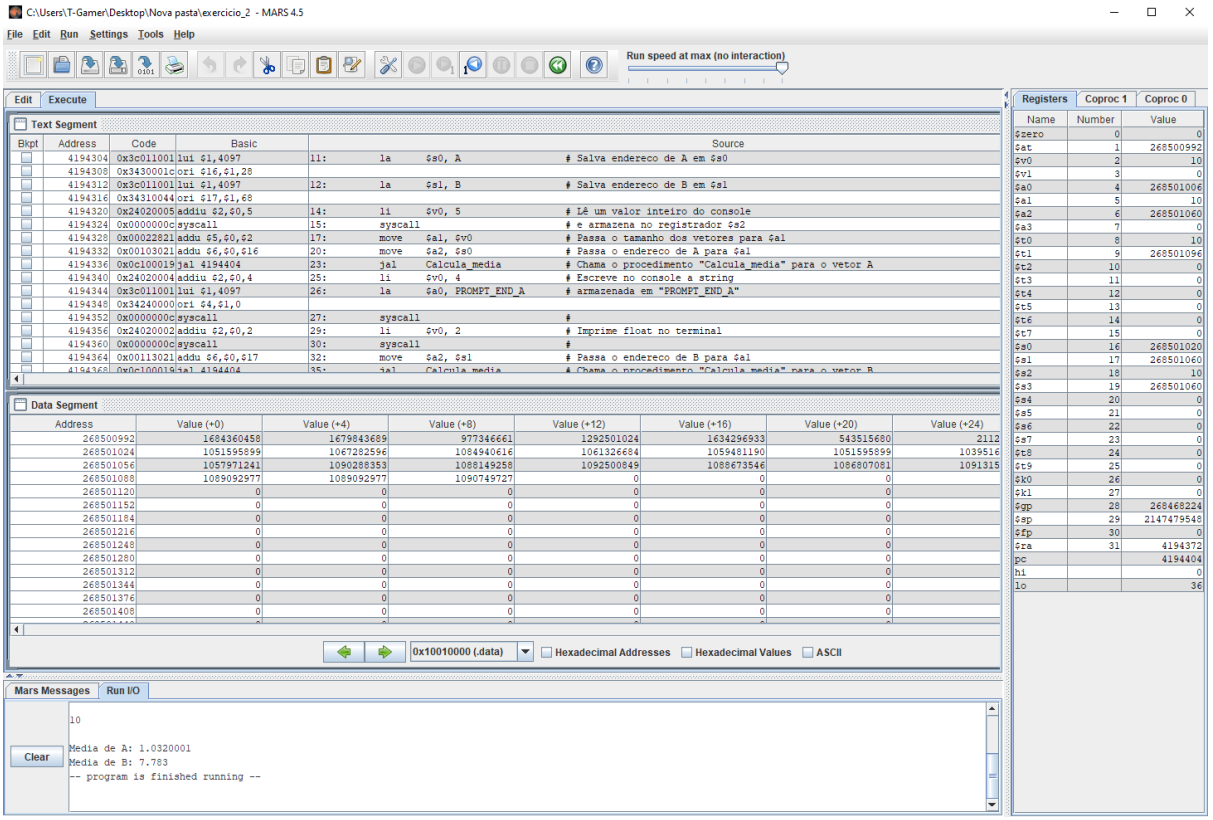

Fonte: Elaborado por autores(2023)

Figura 2: Estatísticas código 1



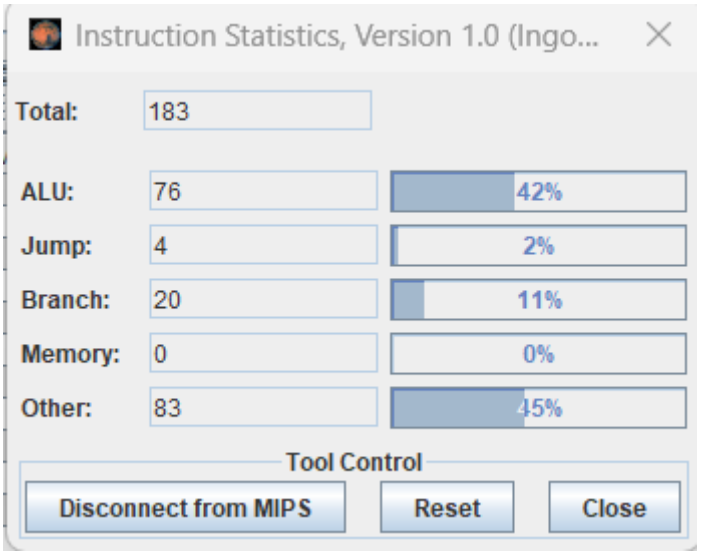

Fonte: Elaborado por autores(2023)

Figura 3: Saída do código 2




Fonte: Elaborado por autores(2023)

Figura 4: Estatísticas código 2




Fonte: Elaborado por autores(2023)

## 4- Conclusões

Em conclusão, as questões 1 e 2 deste projeto foram executadas com sucesso. No decorrer da segunda questão, implementamos otimizações que resultaram em melhorias significativas no código assembly. Com essas otimizações, conseguimos reduzir a quantidade total de instruções de 209 para 183, sendo 12.5% a menos de instruções, demonstrando uma eficaz redução na complexidade do programa. Além disso, conseguimos eliminar completamente as instruções de memória, que são notoriamente mais lentas em comparação com outras operações, de 12 para 0, conseguimos alcançar um desempenho ainda mais eficiente. Essas melhorias contribuíram substancialmente para a eficiência geral do código, tornando-o mais rápido e eficaz na execução de cálculos de média para as séries de dados A e B.