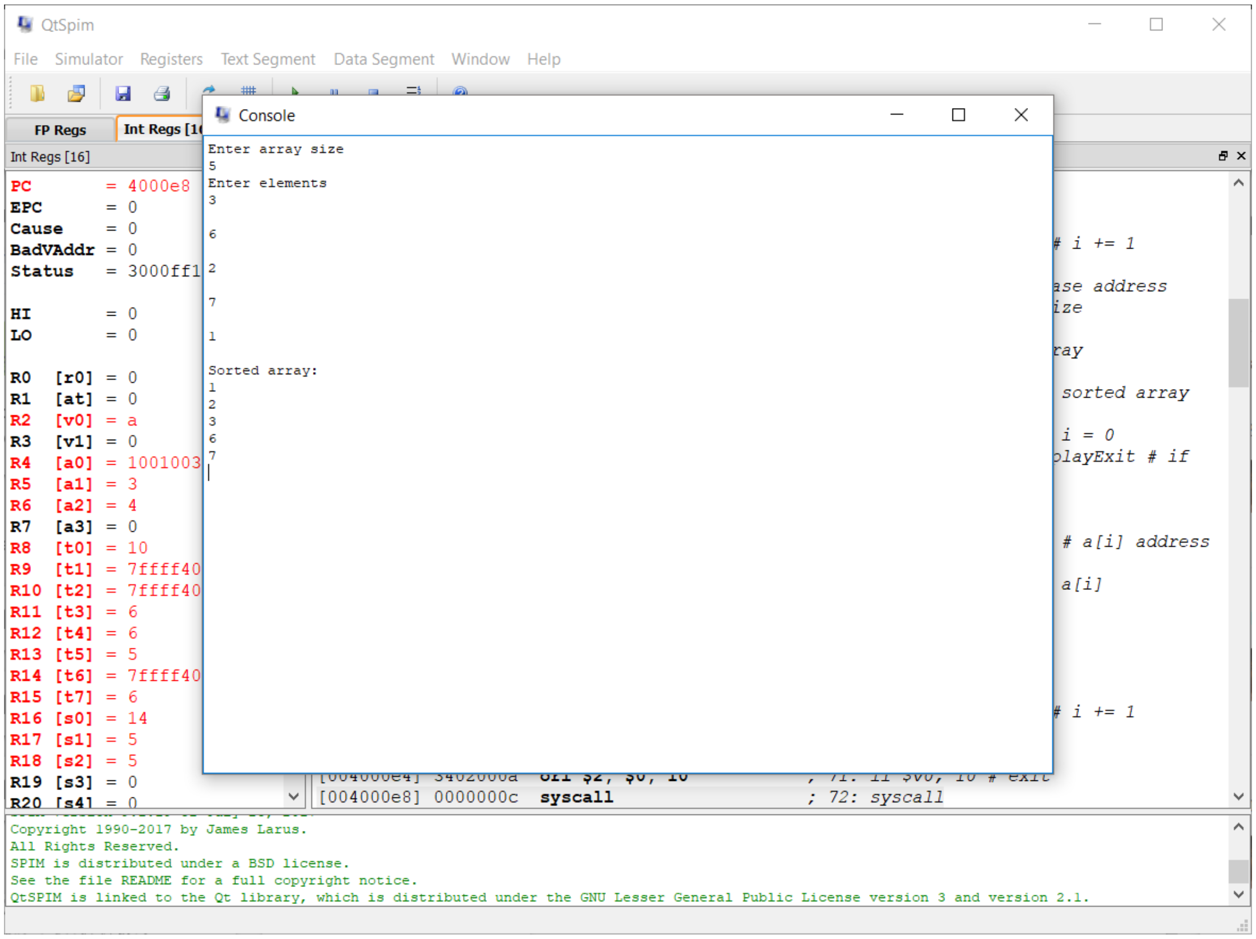
```
#   Zayadur Khan    03/29/18    CIS-341

.data
    sizePrompt:     .asciiz "Enter array size \n"
    entryPrompt:    .asciiz "Enter elements \n"
    sortedArray:    .asciiz "Sorted array: \n"
    lineBreak:      .asciiz "\n"

.text
.globl main
    main:
        la $a0, sizePrompt      # ask for array size
        li $v0, 4               # print prompt
        syscall

        li $v0, 5               # read input
        syscall

        move $s2, $v0
        sll $s0, $v0, 2
        sub $sp, $sp, $s0
        la $a0, entryPrompt     # ask for entries
        li $v0, 4
        syscall

        move $s1, $zero         # i = 0

    # read elements of array in, 1 line at a time
    get:
        bge $s1, $s2, getExit  # if i>=n go to getExit
        sll $t0, $s1, 2
        add $t1, $t0, $sp
        li $v0, 5               # read input
        syscall

        sw $v0, 0($t1)          # The element is stored
        la $a0, lineBreak
        li $v0, 4
        syscall

        addi $s1, $s1, 1    # i += 1
        j get

    getExit:
        move $a0, $sp       # base address
    move $a1, $s2       # size
    jal sort            # sort

la $a0, sortedArray
li $v0, 4               # print sorted array
syscall

move $s1, $zero         # i = 0
display:
    bge $s1, $s2, displayExit     # if i>=n go to displayExit
    sll $t0, $s1, 2
    add $t1, $sp, $t0           # a[i] address
    lw $a0, 0($t1)
    li $v0, 1                   # print a[i]
    syscall

la $a0, lineBreak
li $v0, 4
syscall

addi $s1, $s1, 1        # i += 1
j display

displayExit:
    add $sp, $sp, $s0
    li $v0, 10          # exit
    syscall

# sort procedure
sort:
    addi $sp, $sp, -20      # save values on stack
    sw $ra, 0($sp)
    sw $s0, 4($sp)
    sw $s1, 8($sp)
    sw $s2, 12($sp)
    sw $s3, 16($sp)
    move $s0, $a0           # base address of the array
    move $s1, $zero         # i=0
    sub $s2, $a1, 1         # lenght -1

sortFor:
    bge $s1, $s2, exitSort      # if i >= length-1 -> exit loop
    move $a0, $s0               # base address
    move $a1, $s1               # i
    move $a2, $s2               # length - 1
    jal minimum

