Exemplo 4 - Chamada de procedimentos e armazenamento de dados na pilha

Objetivo: Implementar um procedimento para o cálculo do fatorial de n. Tal procedimento deverá ser feito de tal maneira a chamar um segundo procedimento.
$v0 = retorna o resultado ao programa principal
$v0 = 0 se n < 1
$v0 = 1 se n = 1
$v0 = n! se n > 1
n! = n.(n-1).(n-2)... .1

```
.text
.globl main
main:                                   # Programa Principal
li $a0, 5                               # Parametro = 5
jal Fatorial                            # Chama a função fatorial,
                                        # armazenando o endereco da proxima instrucao em
                                        $ra
move $s0, $v0                           # Registrador $s0 recebe o valor resultado
li $v0, 10                              # Serviço do sistema no. 10 : Exit
syscall                                 # Chamada de serviço do sistema
                                        # Obs. Isto é necessário para encerrar aqui o
                                        programa

Fatorial:                               # Função Fatorial. Retorna o fatorial de n ($a0)
sub $sp,$sp,4                           # Abre espaço para armazenar 1 item na pilha
sw $ra, 0($sp)                          # Armazena o conteúdo do registrador $ra
                                        # Obs: Caso os registradores $s0 - $s9 fossem
                                        alterados, seria necessário guardá-los na pilha
                                        também, e restaurá-los ao fim do procedimento. Por
                                        convenção, os registradores temporários $t* não
                                        precisam ser guardados.
li $t1, 1                               # $t1 = 1
slti $t0, $a0, 2                        # Seta $t0 se $a0 < 2
beq $t0, $zero, Calcula                 # Se $t0 não setado, $a0 > 1, portanto Calcula
add $v0, $zero, $zero                   # Se não, $v0 = 0
beq $a0, $zero, Sai                     # Se $a0 = 0, Sai
add $v0, $t1, $zero                     # Se não, $v0 = 1

Sai:
lw $ra, 0($sp)                          # Carrega o registrador $ra da pilha
add $sp, $sp, 4                         # Elimina 1 item da pilha
jr $ra                                  # Retorna ao programa principal

Calcula:
add $a1, $a0, $zero                     # $a1 = $a0
Loop:
```

```
sub $a1, $a1, $t1                   # $a1 = $a1 - 1
jal Multiplica                      # Multiplica $a0 por $a1
add $a0, $v0, $zero                 # $a0 = resultado da multiplicação
bne $a1, $t1, Loop                  # Enquanto $a1 for diferente de 1, Loop
j Sai


Multiplica:
mult $a0, $a1                       # Multiplica $a0 por $a1
mflo $v0                            # resultado em $v0
                                    # Obs:. O resultado da instrução mult fica
                                    armazenado nos registradores # $hi e $lo. Para
                                    buscar os seus valores, utiliza-se mflo e mfhi.
                                    Para # podermos trabalhar com valores maiores em
                                    nosso programa bastaria retornar o $lo em $v0 e o
                                    $hi em $v1.
jr $ra                              # Retorna
```

Salve o arquivo com o nome de "*exercicio4.s*" e abra-o no SPIM. Antes de executá-lo, abra as janelas de Text Segment e a de Data Segment e redimensione-as de modo que as duas fiquem visíveis na tela. Vamos executar o programa passo a passo, através do Single Step (F10). As primeiras 8 linhas de código no Text Segment foram geradas automaticamente pelo SPIM para controlar o fluxo do programa. Continue executando linha por linha até chegar ao nosso código, que começa com `[ 5: li $a0, 5 ]`. A próxima linha muda o fluxo do programa, chamando a função fatorial. Continue executando. Observe que dentro da função fatorial armazenamos o conteúdo de $ra na pilha, e isto pode ser confirmado na janela do segmento de dados. Agora abra a janela de Registradores e continue fazendo o passo a passo, acompanhando a evolução do resultado do fatorial em $a0. Ao final do laço, retornamos ao programa principal que coloca o resultado em $s0.

Outra maneira interessante de se resolver este exercício é através de procedimentos recursivos, ou seja, uma função fatorial que chama a si mesma com um parâmetro (que seria o resultado intermediário) até chegar a um resultado final.