```
#Derek Albosta
        .data
nums:   .word 5, 8, 12, 14, 16, 19, 23, 28, 35, 39, 41, 43, 44, 52, 55, 58, 66,
72, 74, 76, 81
length: .word 21
str:    .asciiz "Searching from index "
to:     .asciiz " to "
endl:   .asciiz "\n"

        .text
        #
        # This code loads arguments into $a registers and calls the search routi
ne.
        # Once we get back from sorting it prints the returned value and exits.
        #
main:
        li $a0, 0       # left index value
        la $t0, length  # size is right index value
        lw $a1, 0($t0)
        la $a2, nums    # pass array's base address in $a2
        li $a3, 19      # value to search for in array
        # Args are all loaded into $a register -- time to jump to search procedu
re
        jal binary_search
        # Now we're back, with the return value in $v0
        move $a0, $v0   # Move result into $a0 to print
        li $v0, 1
        syscall         # Print the result
        li $v0, 10      # syscall 10 is exit
        syscall


# PROCEDURE:  print_status
#  Prints a line of output that describes current search region.
#  NOTE: This routine alters the $a register values.
#
# Inputs:
#  $a0  Low index of search region
#  $a1  High index of search region
#
# Outputs:
#  None

print_status:
        addi $sp, $sp, -8   # Make room for two words on stack
        sw   $a0, 0($sp)    # Store initial $a0 value on stack
        sw   $a1, 4($sp)    # Store initial $a1 value on stack
        # Get on with the printing
        la $a0, str     # put address of str in $a0 for syscall
        li $v0, 4       # print string syscall #
        syscall         # print the bulk of the string
        lw $a0, 0($sp)  # bring $a0 (low) in from stack
        li $v0, 1       # print integer syscall #
        syscall
        la $a0, to      # put address of " to " string in $a0
        li $v0, 4       # print string syscall #
        syscall
        lw $a0, 4($sp)  # bring in $a1 (high) from stack, put in $a0
        li $v0, 1       # print integer syscall #
        syscall
        la $a0, endl    # put address of newline string in $a0
        li $v0, 4
        syscall
        addi $sp, $sp, 8
        jr $ra


# PROCEDURE:  binary_search
#  Searches for a specific value in an array using binary search.
#
# Inputs:
#  $a0  Index where search begins (inclusive)
```

```
#  $a1  Index where search ends (exclusive)
#  $a2  Base address of array
#  $a3  Value to search for
#
# Outputs:
#  $v0  Contains the position within the array at which value occurs, or
#       where it *would* be located if it's not in the array currently.

binary_search:
        #setup stack
        addi $sp, $sp, -12   # make sure stack has room
        sw   $a0, 0($sp)
        sw   $a1, 4($sp)
        sw   $ra, 8($sp)

        li $a0, 0 #value of low
        lw $a1, length #value of high

        jal print_status #print range

        #if low+1 == high, result found
        addi $t1, $a0, 1
        bne $t1, $a1, recurse #if not found recurse
        j end

recurse:
        #find midpoint
        add $t2, $a0, $a1 #sum of high and low
        sra $t4, $t2, 1   # divide by 2

        #find value of midpoint
        sll $t5, $t4, 2 #calculate 1 shift
        add $t5, $t5, $a2 # move over by midpoint
        lw $t5, 0($t5) #get value of mid


        #check whether num is bigger or smaller than mid
        slt $t6, $t5, $a3
        addi $t7, $zero, 1
        beq $t7, $t6, higher #value is higher than mid

        #value is lower than mid
        move $a1, $t4 #set high to mid
        jal binary_search
        j end

higher:
        #value is higher than mid
        move $a0, $t4 #set low to mid
        jal binary_search
        j end

end:
        #restore registers
        lw $a0, 0($sp)
        lw $a1, 4($sp)
        lw $ra, 8($sp)

        #pop stack
        addi $sp, $sp, 12
        jr $ra
```