Relatório do Laboratório 06 - Organização de Computadores

Grupo:

- Henrique Mateus Teodoro - 23100472
- Rodrigo Schwartz - 23100471

Exercício 1

O exercício 1 consiste em criar um programa em Assembly MIPS que calcule a média aritmética de uma série, a partir da fórmula:

$$\mu = \frac{1}{N}\sum_{i=0}^{N-1} a_i$$

É necessário ler dois vetores, A e B, com números de ponto flutuante em precisão simples digitados via teclado. Antes disso, o usuário deve informar o tamanho N dos dois vetores. Assim, é necessário calcular a média entre os elementos de cada vetor.

Inicialmente, no segmento .data são declaradas as variáveis que serão utilizadas no decorrer do código para impressão, além de A e B, vetores que irão armazenar os valores digitados pelo usuário (inicialmente eles possuem a mesma posição na memória).

```
.data
        string_input_n: .asciiz "insira o tamanho dos vetores N: "
        string_input_a: .asciiz "digite os valores do vetor A: \n"
        string_input_b: .asciiz "digite os valores do vetor B: \n"
        string_output_a: .asciiz "\nMédia de A: "
        string_output_b: .asciiz "\nMédia de B: "
        .align 2
        A: .float
        .align 2
        B: .float
```

Logo no início do segmento .data a mensagem string_input_n é escrita na tela e é solicitado ao usuário para digitar o valor N, que representa o tamanho dos vetores. Esse inteiro é armazenado em $s0.

```
.text
        li $v0, 4                   # seta a operação de impressão de string
        la $a0, string_input_n      # passa um ponteiro que aponta para string_input_n para $a0
        syscall

        li $v0, 5                   # seta a operação de leitura de inteiro
        syscall
        move $s0, $v0               # salva em $s0
```

Após ler o valor de N, é necessário ler os números de cada vetor. Então, primeiramente a string_input_a é imprimida na tela, para mostrar que o usuário deve digitar os valores de A, e um laço de repetição é inicializado:

```
        li $v0, 4                   # seta a operação de impressão de string
        la $a0, string_input_a      # passa um ponteiro que aponta para string_input_a para $a0
        syscall

        la $s1, A                   # endereço de A
        move $t1, $s1
        li $t0, 0                   # contador do loop
laco_A:
        bge $t0, $s0, fim_laco_A

        li $v0, 6                   # seta a operação de leitura de float
        syscall                     # salva por padrão em $f0

        s.s $f0, 0($t1)
        addi $t1, $t1, 4
        addi $t0, $t0, 1

        j laco_A

fim_laco_A:
```

O endereço de A é carregado para o $s1, e esse valor é passado para o $t1. Assim, inicia-se um laço de repetição que será executado N vezes (conforme o número digitado anteriormente). O contador do laço é $t0 e para o controle é utilizada a instrução bge. Caso $t0 seja maior ou igual que $s0 (N), o laço é finalizado. A cada iteração do loop um float é lido do teclado e escrito na memória, a partir do endereço de A, salvo em $t1. Após gravar, $t1 é incrementado em 4, passando a apontar para o próximo endereço de memória, que será gravado na próxima iteração. No final do loop o contador também é incrementado em 1 e através da instrução jump o laço é repetido novamente. Os valores do vetor B são lidos da mesma maneira que A, exceto por uma mudança:

```
fim_laco_A:

        li $v0, 4                   # seta a operação de impressão de string
        la $a0, string_input_b      # passa um ponteiro que aponta para string_input_b para $a0
        syscall

        la $s2, B                   # endereço de B
        move $t1, $s2               # move pra $t1 o endereço de B
        mul $t2, $s0, 4             # faz o cálculo do endereço do vetor B com base no vetor A
        add $t1, $t1, $t2
        move $s2, $t1               # passa o endereço do vetor B calculado através o tamanho do vetor
        li $t0, 0                   # contador do loop
laco_B:
        bge $t0, $s0, fim_laco_B

        li $v0, 6                   # seta a operação de leitura de float
        syscall                     # salva por padrão em $f0

        s.s $f0, 0($t1)
        addi $t1, $t1, 4
        addi $t0, $t0, 1

        j laco_B

fim_laco_B:
```

No início do código, nota-se que A e B ocupam o mesmo endereço de memória. Então, para diferenciar as posições de cada vetor de maneira dinâmica. O endereço base do vetor B (armazenado em $s2) passa a ser: B = 4*N + B. Assim, a primeira posição do vetor B será imediatamente após a última posição do vetor A. Decidimos fazer dessa maneira a implementação pois assim o tamanho de cada vetor seria dinâmico, além de que sabíamos que outros valores não seriam escritos em memória, evitando sobrescrita de dados.

