Atividade Prática I
Organização de Computadores I

Gabriel Salmoria e Gustavo Nunes Viana

- Atividade 1:

  Realizar a multiplicação de dois inteiros A e B de forma recursiva.

  Resolução:

  A nossa resolução se resumirá em criar um procedimento e chamá-lo B vezes, cada uma dessas vezes, somando o valor A ao total. Dessa forma, vamos inicialmente definir as variáveis A e B na seção ".data":

```
.data
        a: .word 4
        b: .word 5
```

  Depois disso, vamos adicionar essas variáveis à registradores do tipo A, para enviá-los ao procedimento que irá realizar a multiplicação. Além disso, vamos realizar a instrução de "Jump And Link" para iniciar o procedimento (já incluimos também a label de Exit, que irá finalizar o código ao terminarmos de executar o procedimento):

```
.text
        lw $a0, a
        lw $a1, b
        jal Multiplicacao
        j Exit
```

  Agora, falta apenas implementar o procedimento em si. Para isso, vamos carregar os valores de A e B nos registradores $a1 e $ra. (Perceba que, neste momento, estamos adicionando os valores à stack e mudando o valor do stack pointer):

```
        addi $sp, $sp -8
        sw $a1, 4($sp)
        sw $ra, 0($sp)
```

  Por fim, vamos implementar a lógica de recursão do código. Para isso, vamos criar um condicional If para checar se o valor de B chegou à zero, caso não tenha, vamos decrementar B e vamos chamar novamente a função e depois de todas as funções serem retornadas, faz a soma da variável.

```
    If:
        addi $a1, $a1, -1
        jal Multiplicacao
        add $v0, $a0, $v0
```

Caso B tenha chegado à zero, a função irá retornar zero e salvar os valores das variáveis A e B. (Perceba que, neste momento, estamos retirando os valores da stack e liberando o espaço alocado pelas variáveis):

```
    EndIf:
        lw $ra, 0($sp)
        lw $a1, 4($sp)
        addi $sp, $sp, 8
        jr $ra
```

- Atividade 2:

  Escrever uma função recursiva que irá somar os valores de um array.

  Resolução:

  Assim como na questão anterior, vamos primeiramente definir os valores na nossa seção ".data". Nesse caso, precisamos definir qual será o nosso array, além do tamanho de tal array. Portanto:

```
    .data
        v: .word 11 2 3 14 15
        n: .word 5
```

  De maneira similar à antes, vamos criar um procedimento que irá usar recursão para realizar a soma. Portanto, vamos iniciar o código adicionando a nossa variável N no registrador tipo A, além de dar início ao procedimento:

```
.text
        lw $a0, n
        jal Soma
        j Exit
```

  Depois disso, vamos salvar as nossas variáveis nos registradores de argumento e de retorno (Da mesma forma que na questão anterior, estamos agora atualizando a stack):

```
Soma:
        addi $sp, $sp -8
        sw $a0, 4($sp)
        sw $ra, 0($sp)
```

  Da mesma maneira, vamos criar um condicional If para checar se o número de elementos chegou à 1:

```
        li $t0, 1
        bne $a0, $t0, If

        lw $v0, v($zero)
        j EndIf
```

Caso contrário, vamos chamar novamente o procedimento e, depois que todos os procedimentos retornarem seus valores, vamos realizar a soma:

```
        If:
        addi $a0, $a0, -1
        jal Soma
        sll $a0, $a0, 2
        lw $t0, v($a0)
        add $v0, $t0, $v0
```

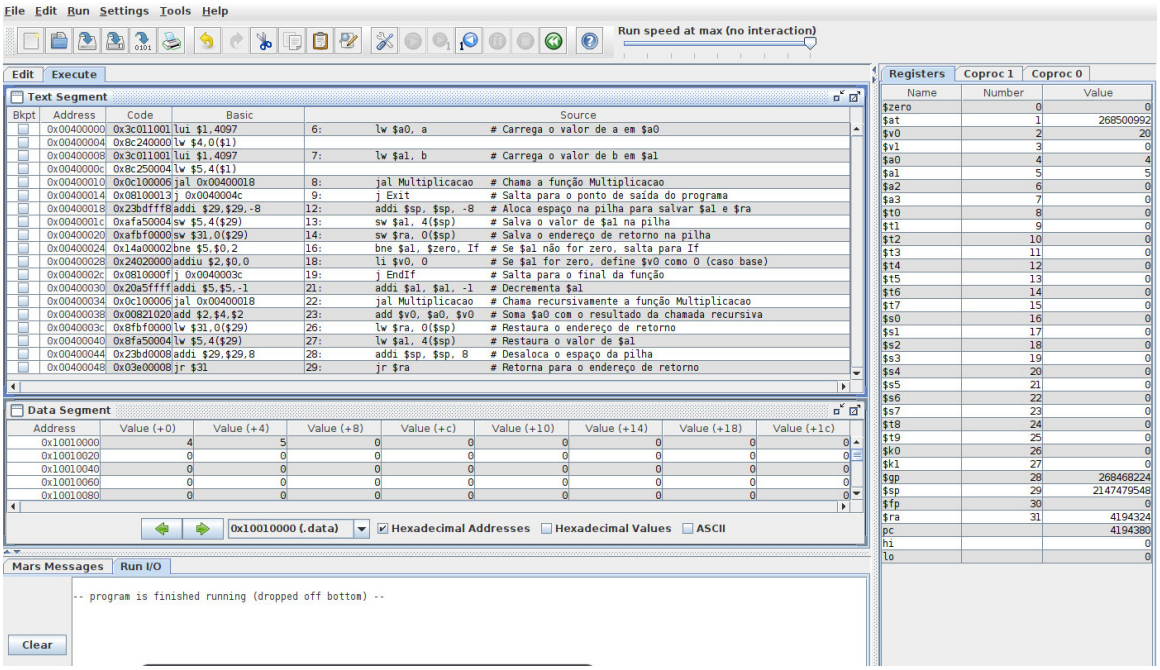
Por fim, se o valor for 1 (ou seja, estamos no ultimo valor do array), vamos retornar o valor para o procedimento anterior e fechar este (Novamente, perceba que estamos liberando os valores da stack):

```
EndIf:
        lw $ra, 0($sp)
        lw $a0, 4($sp)
        addi $sp, $sp, 8
        jr $ra
```

No caso acima, os procedimentos que não possuem valor 1 abrem outros procedimentos até que o valor final seja realizado. Quando este acontece, ele retorna o ultimo valor do array e libera o próprio espaço da stack. Depois disso, vamos somando o valor de $v0 ao valor atual do array e liberando o espaço do procedimento atual da stack. Assim, realizamos uma soma dos elementos do array de forma recursiva.

No nosso código da Atividade 1 há 18 linhas de instrução, sem contar pseudo-instruções

Ao fim de uma execução completa, esse é o resultado do programa:



No nosso código da Atividade 2 há 24 linhas de instrução, sem contar pseudo-instruções

Ao fim de uma execução completa, esse é o resultado do programa: