Name: Phan Trần Thanh Huy

ID: ITCSIU22056

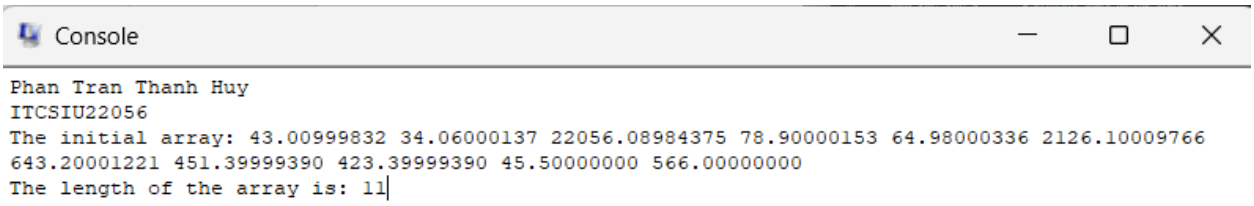
Lab 9

Exercise 1:

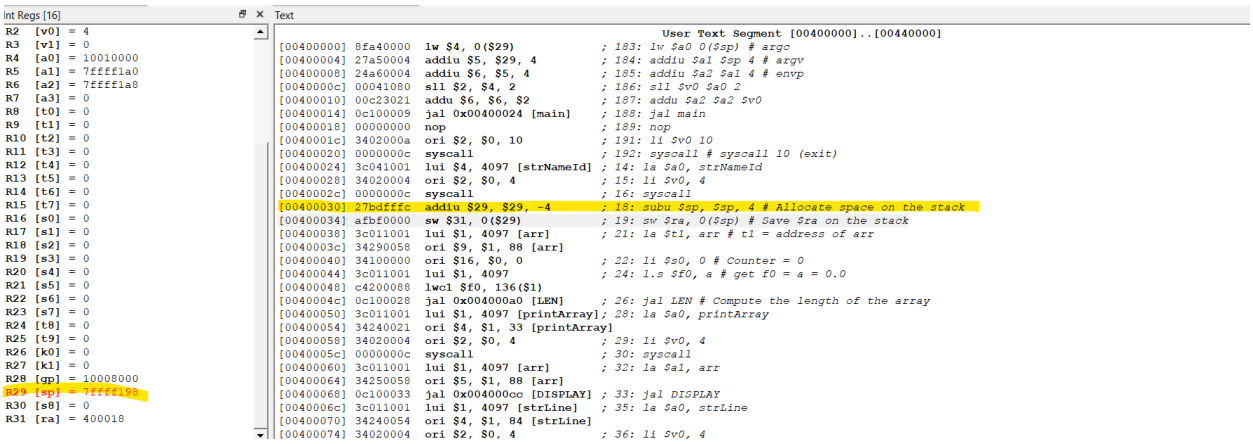
Code:

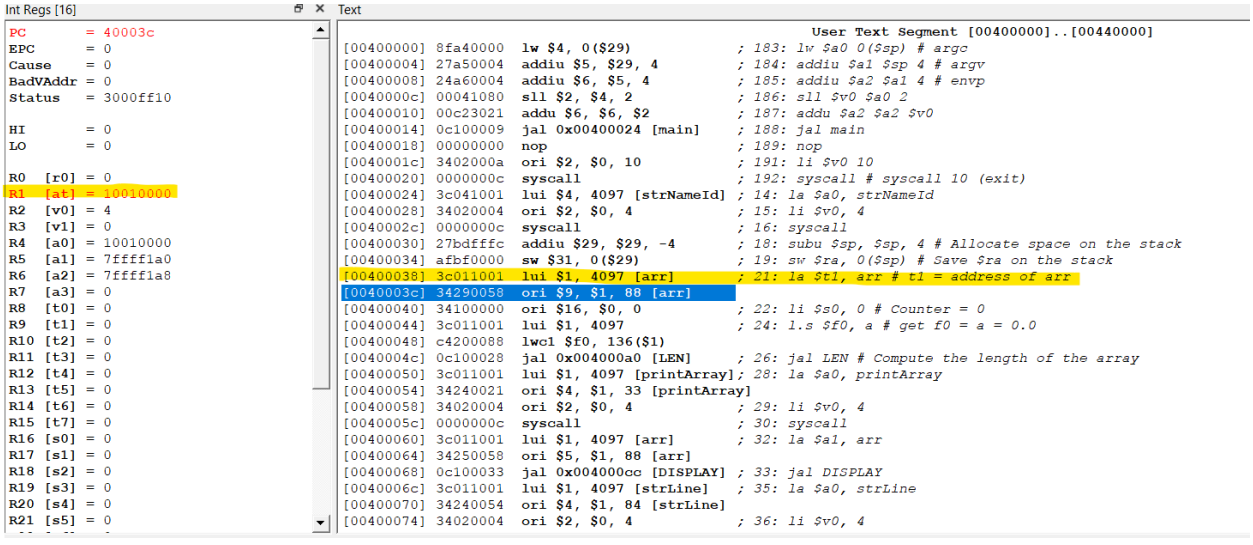
```
.data
strNameId: .asciiz "Phan Tran Thanh Huy\nITCSIU22056\n"
printArray: .asciiz "The initial array: "
printLength: .asciiz "The length of the array is: "
strSpace: .asciiz " "
strLine: .asciiz "\n"
arr: .float 43.01, 34.06, 22056.09, 78.9, 64.98, 2126.1, 643.2, 451.4, 423.4, 45.5, 566.0, 0.0
a: .float 0.0

.text
.globl main

main:
    la $a0, strNameId
    li $v0, 4
    syscall

    subu $sp, $sp, 4       # Allocate space on the stack
    sw $ra, 0($sp)         # Save $ra on the stack

    la $t1, arr         # t1 = address of arr
    li $s0, 0           # Counter = 0

    l.s $f0, a          # get f0 = a = 0.0

    jal LEN             # Compute the length of the array

    la $a0, printArray
    li $v0, 4
    syscall

    la $a1, arr
    jal DISPLAY

    la $a0, strLine
    li $v0, 4
    syscall

    la $a0, printLength
    li $v0, 4
    syscall

    move $a0, $s0          # Print the length of the array
    li $v0, 1
    syscall

    li $v0, 10
    syscall

LEN:
    lwc1 $f2, 0($t1)       # Load the first float from the array

    c.eq.s $f2, $f0        # Check if the float is 0.0 (end of array)
    bc1t END_FUNC          # If true, go to END_FUNC

    addi $t1, $t1, 4       # Move to the next element

    subu $sp, $sp, 4       # Allocate space on the stack
    sw $ra, 0($sp)         # Save $ra on the stack

    jal LEN             # Call the recursive

    lw $ra, 0($sp)         # Load $ra on the stack
    addu $sp, $sp, 4       # Move to the next element

    addi $s0, $s0, 1       # Increase counter

    j $ra               # Return the $ra

DISPLAY:
    lwc1 $f2, 0($a1)

    c.eq.s $f2, $f0        # Check if the float is 0.0 (end of array)
    bc1t END_FUNC          # If true, go to END_FUNC

    mov.s $f12, $f2
    li $v0, 2
    syscall

    la $a0, strSpace
    li $v0, 4
    syscall

    addi $a1, $a1, 4
    j DISPLAY

END_FUNC:
    j $ra
```

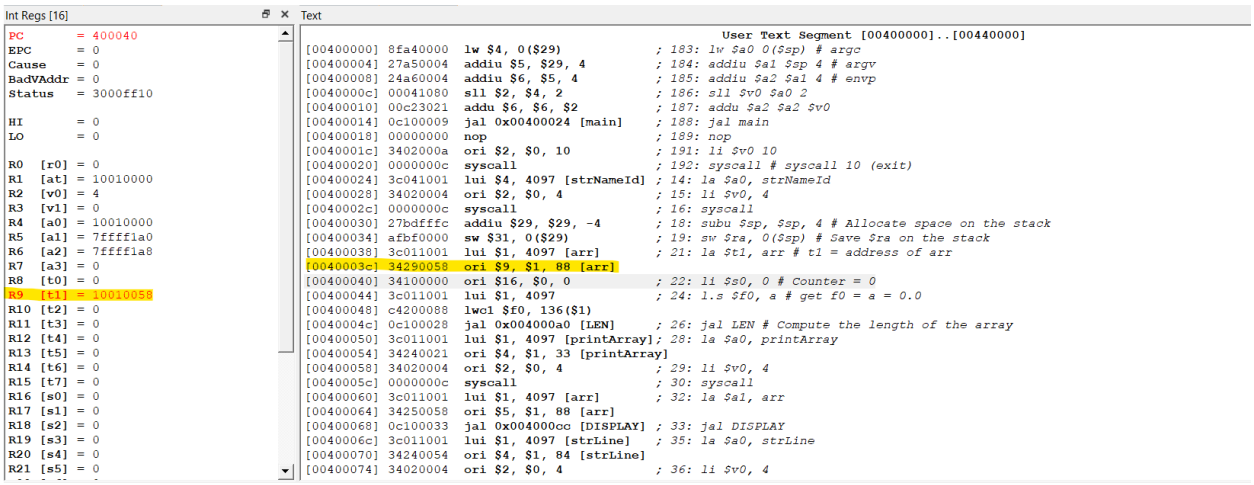
Test case:



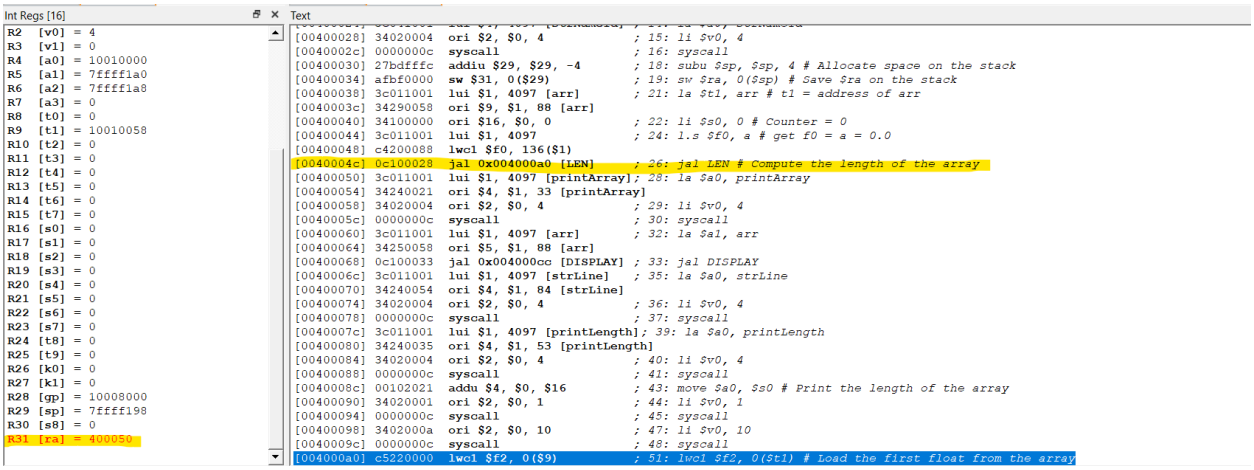
Single Step:

