```
 1 # Recursive factorial program.
 2 # Version with as few registers as possible
 3 # Written by NR Wagner, 23 Sep 1998
 4 #
 5 #    int fact(n)
 6 #    {  if (n > 1) goto recur;
 7 #       return n;
 8 #   recur: return n * fact(n-1); }
 9 #
10 #                   STACK
11 #                  |              |                    |  Old
12 #                  |--------------|                    |  Activation
13 #   old $sp --->   | Return addr  |                    |  Record
14 #              ===|==============|===          -------+
15 #   old -4   --->  | Return value |  <--- new +8       |
16 #                  |--------------|                    |  New
17 #   old -8   --->  | Parameter n  |  <--- new +4       |  Activation
18 #                  |--------------|                    |  Record
19 #   old -12  --->  | Return addr  |  <--- new $sp      |
20 #              ===|==============|===          -------+
21 #                  | Return value |  <--- new -4       |
22 #                  |--------------|                    |  Next (recur)
23 #                  | Parameter n  |                    |  Activation
24 #                  |--------------|                    |  Record
25 #                  | Return addr  |                    |
26 #              ===|==============|===          -------+
27         .globl  main
28 main:   addu    $s7, $zero, $ra # save return address
29
30         li      $v0, 4                  # print prompt
31         la      $a0, prompt
32         syscall
33         li      $v0, 5                  # read n
34         syscall
35         sw      $v0, -8($sp)            # n was in $v0
36         jal     fact                    # call fact
37         lw      $a0, -4($sp)            # get return value off stack
38         li      $v0, 1                  # print ret value = fact(n)
39         syscall
40         li      $v0, 4                  # print newline
41         la      $a0, newline
42         syscall
43
44         addu    $ra, $zero, $s7 # restore return address
45         jr      $ra
46
47
48         .globl  fact
49 fact:   addi    $sp, $sp, -12           # activation rec on stack
50         sw      $ra, 0($sp)             # save return addr on stack
51         lw      $t1, 4($sp)             # $t1 = n
52         li      $t2, 1                  # $t2 = 1
53         bgt     $t1, $t2, recur # goto recur if n > 1
54         sw      $t1, 8($sp)             # put ret val (n) into stack
55         b       endit                   # same ending either way
56 recur: addi    $t3, $t1, -1            # $t3 = n - 1
57         sw      $t3, -8($sp)            # put param n-1 into stack
58         jal     fact                    # call fact
59         lw      $t4, -4($sp)            # $t4 = ret value from stack
60         lw      $t1, 4($sp)             # $t1=n ($t1 may be changed)
61         mul     $t5, $t1, $t4           # $t5 = $t1*$t4 = n*fact(n-1)
62         sw      $t5, 8($sp)             # store ret value into stack
63 endit: lw      $ra, 0($sp)             # restore ra from stack
64         addi    $sp, $sp, 12            # pop activation record
65         jr      $ra                     # return
66
67         .data
68 prompt: .asciiz "\nEnter n: "
69 newline: .asciiz        "\n"
70 ################ Output: ###################################
71 #  Enter n: 6
72 #  720
```

```
 1 # A recursive version of N!
 2 # (version from the text, page 137, which
 3 #  makes extensive use of registers)
 4 #
 5 #  void main()
 6 # {  int N, f;
 7 #    printf("Enter N: ");
 8 #    scanf("%d", N);
 9 #    f = fact(N);
10 #    printf("\nN! = %d", f);
11 #    return;
12 # }
13 # int fact(int n)
14 # {
15 #    if (n < 1) return 1;
16 #    else return (n * fact(n-1));
17 # }
18 # main assumes:
19 #  f is in $s0 and n is in $s1
20         .globl main
21 main:                           # main has to be a global label
22         addu    $s7, $0, $ra    # save return address in global reg
23 ##### Prompt and input the value of n ##############################
24         .data
25 inNmsg: .asciiz "\nEnter n :"
26         .text
27         li      $v0, 4          # print_str (system call 4)
28         la      $a0, inNmsg     # takes address of string as an arg
29         syscall
30         li      $v0, 5
31         syscall
32         add     $s1, $0, $v0    # The value of N was read into $s1
33 ##### Call the factorial function ##################################
34         add     $a0, $0, $s1    # set parameter to N for fact call
35         jal     fact
36         add     $s0, $0, $v0    # f = fact(N);
37 ##### Output the result #########################################
38         .data
39 outMsg: .asciiz "n! = "
40         .text
41         li      $v0, 4          # print_str (system call 4)
42         la      $a0, outMsg     # takes addr of string as argument
43         syscall                 # output the label
44         li      $v0, 1
45         add     $a0, $0, $s0
46         syscall                 # output f
47 #### Output a newline ############################################
48         .data
49 newLn:  .asciiz "\n"
50         .text
51         li      $v0, 4          # print_str (system call 4)
52         la      $a0, newLn      # takes addr of string as argument
53         syscall
54                                 # Usual stuff at end of main
55         addu    $ra, $0, $s7    # restore the return address
56         jr      $ra             # return to the main program
57 #### Definition of fact function ##################################
58 fact:   sub     $sp, $sp, 8     # make space on stack for two items
59         sw      $ra, 4($sp)     # save the return addr on the stack
60         sw      $a0, 0($sp)     # save the argument n on the stack
61         slt     $t0, $a0, 1     # test for n < 1
62         beq     $t0, $zero, L1  # if (n >= 1) go to L1
63         add     $v0, $zero, 1   # otherwise return 1
64         add     $sp, $sp, 8     #  (just pop saved items since no
65         jr      $ra             #  call and return)
66
67 L1:     sub     $a0, $a0, 1     # when n >= 1:  decrement argument
68         jal     fact            # call fact(n-1)
69         lw      $a0, 0($sp)     # restore the value of argument n
70         lw      $ra, 4($sp)     # restore the return address
71         add     $sp, $sp, 8     # release the save area on stack
72         mul     $v0, $a0, $v0   # multiply:  (n*fact(n-1))
73         jr      $ra             # return
```