Com os valores de A e B já lidos, chamamos a função de media para cada um dos dois vetores. A média dos valores do vetor A é salva em $f0 e do vetor B em $f1.

```
fim_laco_B:

        li $t0, 0                   # Seta em $t0 o valor de 0, que será usado como contador
        move $a0, $s1               # Passa o endereço de A como argumento da função
        move $a1, $s0               # Passa o valor de N para a função
        jal media                   # Chama a função media
        mov.s $f0, $f2              # Move o valor retornado da média de A para $f0
        sub.s $f2, $f2, $f2         # Seta o $f2 o valor 0

        li $t0, 0                   # Seta em $t0 o valor de 0, que será usado como contador

        move $a0, $s2               # Passa o endereço de B como argumento da função
        jal media                   # Chama a função media
        mov.s $f1, $f2              # Move o valor retornado da média de B para $f1

        j exit
```

Para utilizar a função que calcula a média é necessário passar para $a0 o endereço base do vetor e para $a1 o tamanho do vetor. O retorno da função é dado em $f2.

```
media:
        # retorno em $f2
        beq $t0, $a1, fim_media

        l.s $f3, 0($a0)             # Carega o elemento ai para $f3
        add.s $f2, $f2, $f3

        addi $a0, $a0, 4
        addi $t0, $t0, 1

        j media

fim_media:
        mtc1 $a1, $f4           # Move o valor inteiro de $a1 para $f4
        cvt.s.w $f4, $f4        # Converte o valor de inteiro para float

        div.s $f2, $f2, $f4

        jr $ra                  # termina a função de media
```

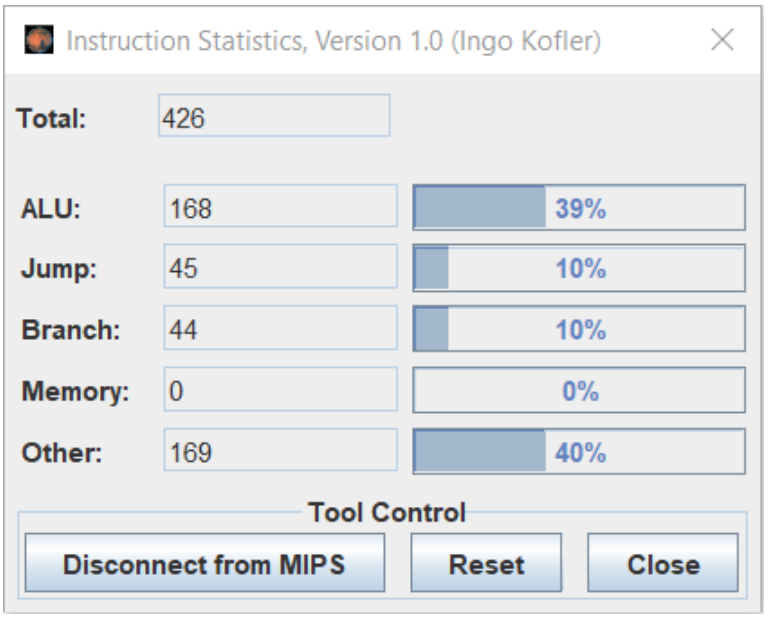
Na função média há um loop que é executado N vezes (tamanho do vetor). A cada iteração é lido um valor do vetor passado como parâmetro, e isso é somado na variável de acumulação $f2. Ou seja, todos os valores do vetor B são somados na variável $f2. Após somar todos os valores, o laço é finalizado. Por fim, basta dividir o somatório ($f2) pelo tamanho do vetor N ($a1) e armazenar o resultado em $f2. Assim, a média é obtida e a função retorna. No final, o resultado das operações é imprimido na tela, através das seguintes instruções:

```
exit:
        li $v0, 4                   # seta a operação de impressão de string
        la $a0, string_output_a     # passa um ponteiro que aponta para string_output_a para $a0
        syscall

        li $v0, 2                   # seta a operação de impressão de float
        mov.s $f12, $f0
        syscall

        li $v0, 4                   # seta a operação de impressão de string
        la $a0, string_output_b     # passa um ponteiro que aponta para string_output_b para $a0
        syscall

        li $v0, 2                   # seta a operação de impressão de float
        mov.s $f12, $f1
        syscall
```

