

CS 2734, Recursive Factorial Program, Page 1 of 1

```

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 {
7     if (n > 1) goto recur;
8     return n;
9     recur: return n * fact(n-1);
10
11 # STACK
12 #-----#
13 # Return addr -----> Old Activation Record
14 #-----#
15 # old $sp ----> <--- new +8
16 # old -4 ----> <--- new +4
17 # old -8 ---->
18 # old -12 ----> <--- new $sp
19 #-----#
20 # Return value -----> New Activation Record
21 #-----#
22 # Parameter n ----->
23 # Return addr -----> Next (recur) Activation Record
24 #-----#
25 # Return value ----->
26 # Parameter n ----->
27 # Return addr ----->
28 #-----#
29 .globl main
30 addu $s7, $zero, $ra # save return address
31
32 li $v0, 4 # print return address
33 la $a0, prompt # print prompt
34
35 syscall
36
37 li $v0, 5 # read n
38
39 sw $v0, -8($sp) # n was in $v0
40 jal fact # call fact
41 lw $a0, -4($sp) # get return value off stack
42 li $v0, 1 # print ret value = fact(n)
43
44 syscall
45
46 li $v0, 4 # print newline
47 la $a0, newline
48
49 syscall
50
51 $ra, $zero, $s7 # restore return address
52 $ra
53 $zero, $zero, $zero
54
55 .globl fact
56 addi $sp, $sp, -12 # activation rec on stack
57 sw $ra, 0($sp) # save return addr on stack
58 lw $t1, 4($sp) # $t1 = n
59 li $t2, 1 # $t2 = 1
60 bgt $t1, $t2, recur # goto recur if n > 1
61 sw $t1, 8($sp) # same ending either way
62 b endit # $t3 = n - 1
63 addi $t3, $t1, -1 # call fact
64 sw $t3, -8($sp) # put param n-1 into stack
65 lw $t4, -4($sp) # $t4 = ret value from stack
66 lw $t1, 4($sp) # $t1 = n (ret value may be changed)
67 mul $t5, $t1, $t4 # $t5 = $t1*$t4 = n*fact(n-1)
68 sw $t5, 8($sp) # store ret value into stack
69 lw $ra, 0($sp) # restore ra from stack
70 addi $sp, $sp, 12 # pop activation record
71 jr $ra # return
72
73 .data
74 prompt: .asciiz "\nEnter n: "
75 newline: .asciiz "\n"
76 ##### Output: #####
77 Enter n: 6
78 720

```

```

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 {
7     int N, f;
8     printf("Enter N: ");
9     scanf("%d", &N);
10    f = fact(N);
11    printf("\nN! = %d", f);
12    return;
13
14 }
15
16 int fact(int n)
17 {
18     if (n < 1) return 1;
19     else return (n * fact(n-1));
20 }
21
22 # main assumes:
23 # f is in $s0 and n is in $s1
24 # .globl main
25
26 main:
27     addu $s7, $0, $ra # main has to be a global label
28     ##### Prompt and input the value of n #####
29     .data
30     inMsg: .asciiz "\nEnter n: "
31     .text
32     li $v0, 4 # print_str (system call 4)
33     la $a0, inMsg # takes address of string as an arg
34
35     syscall
36
37     li $v0, 5
38     syscall
39
40     add $s1, $0, $v0 # The value of N was read into $s1
41     ##### Call the factorial function #####
42     add $a0, $0, $s1 # set parameter to N for fact call
43     jal fact
44     add $s0, $0, $v0 # f = fact(N);
45     ##### Output the result #####
46     .data
47     outMsg: .asciiz "n! = "
48     .text
49     li $v0, 4 # print_str (system call 4)
50     la $a0, outMsg # takes addr of string as argument
51
52     syscall
53     li $v0, 1 # output the label
54     add $a0, $0, $s0
55     syscall # output f
56
57     ##### Output a newline #####
58     .data
59     newLn: .asciiz "\n"
60     .text
61     li $v0, 4 # print_str (system call 4)
62     la $a0, newLn # takes addr of string as argument
63
64     syscall
65
66     $ra, $0, $s7 # Usual stuff at end of main
67     jr $ra # restore the return address
68     ##### Definition of fact function #####
69     sub $sp, $sp, 8 # make space on stack for two items
70     $ra, 4($sp) # save the return addr on the stack
71     sw $a0, 0($sp) # save the argument n on the stack
72     $t0, $a0, 1 # test for n < 1
73     slt beg, $t0, $zero, 1 # if (n >= 1) go to L1
74     beq $v0, $zero, 1 # otherwise return 1
75     add $sp, $sp, 8 # (just pop saved items since no
76     jr $ra # call and return)
77
78 L1:
79     sub $a0, $a0, 1 # when n >= 1: decrement argument
80     jal fact # call fact(n-1)
81     lw $a0, 0($sp) # restore the value of argument n
82     lw $ra, 4($sp) # restore the return address
83     add $sp, $sp, 8 # release the save area on stack
84     mul $v0, $a0, $v0 # multiply: (n*fact(n-1))
85     jr $ra # return

```