```
#====================================================================
# Static data allocation and initialization
#====================================================================
.data
#
#   String for output
#
output: .asciiz " factorial of five is"
newline: .asciiz "\n"

#====================================================================
# Program text
#====================================================================
.text
.globl main

main:
#
# Main program entry
#
        sw      $ra,0($sp)     #save program address
        subu    $sp,$sp,4      #main is a procedure for the O.S.
#
# Main body
#
        li      $a0, 5          #will compute factorial of 5
        jal     fact
        move    $s0, $v0        #save result
        la      $a0, output     #prepare to print
        li      $v0,4           #header
        syscall
        la      $a0, newline    #prepare to print
        li      $v0,4           #a new line
        syscall
        move    $a0, $s0        #prepare to print
        li      $v0,1           #print factorial
        syscall
#
# main exit
#
        addu    $sp,$sp,4       #restore program address
        lw      $ra,0($sp)
        j       $ra             #return
```

```
#=====================================================================
# PROCEDURE: Fact
# PARAMETERS: an integer in a0
# LOCALS:
# RETURN: fact(a0) in v0
# DESCRIPTION: Computes recursively factorial

fact:
#
# Procedure entry
#
        sw      $ra,0($sp)      #save return address
        subu    $sp,$sp,4       #move sp to new position in frame
#
# Procedure body
#
# test argument for 0; if so return with value 1
        beq     $a0,$0,nore     #test a0; if zero go to out
# otherwise save a0 on the stack
        sw      $a0,0($sp)      #push argument on stack
        subu    $sp,$sp,4
        addi    $a0,$a0,-1      #decrement argument
#
#recursive entry
#
        jal     fact         #call
#
#back from recursive level v0 =v 0 * ao
#
        addu    $sp,$sp,4
        lw      $t0,0($sp)      #pop argument from stack
        mul     $v0,$v0,$t0     #get result at this level
        b       out             #exit that level
#
#end of recursion
#
nore:   li      $v0,1           #result in v0 and exit
#
# Procedure exit
#
out:    addu    $sp,$sp,4
        lw      $ra,0($sp)      #restore return address
        j       $ra             #return
```