Utilizando o exemplo dado pelo professor, obtivemos resultados positivos, mostrando o bom funcionamento do código:

```
insira o tamanho dos vetores N: 10
digite os valores do vetor A:
0.11
0.34
1.23
5.34
0.76
0.65
0.34
0.12
0.87
0.56
digite os valores do vetor B:
7.89
6.87
9.89
7.12
6.23
8.76
8.21
7.32
7.32
8.22

Média de A: 1.0320001
Média de B: 7.783
-- program is finished running (dropped off bottom) --
```

Ainda, utilizando a ferramenta Instruction Statistics do MARS, obtivemos o seguinte resultado com o exercício 1:



Exercício 2

No exercício 2 é pedido que sejam propostas alterações no cálculo de média do exercício 1, visando mudar as estatísticas de instruções. Com base nisso, visando obter uma drástica melhora no que diz respeito a desempenho, elaborou-se um código mais simples e eficiente.

```
data
        string_input_n: .asciiz "insira o tamanho dos vetores N: "
        string_input_a: .asciiz "digite os valores do vetor A: \n"
        string_input_b: .asciiz "digite os valores do vetor B: \n"
        string_output_a: .asciiz "\nMédia de A: "
        string_output_b: .asciiz "\nMédia de B: "
```

O programa inicia com a declaração das variáveis no segmento .data, cujas variáveis serão utilizadas para exibir mensagens de I/O.

```
.text
        li $v0, 4                   # seta a operação de impressão de string
        la $a0, string_input_n      # passa um ponteiro que aponta para string_input_n para $a0
        syscall

        li $v0, 5                   # seta a operação de leitura de inteiro
        syscall
        move $s0, $v0               # salva em $s0

        li $v0, 4                   # seta a operação de impressão de string
        la $a0, string_input_a      # passa um ponteiro que aponta para string_input_a para $a0
        syscall

        li $t0, 0                   # contador do loop
```

No primeiro bloco de código do segmento .text, são realizadas as impressões das strings de input, para que o usuário insira o tamanho dos vetores, e os respectivos valores do vetor A e B.

```
laco_A:
        bge $t0, $s0, fim_laco_A

        li $v0, 6                   # seta a operação de leitura de float
        syscall                     # salva por padrão em $f0

        add.s $f2, $f2, $f0
        addi $t0, $t0, 1

        j laco_A

fim_laco_A:
        mtc1 $s0, $f1           # Move o valor inteiro de $s0 para $f1
        cvt.s.w $f1, $f1        # Converte o valor de inteiro para float

        div.s $f2, $f2, $f1

        li $v0, 4                   # seta a operação de impressão de string
        la $a0, string_input_b      # passa um ponteiro que aponta para string_input_b para $a0
        syscall

        li $t0, 0                   # contador do loop
```

Na label laco_A, é feita a leitura dos N floats do vetor A, que são passados via console. É neste trecho de código que ocorre uma grande redução de instruções e ganho de eficiência. Ao invés de utilizar dois procedimentos para realizar a soma das entradas dos vetores A e B, assim que as entradas são lidas, já são somadas e computadas diretamente em $f2 e $f3, respectivamente. Com isso, é possível reduzir a quantidade de desvios a serem tomados, e também, várias linhas de código (redução no número de loops). Após o término do loop de leitura das entradas do vetor A, na label fim_laco_A, já é realizada a média, e assim, antes mesmo do vetor B ter seus valores lidos e computados, o vetor A já está com sua média calculada e pronta.

```
laco_B:
        bge $t0, $s0, fim_laco_B

        li $v0, 6                   # seta a operação de leitura de float
        syscall                     # salva por padrão em $f0

        add.s $f3, $f3, $f0
        addi $t1, $t1, 4
        addi $t0, $t0, 1

        j laco_B

fim_laco_B:

        div.s $f3, $f3, $f1
```

Realiza-se então o mesmo bloco de código visto anteriormente para o vetor A, e assim, ao término da primeira instrução da label fim_laco_B, já se tem a média das entradas do vetor B calculada e pronta em $f3.