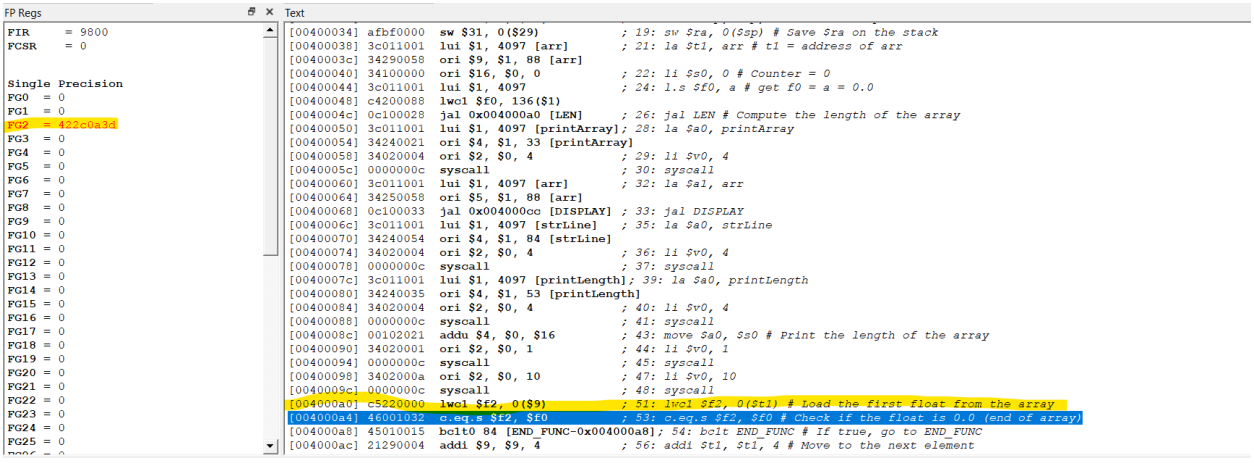


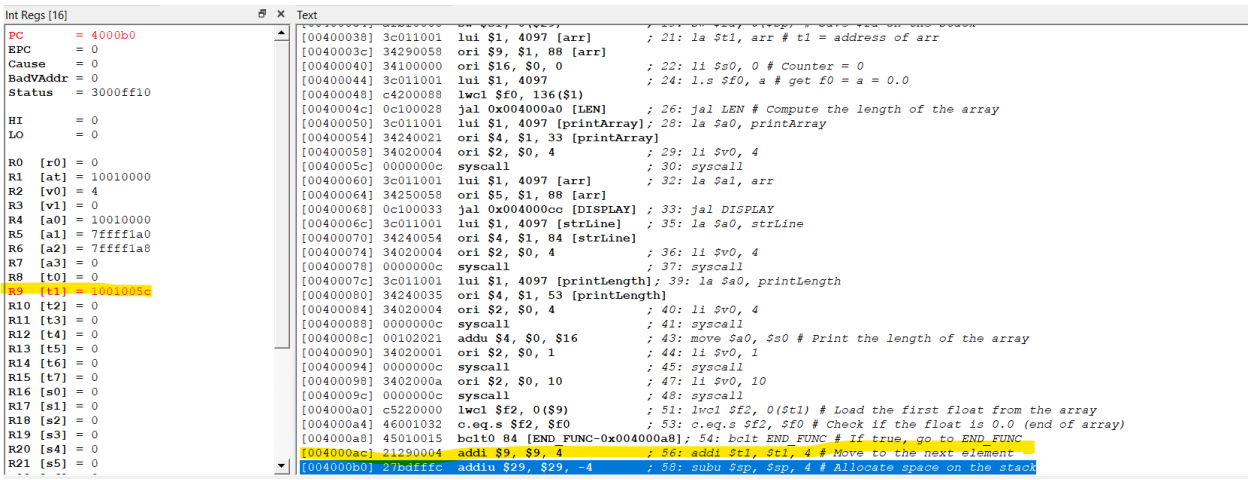


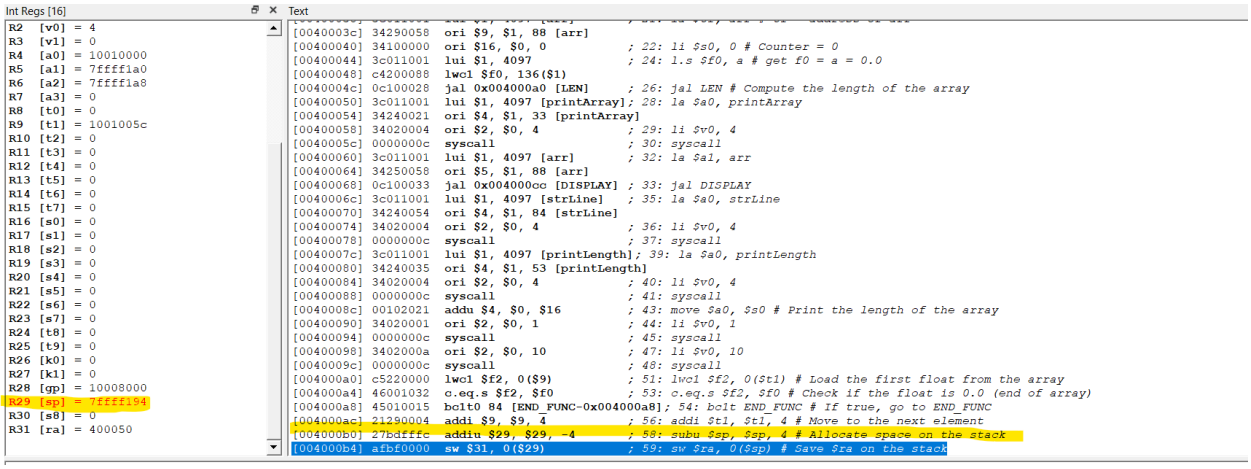


Save the first element to the stack:

