```
    1: # factorial.spim --- A recursive SPIM program.  Demonstrates function
    2: #    call and return.
    3:
    4:
    5:              .data
    6: prompt:      .asciiz "Number: "
    7: nl:          .asciiz "\n"
    8: instr:       .asciiz "Enter 0 to exit.\n"
    9: response:    .asciiz "Factorial: "
   10:
   11:
   12: ############################################################################
   13: # main
   14: ############################################################################
   15:
   16:              .text
   17:              .globl main
   18: main:
   19:              sub $sp, $sp, 16     # Push frame & save registers.
   20:              sw $fp, 16($sp)
   21:              sw $ra, 12($sp)
   22:              sw $s0,  8($sp)
   23:              add $fp, $sp, 16
   24:
   25:              li $v0, 4            # Print instruction.
   26:              la $a0, instr
   27:              syscall
   28:
   29:              jal getnum           # Get a number from keyboard.
   30:
   31:              sw $v0, -12($fp)     # Store locally.
   32:              move $s0, $v0
   33:
   34: while1:
   35:              beqz $s0, endwhile1  # Compute until 0 entered.
   36:
   37:              li $v0, 4            # Print response prompt.
   38:              la $a0, response
   39:              syscall
   40:
   41:              move $a0, $s0        # Pass argument
   42:
   43:              jal factorial        # Call
   44:
   45:              move $a0, $v0        # Copy return value to print it
   46:              li $v0, 1
   47:              syscall
   48:
   49:              li $v0, 4            # Prepare to read next number.
   50:              la $a0, nl
   51:              syscall
   52:
   53:              jal getnum
   54:
   55:              sw $v0, -12($fp)     # Store number just read.
   56:              move $s0, $v0
   57:
   58:              b while1
   59:
   60: endwhile1:
   61:              lw $s0,  8($sp)      # Restore registers and pop frame.
   62:              lw $ra, 12($sp)
   63:              lw $fp, 16($sp)
```

```
   64:             add $sp, $sp, 16
   65:             li $v0, 0
   66:             jr $ra              # Exit.
   67:
   68:
   69: ###########################################################################
   70: # getnum --- returns integer read from keyboard through $v0.
   71: ###########################################################################
   72:
   73:             .text
   74: getnum:
   75:             sub $sp, $sp, 12    # push frame & save registers.
   76:             sw $fp, 12($sp)
   77:             sw $ra,  8($sp)
   78:             add $fp, $sp, 12
   79:
   80:             li $v0, 4
   81:             la $a0, prompt
   82:             syscall
   83:
   84:             li $v0, 5
   85:             syscall             # Value read left in $v0.
   86:
   87:             sw $v0, -8($fp)
   88:
   89:             lw $ra,  8($sp)     # restore registers & pop frame.
   90:             lw $fp, 12($sp)
   91:             add $sp, $sp, 12
   92:             jr $ra
   93:
   94:
   95: ###########################################################################
   96: # factorial --- compute factorial of $a0 and return result through
   97: #    $v0
   98: ###########################################################################
   99:
  100:             .text
  101: factorial:
  102:             sub $sp, $sp, 16
  103:             sw $fp, 16($sp)
  104:             sw $ra, 12($sp)
  105:             sw $s0,  8($sp)
  106:             add $fp, $sp, 16
  107:
  108:             sw $a0, -12($fp)       # Our value n.
  109:             move $s0, $a0
  110:
  111:             bgt $s0, 1, recurse  # Base case.  Return 1
  112:             li $v0, 1
  113:             lw $s0, 8($sp)
  114:             lw $ra, 12($sp)
  115:             lw $fp, 16($sp)
  116:             add $sp, $sp, 16
  117:             jr $ra
  118:
  119: recurse:    sub $a0, $a0, 1      # Recursive call.  Compute (n-1)!
  120:             jal factorial
  121:             mul $v0, $v0, $s0    # n * (n-1)!
  122:             lw $s0, 8($sp)
  123:             lw $ra, 12($sp)
  124:             lw $fp, 16($sp)
  125:             add $sp, $sp, 16
  126:             jr $ra
```