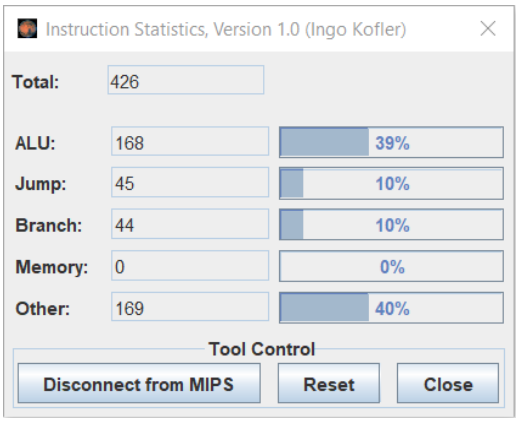
```
        li $v0, 4                   # seta a operação de impressão de string
        la $a0, string_output_a     # passa um ponteiro que aponta para string_output_a para $a0
        syscall

        li $v0, 2                   # seta a operação de impressão de float
        mov.s $f12, $f2
        syscall

        li $v0, 4                   # seta a operação de impressão de string
        la $a0, string_output_b     # passa um ponteiro que aponta para string_output_b para $a0
        syscall

        li $v0, 2                   # seta a operação de impressão de float
        mov.s $f12, $f3
        syscall
```

Após o cálculo das médias, são realizadas as impressões desses resultados na tela do console.

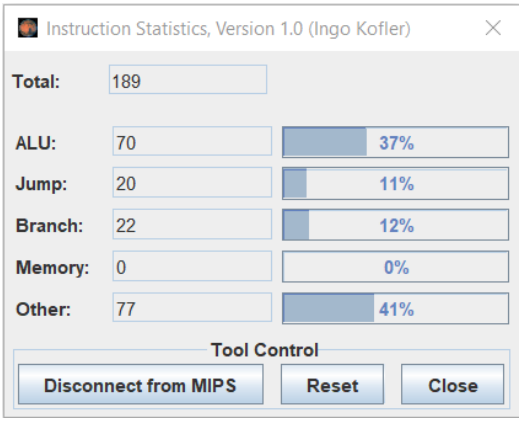
Realizando o teste deixado pelo professor, obtivemos o resultado esperado, assim como no primeiro exercício:

```
insira o tamanho dos vetores N: 10
digite os valores do vetor A:
0.11
0.34
1.23
5.34
0.76
0.65
0.34
0.12
0.87
0.56
digite os valores do vetor B:
7.89
6.87
9.89
7.12
6.23
8.76
8.21
7.32
7.32
8.22

Média de A: 1.0320001
Média de B: 7.783
-- program is finished running (dropped off bottom) --
```

Vamos então comparar o desempenho e gasto de instruções dos dois programas, visando saber se houve uma melhora no que diz respeito às estatísticas do segundo programa em relação ao primeiro. Para se obter uma análise com base nesses parâmetros, utilizou-se a ferramenta Instruction Statistics do MIPS, que provê o número de instruções lógicas e aritméticas (ALU), o número de jumps, de branchs e outras informações a respeito do programa.



Estatísticas do primeiro programa



Estatísticas do segundo programa

Como é possível perceber, o segundo programa utilizou menos da metade das instruções do primeiro. Além de realizar também menos da metade dos desvios condicionais e incondicionais. Com base nessas informações, é possível afirmar que não só o segundo programa é menos custoso que o primeiro, devido a grande diferença de instruções, mas também que o segundo programa seria melhor visando-se obter um maior ganho de desempenho através do pipeline. Isso porque a redução no número de desvios condicionais possibilita com que o pipeline seja executado “cheio” mais vezes. Em outras palavras, serão realizadas menos esperas, devido a menor quantidade de desvios condicionais, e assim, o fluxo do pipeline seguirá mais vezes sem interrupções. Assim, conclui-se que o segundo programa não só diminui o custo computacional, como melhora o desempenho através de uma execução mais eficiente do pipeline na CPU.