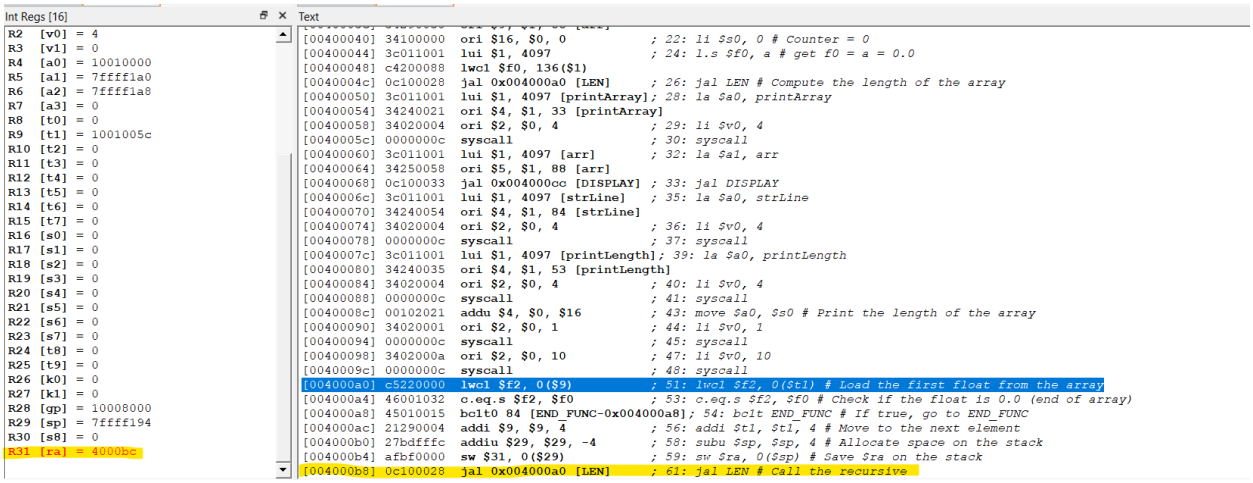


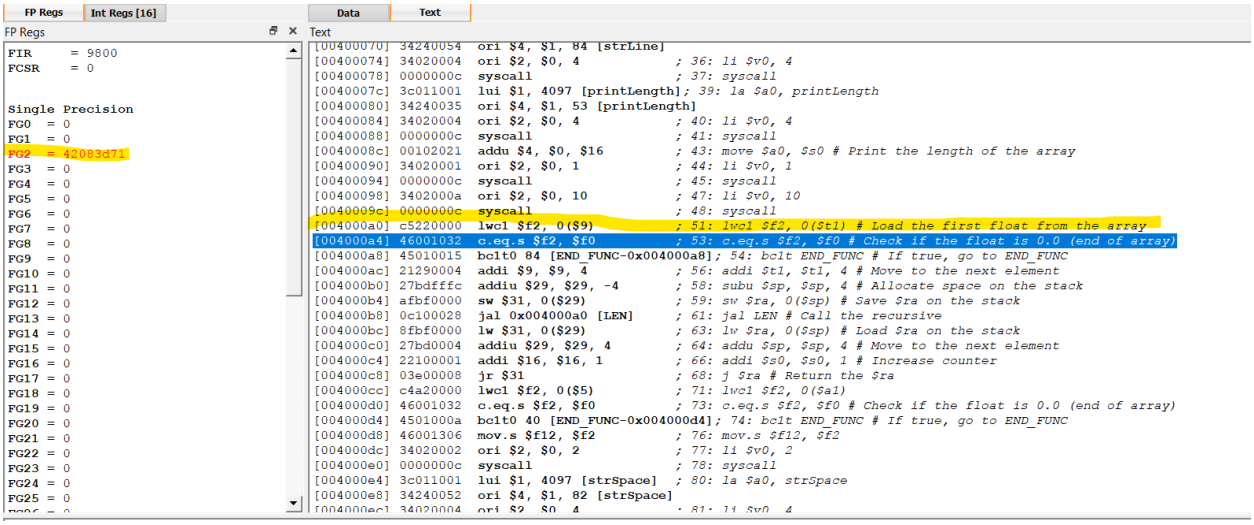


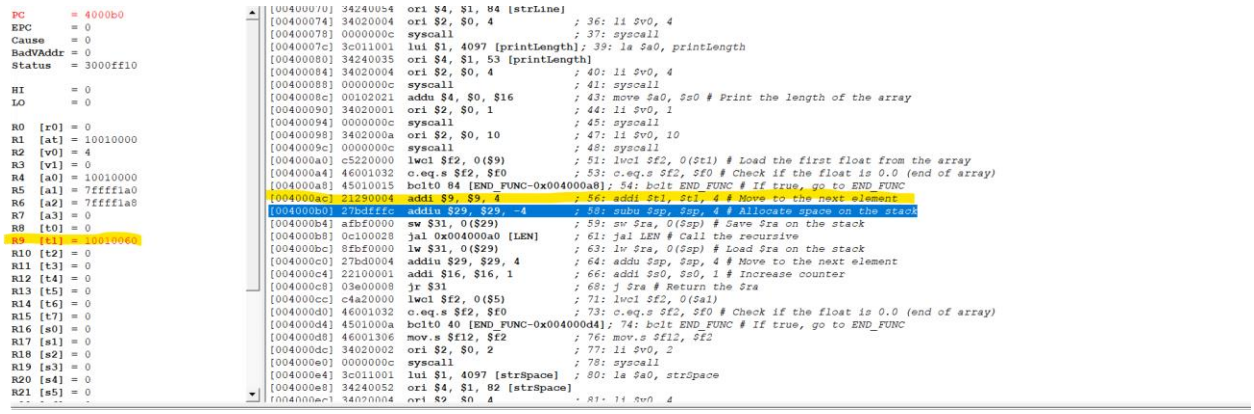


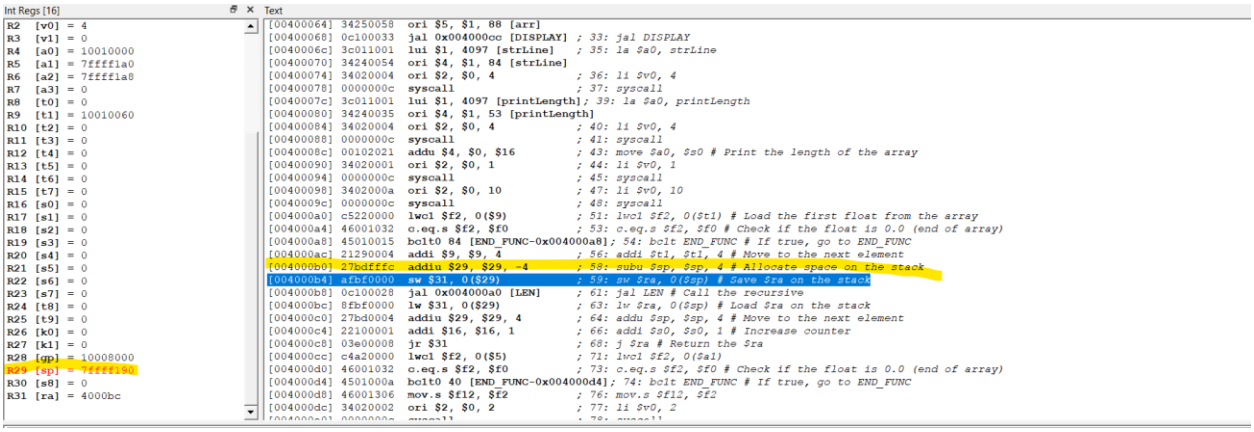


Save the second element to the stack:





