RELATÓRIO LABORATÓRIO 3

ORGANIZAÇÃO DE COMPUTADORES I

Emanuelle Guse - 23100486

EXERCÍCIO 1

1. Inicialmente, definimos valores em .asciiz para mostrar as mensagens no output. O programa inicia pedindo para o usuário digitar os valores desejados para A e B, que são salvos em registradores temporários. Mesmo sendo necessário passá-los para registradores do tipo $a mais tarde, ainda não é possível fazer isso já que para as interações i/o é necessário passar argumentos também.

```
.data
    askInputA: .asciiz "insira A: "
    askInputB: .asciiz "insira B: "
    showResult: .asciiz "o produto é: "
    breakLine: .asciiz "\n"
```

$t0	8	5
$t1	9	3

2. *Product*: após isso, transferimos os valores de $ti para $ai e chamamos o procedimento *product*, passando os valores recebidos como parâmetro.

2.1) Começamos ajustando o ponteiro subtraindo 4 (uma posição a menos) e o endereço de retorno é salvo nessa posição.

$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	4194372
pc		4194412

antes

$gp	28	268468224
$sp	29	2147479544
$fp	30	0
$ra	31	4194372
pc		4194416

depois

2.2) É verificado e o $a1 (b) é zero. Se for, ele passa para o caso base (*3). Caso contrário, b é decrementado e o procedimento é chamado novamente. A cada chamada o ponteiro é decrementado e vai guardando novos valores. Finalmente, ***b*** chega a zero e passa para o caso base.

$v0	2	3
$v1	3	0
$a0	4	5
$a1	5	2

$v0	2	3
$v1	3	0
$a0	4	5
$a1	5	0

2.3) No caso base, ele inicia pegando o endereço de retorno e ajusta o ponteiro da pilha, dessa vez incrementando. Ele volta para a posição logo abaixo da chamada de *product* dentro de *product*. O valor da chamada é somado com $v0. Assim, ele vai fazendo isso incrementando a cada vez os ponteiros de retorno e recebendo os retornos.

$v0	2	5

$v0	2	10

$v0	2	15

2.4) Finalmente, o valor de retorno é o endereço da chamada na main. Lá, iniciamos movendo o valor de retorno ($v0) para um registrador temporário, já que não queremos perder o valor ao fazer outras chamadas. Fazemos o processo de fazer um output e mostramos o resultado final na tela. Após isso, é feito um jump para o end, chamando o fim do processo.

```
insira A: 5
insira B: 3
o produto é: 15
```

EXERCÍCIO 2

1. Para o exercício 2, inicialmente foram salvos na memória o valor do vetor, número de elementos e o texto para mostrar com o resultado.

```
vector: .word 13, 2, 33, 14, 142
n: .word 5
result: .asciiz "a soma é: "
```

Na main, o endereço do vetor e o valor do número de elementos são armazenados em $a1 e $a0, respectivamente, servindo como argumento para a função. A função *sum* é, então, chamada.

2. Primeiramente, são alocados dois espaços na sp (dois "blocos"). Esses espaços são carregados com o endereço de retorno e endereço do vetor. É verificado se o valor do argumento 0 (n) é zero. Se for, ele passa para o caso base.

Value (+24)
4194324
Value (+20)
268500992

Caso contrário, o valor do vetor no momento atual é passado para um registrador temporário e o endereço do vetor é incrementado, para receber os próximos valores. Além disso, a contagem é decrementada.

É chamada novamente sum e o processo é repetido, lembrando que o endereço do vetor agora está um byte para a frente.

Ao cair no caso base, é selecionado o endereço de retorno e ajustado o ponteiro da pilha. O valor de retorno de cada endereço é somado a um registrador temporário e o ponteiro da pilha é ajustado.

Finalmente voltando à main, o valor retornado em $v0 é armazenado em um registrador temporário e o valor da soma é mostrado na tela. O processo é encerrado.

```
a soma é: 204
-- program is finished running --
```