Atividade Prática VI
Organização de Computadores I

Gabriel Salmoria e Gustavo Nunes Viana

- Atividade 1:
  Calcular a média de dois vetores utilizando da fórmula:

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

  No qual o tamanho n e os valores a_i são dados pelo usuário em tempo de execução.

  – Código:

  Vamos, inicialmente, descrever o código que iremos usar. Este será definido em três etapas:

  1. Receber a variável n e alocar o espaço necessário para os vetores A e B na memória.
  2. Utilizar um **for loop** para receber os n valores para cada vetor.
  3. Calcular o valor da média utilizando-se de um procedimento Media.

  Primeiramente, vamos receber o valor inteiro (utilizando-se da função syscall - 5) que irá determinar n e calcular qual deverá ser o espaço que será utilizado para cada vetor na memória.

```
li $v0, 5
syscall
move $s0, $v0

move $a0, $s0
li $t0, 4
mul $a0, $a0, $t0
```

  Além disso, vamos separar o espaço de $4.n$ bytes na memória para ambos os vetores. Para isso, vamos utilizar a função syscall - 9, que irá pedir ao sistema operacional um determinado espaço na *heap*.

```
li $v0, 9
move $a0, $a0
syscall
move $t1, $v0

== Mesmo Para B ==

li $s3, 0
```

  Depois disso, estamos prontos para realizar o nosso loop. Dentro dele, vamos:

1. Checar se devemos sair do loop.
2. Receber um valor para a_i e armazena na posição atual da memória.
3. Faz o mesmo para o valor b_i.

```
Loop1:
    beq $s3, $s0, OutLoop1

    li $v0, 6
    syscall
    sll $t3, $s3, 2
    add $t4, $t1, $t3
    s.s $f0, 0($t4)

    == Mesmo para B ==

    addi $s3, $s3, 1
    j Loop1
```

Depois disso, estamos preparados para chamar a função Media, além de definir os parâmetros iniciais da função, como receber os valores de argumento e passá-los para registradores do tipo $t.

```
OutLoop1:
    move $a0, $s0
    move $a1, $t1
    move $a2, $t2
    jal Media
    j Exit

Media:
    li $s0, 0
    move $s1, $a0
    move $t1, $a1
    move $t2, $a2
    li $t5, 0
    mtc1 $t5, $f1
    mtc1 $t5, $f2
```

Depois disso, estamos prontos para realizar o nosso somatório. Dentro dele, vamos:

1. Checar se devemos sair do loop.
2. Carregar o valor a_i na memória e somá-lo ao total.
3. Fazer o mesmo para o valor b_i.

```
Loop2:
    beq $s0, $s1, OutLoop2
    sll $t3, $s0, 2

    add $t4, $t1, $t3
    l.s $f0, 0($t4)
    add.s $f1, $f1, $f0

    == Faz o mesmo para B ==

    addi $s0, $s0, 1
    j Loop2
```

Por fim, vamos, finalmente, realizar a divisão por n, obter o valor de Media(B) e Media(B) e mostrar ambos no terminal.

```
OutLoop2:
    mtc1 $a0, $f3
    cvt.s.w $f3, $f3

    div.s $f5, $f1, $f3
    div.s $f6, $f2, $f3

    li $v0, 4
    la $a0, newline
    syscall

    li $v0, 2
    mov.s $f12, $f5
    syscall

    == Mesmo para B ==
```

– Atividade 2:

Vamos alterar o nosso código para que ele faça, em essência, a mesma coisa, mas visando diminuir o número de instruções realizadas pelo código. Para isso, vamos fazer com que, ao invés de:

* Receber o valor a_i
* Guardar o valor a_i na memória
* (Mais tarde) Retirar o valor da memória.
* Acumular o valor no total.

Vamos apenas receber o valor e logo acumulá-lo ao montante. Dessa forma, iremos reduzir o número de instruções desnecessárias para o contexto do nosso código e ainda iremos fazer a mesma utilidade. Vamos passar, então, pelos pontos principais do código alterado:

* Não temos mais a necessidade de alocar dois espaços de tamanho $4.n$ na memória. Mudando, assim, o início do nosso código.
* Não temos mais a necessidade de guardar os valores na memória e resgatá-los depois.

Dessa forma, nossa main será da forma:

```
li $v0, 5
syscall
move $s0, $v0

li $t5, 0
mtc1 $t5, $f1
mtc1 $t5, $f2
li $s3, 0
```

Além disso, nosso loop para obter os valores de a_i e b_i será da forma:

```
beq $s3, $s0, OutLoop

li $v0, 6
syscall
```

```
add.s $f1, $f1, $f0

== Faz o mesmo para B ==

addi $s3, $s3, 1
j Loop
```

– Análise do desempenho para o Primeiro caso com variações no tamanho do vetor em 3, 5 e 10:

Instrução	**Tam 3**	**Tam 5**	**Tam 10**
Total	138	198	348
ALU	57	85	155
Jump	9	13	23
Branch	8	12	22
Memory	0	0	0
Other	64	88	148

Estatísticas das Instruções

– Análise do desempenho do código após as nossas alterações, com variações no tamanho do vetor em 3, 5 e 10:

Instrução	**Tam 3**	**Tam 5**	**Tam 10**
Total	59	77	122
ALU	21	27	42
Jump	6	8	13
Branch	4	6	11
Memory	0	0	0
Other	28	36	56

Estatísticas das Instruções

Podemos observar que a diminuição no número de instruções foi considerável. Para todos os casos, tivemos 2.5 vezes menos instruções. Essa diminuição deve-se principalmente ao discutido anteriormente, nossa redução da quantidade de operações Aritméticas deve-se, principalmente, à não termos mais a necessidade de ficar calculando offsets para guardar e, depois, resgatar cada valor do nosso vetor. Além disso tivemos uma redução drástica na quantidade de instruções do tipo "Other", cuja, dentre outras, pode ser avaliada devido à diminuição de chamadas de sistema (syscalls), além do menor número de instruções como Load Word e Store Word.