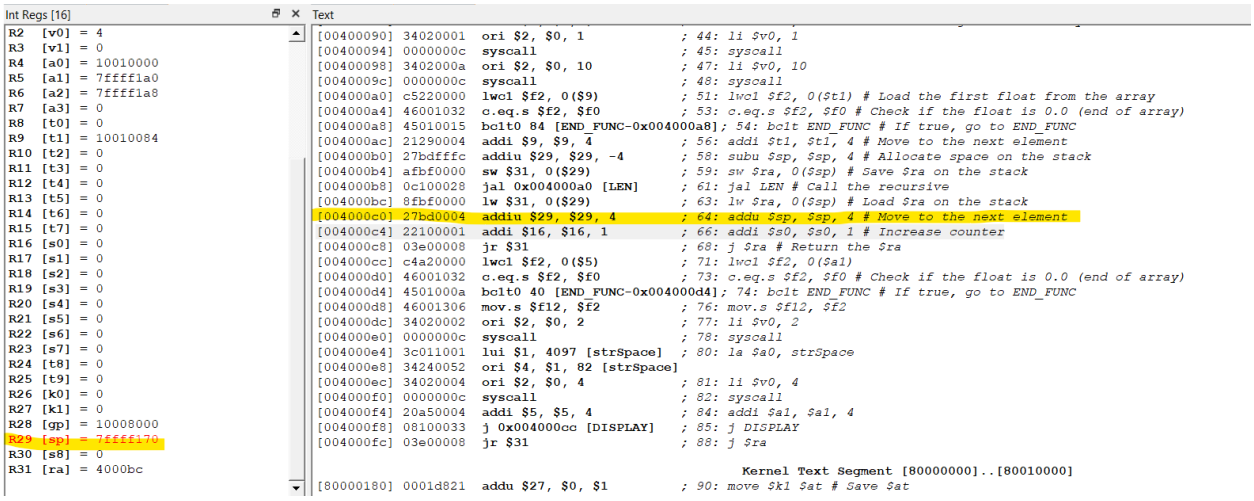


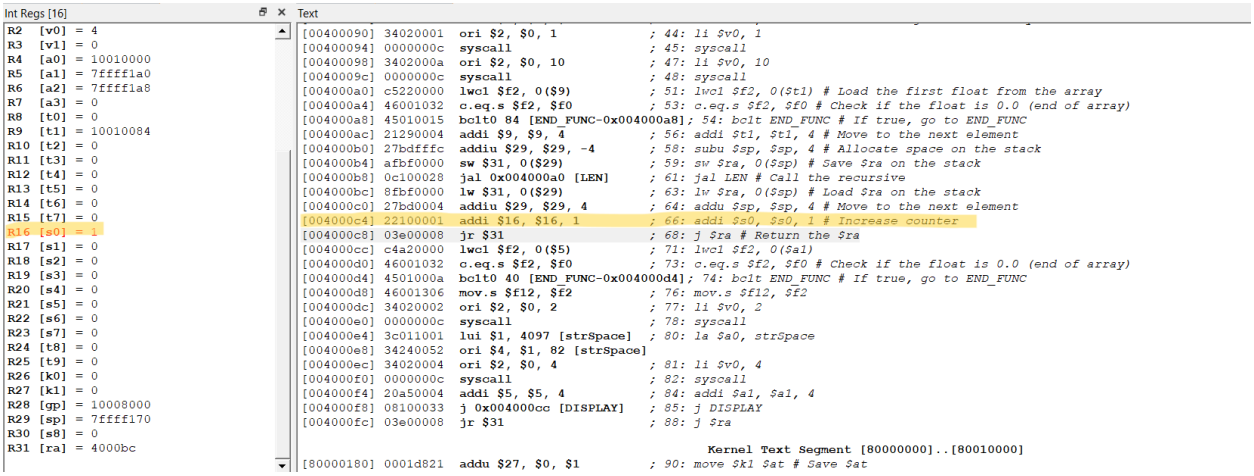


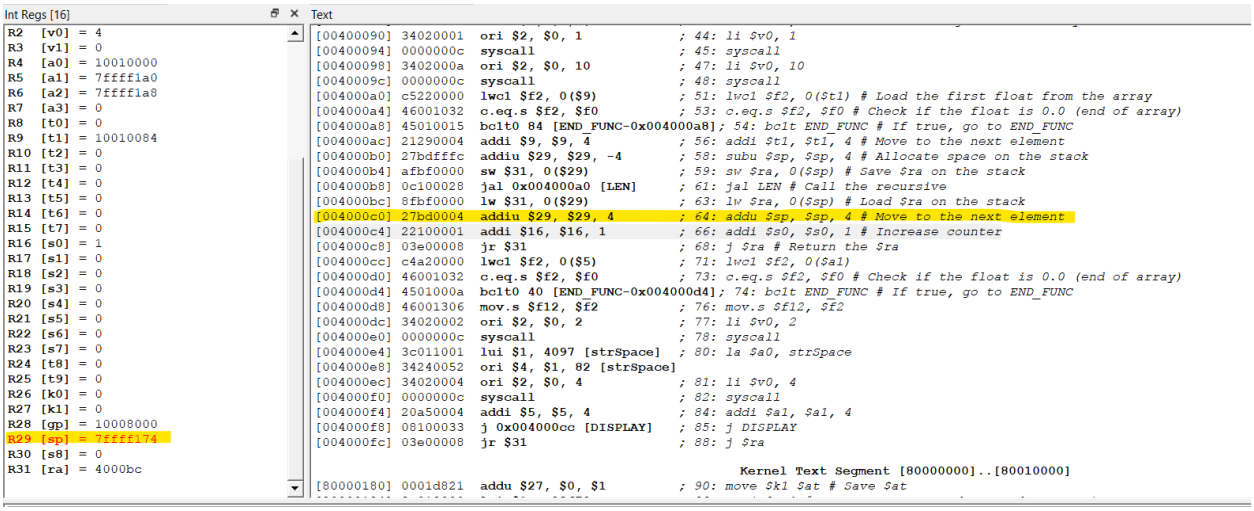
