```
# CS 2734, Fall 2001
# Switch example (PH p. 129)
#   f = 0; g = 5; h = -20; i = 13; j = 3; k = 2;
#   scanf("%i", &k);
#   switch (k) {
#       case 0:  f = i + j; break;
#       case 1:  f = g + h; break;
#       case 2:  f = g - h; break;
#       case 3:  f = i - j; break;
#       default: break;
#   }
# assumes:
#  variables f through  j are in registers $s0 through $s5
#  $t2 has 4 and $t4 has the JumpTable

        .globl main
main:                           # main has to be a global label
        addu    $s7, $0, $ra    # save the return address in a global register

## Initialize variables
        add     $s0, $0, $0     # f = 0
        addi    $s1, $0, 5      # g = 5
        addi    $s2, $0, -20    # h = -20
        addi    $s3, $0, 13     # i = 13
        addi    $s4, $0, 3      # j = 3
        addi    $s5, $0, 2      # k = 2 (but input k below)
        addi    $t2, $0, 4      # register $t2 has 4
        la      $t4, JumpTable  # register $t4 has JumpTable

## Input a value for k, with prompt message
        li      $v0, 4          # print_str (system call 4)
        la      $a0, prompt     # takes the address of string as an argument
        syscall
        li      $v0, 5          # read_int (system call 5)
        syscall                 # value is in $v0
        add     $s5, $0, $v0    # set k ($s0) = value read

## Handle k not in [0, 3] first
        slt     $t3, $s5, $zero # if (k < 0)
        bne     $t3, $0, Exit   #    go to Exit
        slt     $t3, $s5, $t2   # if (k >= 4)
        beq     $t3, $0, Exit   #    go to Exit

## switch (k) performed by using a table of jump addresses
        add     $t1, $s5, $s5   #
        add     $t1, $t1, $t1   # $t1 is k*4
        add     $t1, $t1, $t4   # put address of JumpTable[k] in $t1
        lw      $t0, 0($t1)     # $t0 has value of JumpTable[k]
        jr      $t0             # Execute switch based on index k

L0:     add     $s0, $s3, $s4   # case 0: f = i + j
        j       Exit            #   break
L1:     add     $s0, $s1, $s2   # case 1: f = g + h
        j       Exit            #   break
L2:     sub     $s0, $s1, $s2   # case 2: f = g - h
        j       Exit            #   break
L3:     sub     $s0, $s3, $s4   # case 3: f = i - j
Exit:                           # end of switch

## print the values of k and f
        li      $v0, 4          # print_str (system call 4)
        la      $a0, message1   # takes the address of string as an argument
        syscall
        li      $v0, 1          # print_int (system call 1)
        add     $a0, $0, $s5    # put value f in $a0 for printing
        syscall
        li      $v0, 4          # print_str (system call 4)
        la      $a0, blank      # takes the address of string as an argument
        syscall
        li      $v0, 1          # print_int (system call 1)
        add     $a0, $0, $s0    # put value f in $a0 for printing
        syscall
        li      $v0, 4          # print_str (system call 4)
        la      $a0, newl       # takes the address of string as an argument
        syscall

## Usual stuff at the end of the main
        addu    $ra, $0, $s7    # restore the return address
        jr      $ra             # return to the main program

## Here is the data
        .data
JumpTable:      .word   L0, L1, L2, L3
prompt:         .asciiz "  Input a value for k:"
message1:       .asciiz "  The values of k and f are: "
newl:           .asciiz "\n"
blank:          .asciiz " "

###### OUTPUT #########
# four06% spim -file switch.s
#   Input a value for k:0
#   The values of k and f are: 0 16
# four06% spim -file switch.s
#   Input a value for k:1
#   The values of k and f are: 1 -15
# four06% spim -file switch.s
#   Input a value for k:2
#   The values of k and f are: 2 25
# four06% spim -file switch.s
#   Input a value for k:3
#   The values of k and f are: 3 10
# four06% spim -file switch.s
#   Input a value for k:-1
#   The values of k and f are: -1 0
# four06% spim -file switch.s
#   Input a value for k:4
#   The values of k and f are: 4 0
```