**fact.s: A Recursive Subprogram**

# This program computes and displays the factorial of n, where n is entered

# by the user. The program prompts for new values of n until the user

# enters a negative number.

#

# Note: results are valid only for small values of n.

#

.data

# main program string constants

#

prompt: .asciiz "Please enter a positive integer: "

result: .asciiz "! = "

newline: .asciiz "\n"

done: .asciiz "Program terminated."

# main program variables

#

n: .word 0

nFact: .word 0

.text

# main program

#

main:

li $v0, 4 # issue prompt

la $a0, prompt

syscall

li $v0, 5 # get n from user

syscall

sw $v0, n

bltz $v0, end # terminate if n < 0

lw $a0, n # nFact = factorial(n)

jal factorial

sw $v0, nFact

li $v0, 1 # print n

lw $a0, n

syscall

li $v0, 4 # print "! = "

la $a0, result

syscall

li $v0, 1 # print nFact

lw $a0, nFact

syscall

li $v0, 4 # print two newlines

la $a0, newline

syscall

li $v0, 4

la $a0, newline

syscall

b main # repeat with new n

end:

li $v0, 4 # print a termination message

la $a0, done

syscall

li $v0, 4 # print a newline

la $a0, newline

syscall

li $v0, 10 # terminate the program

syscall

# function factorial

#

# C synopsis:

#

# int factorial(int n)

#

# Typical C call:

#

# nFact = factorial(n);

#

# Effect:

#

# Puts n! into nFact.

#

# MIPS call sequence:

#

# lw $a0, n

# jal factorial

# sw $v0, nFact

#

# Stack usage:

#

# return address ($ra) saved at 0($sp)

# n ($a0) saved at 4($sp)

#

factorial:

bgtz $a0, recur # recur if n > 0

li $v0, 1 # else just return 1

jr $ra

recur:

sub $sp, $sp, 8 # allocate stack frame

sw $ra, 0($sp) # with return address at 0($sp)

sw $a0, 4($sp) # and n at 4($sp)

sub $a0, $a0, 1 # $v0 = factorial(n - 1)

jal factorial

lw $ra, 0($sp) # restore return address

lw $a0, 4($sp) # and n from the stack frame

add $sp, $sp, 8 # and deallocate it

mul $v0, $v0, $a0 # return factorial(n - 1)\*n

jr $ra