It will repeat the same step until a[i] = 0.0

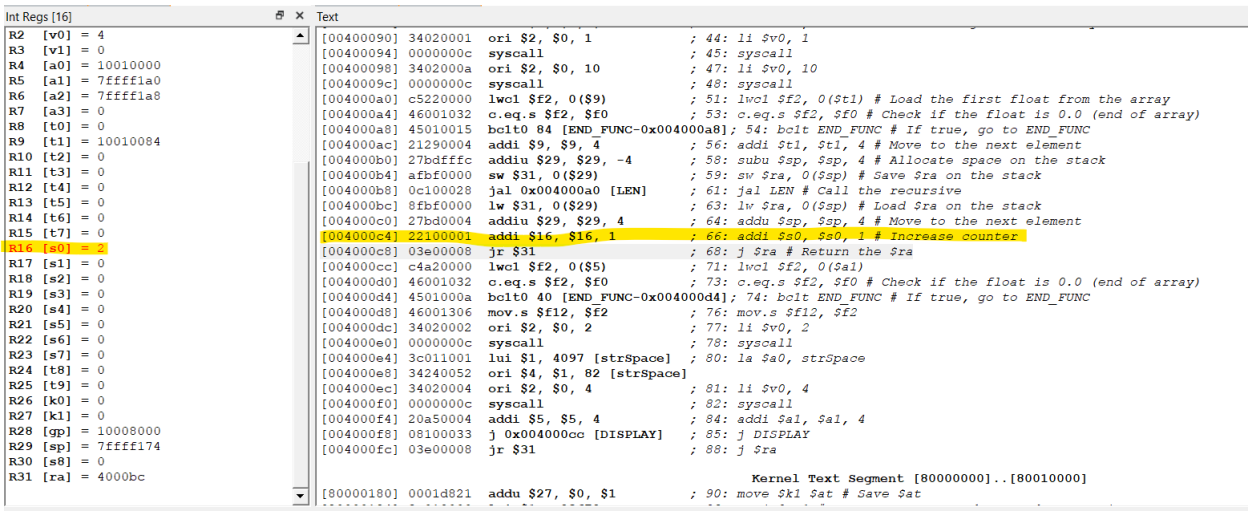
After save all the element to the stack, It will load the stack