    move $s3, $v0               # return value of minimum
    move $a0, $s0               # array
    move $a1, $s1               # i
    move $a2, $s3               # minimum
    jal swap
    addi $s1, $s1, 1            # i += 1
    j sortFor

# restore stack
exitSort:
    lw $ra, 0($sp)
    lw $s0, 4($sp)
    lw $s1, 8($sp)
    lw $s2, 12($sp)
    lw $s3, 16($sp)
    addi $sp, $sp, 20
    jr $ra

minimum:
    move $t0, $a0               # base of the array
    move $t1, $a1
    move $t2, $a2               # last
    sll $t3, $t1, 2             # first * 4
    add $t3, $t3, $t0           # base array + first * 4
    lw $t4, 0($t3)              # min = v[first]
    addi $t5, $t1, 1            # i = 0

forMinimum:
    bgt $t5, $t2, endMinimum
    sll $t6, $t5, 2
    add $t6, $t6, $t0           # base array + i * 4
    lw $t7, 0($t6)
    bge $t7, $t4, exitMinimum   # skip when v[i] >= min
    move $t1, $t5               # minimum = i
    move $t4, $t7               # min = v[i]

exitMinimum:
    addi $t5, $t5, 1            # i += 1
    j forMinimum

endMinimum:
    move $v0, $t1
    jr $ra
```

```
swap:
    sll $t1, $a1, 2
    add $t1, $a0, $t1           # v + i * 4
    sll $t2, $a2, 2
    add $t2, $a0, $t2           # v + j * 4
    lw $t0, 0($t1)              # v[i]
    lw $t3, 0($t2)              # v[j]
    sw $t3, 0($t1)              # v[i] = v[j]
    sw $t0, 0($t2)              # v[j] = $t0
    jr $ra
```



```
Enter array size
5
Enter elements
3

6

2

7

1

Sorted array:
1
2
3
6
7
```

```
PC        = 4000e8
EPC       = 0
Cause     = 0
BadVAddr  = 0
Status    = 3000ff1

HI        = 0
LO        = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = a
R3  [v1] = 0
R4  [a0] = 1001003
R5  [a1] = 3
R6  [a2] = 4
R7  [a3] = 0
R8  [t0] = 10
R9  [t1] = 7ffff40
R10 [t2] = 7ffff40
R11 [t3] = 6
R12 [t4] = 6
R13 [t5] = 5
R14 [t6] = 7ffff40
R15 [t7] = 6
R16 [s0] = 14
R17 [s1] = 5
R18 [s2] = 5
R19 [s3] = 0
R20 [s4] = 0
```

```
# i += 1
ase address
ize
ray
 sorted array
 i = 0
playExit # if
 # a[i] address
 a[i]
# i += 1
[004000e8] 0000000c  syscall                 ; 72: syscall
```

```
Copyright 1990-2017 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
```