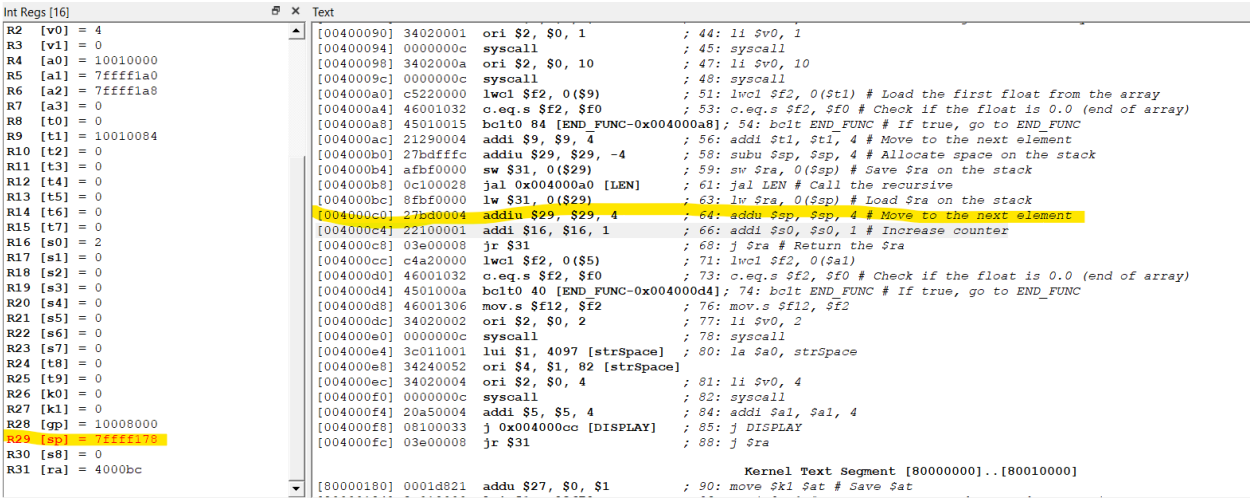
Load the first element

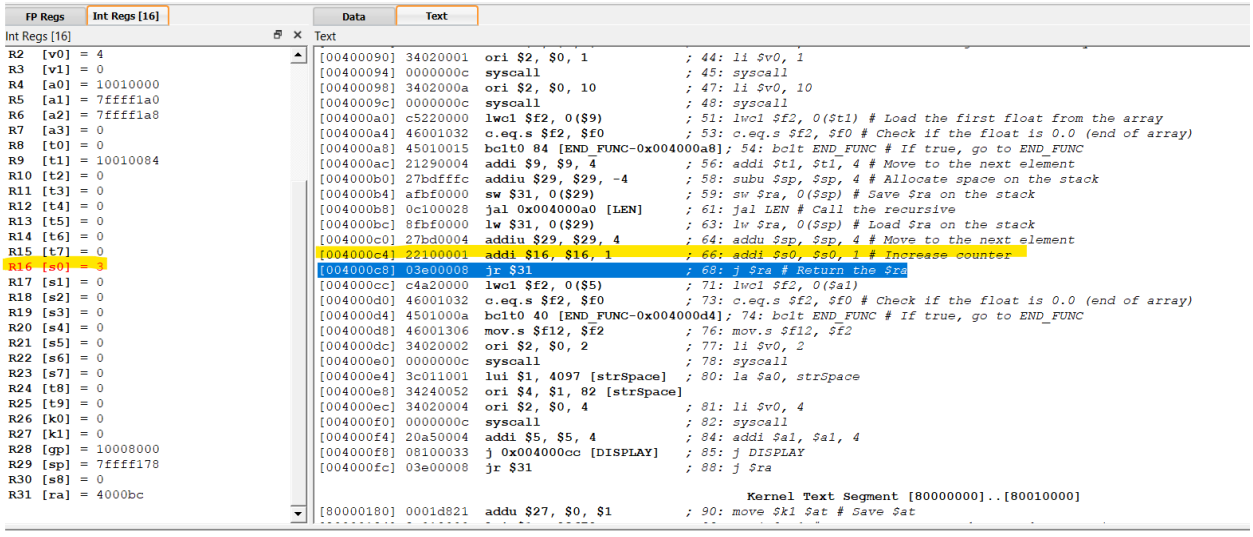








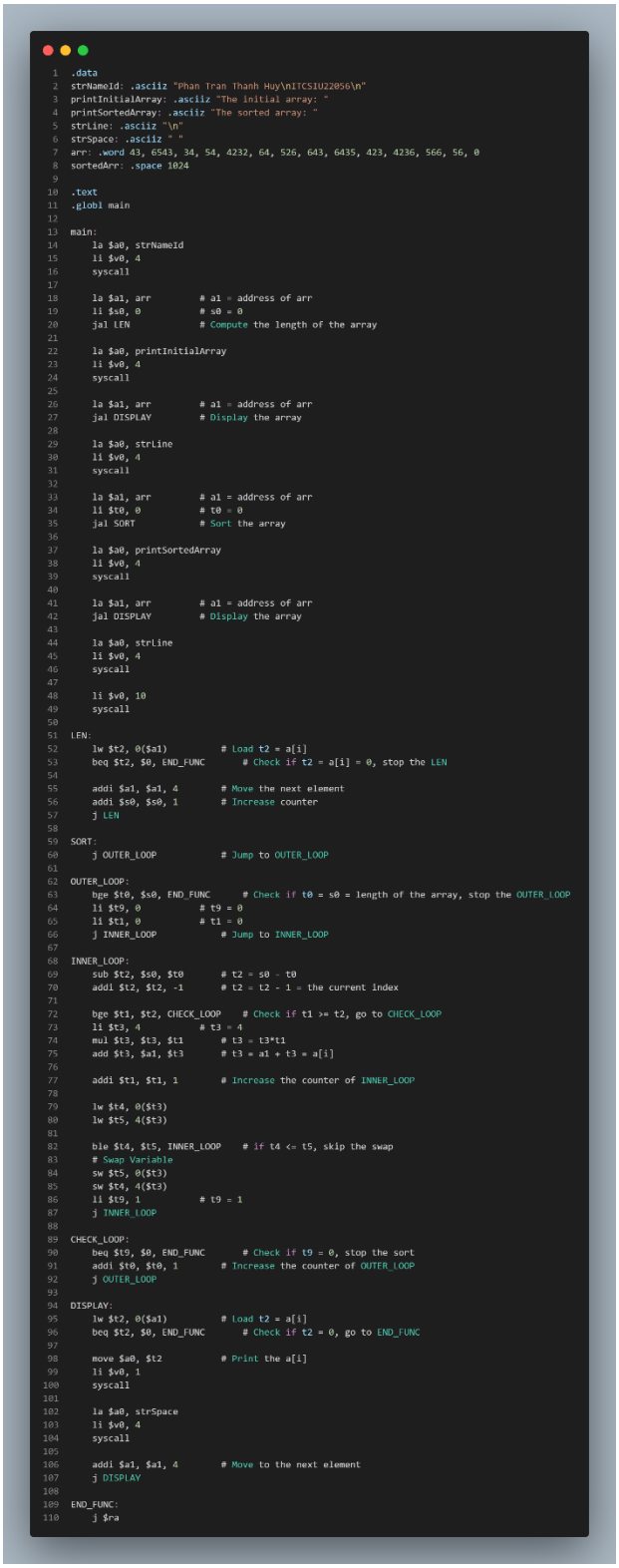




It will repeat the same step until loading all the element in the stack.

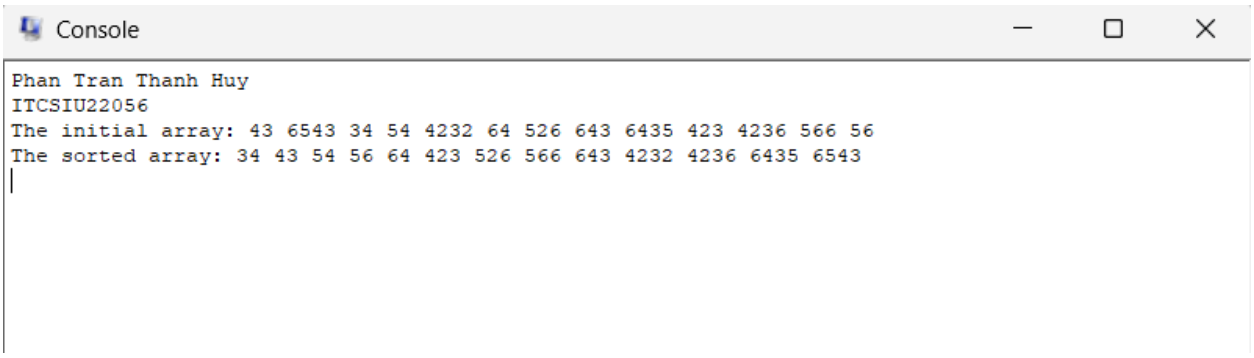
Exercise 2:

Code (Bubble_Sort):



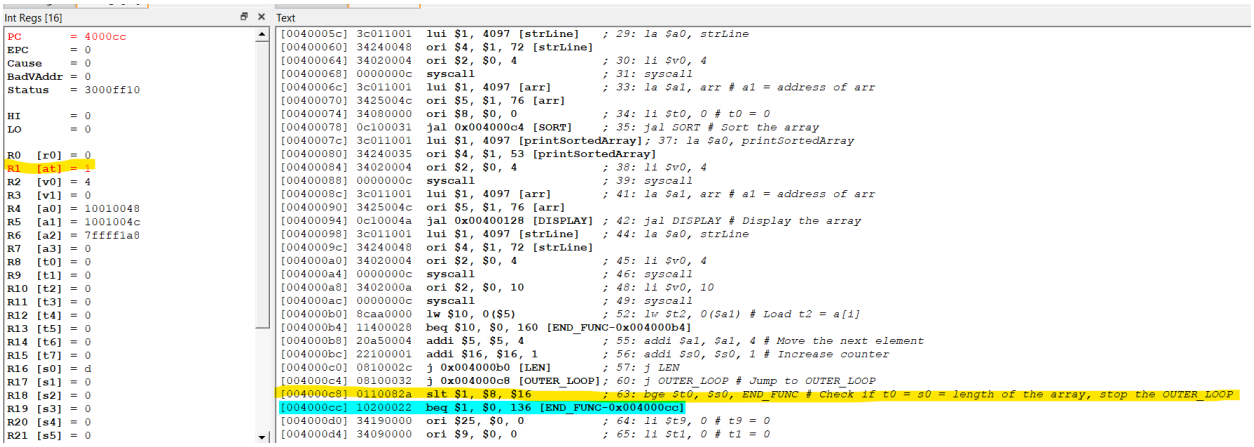
```
.data
strNameId: .asciiz "Phan Tran Thanh Huy\nITCSIU22056\n"
printInitialArray: .asciiz "The initial array: "
printSortedArray: .asciiz "The sorted array: "
strLine: .asciiz "\n"
strSpace: .asciiz " "
arr: .word 43, 6543, 34, 54, 4232, 64, 526, 643, 6435, 423, 4236, 566, 56, 0
sortedArr: .space 1024

.text
.globl main

main:
    la $a0, strNameId
    li $v0, 4
    syscall

    la $a1, arr         # a1 = address of arr
    li $s0, 0           # s0 = 0
    jal LEN             # Compute the length of the array

    la $a0, printInitialArray
    li $v0, 4
    syscall

    la $a1, arr         # a1 = address of arr
    jal DISPLAY         # Display the array

    la $a0, strLine
    li $v0, 4
    syscall

    la $a1, arr         # a1 = address of arr
    li $t0, 0           # t0 = 0
    jal SORT            # Sort the array

    la $a0, printSortedArray
    li $v0, 4
    syscall

    la $a1, arr         # a1 = address of arr
    jal DISPLAY         # Display the array

    la $a0, strLine
    li $v0, 4
    syscall

    li $v0, 10
    syscall

LEN:
    lw $t2, 0($a1)          # Load t2 = a[i]
    beq $t2, $0, END_FUNC       # Check if t2 = a[i] = 0, stop the LEN

    addi $a1, $a1, 4        # Move the next element
    addi $s0, $s0, 1        # Increase counter
    j LEN

SORT:
    j OUTER_LOOP            # Jump to OUTER_LOOP

OUTER_LOOP:
    bge $t0, $s0, END_FUNC      # Check if t0 = s0 = length of the array, stop the OUTER_LOOP
    li $t9, 0           # t9 = 0
    li $t1, 0           # t1 = 0
    j INNER_LOOP            # Jump to INNER_LOOP

INNER_LOOP:
    sub $t2, $s0, $t0       # t2 = s0 - t0
    addi $t2, $t2, -1       # t2 = t2 - 1 = the current index

    bge $t1, $t2, CHECK_LOOP    # Check if t1 >= t2, go to CHECK_LOOP
    li $t3, 4           # t3 = 4
    mul $t3, $t3, $t1       # t3 = t3*t1
    add $t3, $a1, $t3       # t3 = a1 + t3 = a[i]

    addi $t1, $t1, 1        # Increase the counter of INNER_LOOP

    lw $t4, 0($t3)
    lw $t5, 4($t3)

    ble $t4, $t5, INNER_LOOP    # if t4 <= t5, skip the swap
    # Swap Variable
    sw $t5, 0($t3)
    sw $t4, 4($t3)
    li $t9, 1           # t9 = 1
    j INNER_LOOP

CHECK_LOOP:
    beq $t9, $0, END_FUNC       # Check if t9 = 0, stop the sort
    addi $t0, $t0, 1        # Increase the counter of OUTER_LOOP
    j OUTER_LOOP

DISPLAY:
    lw $t2, 0($a1)          # Load t2 = a[i]
    beq $t2, $0, END_FUNC       # Check if t2 = 0, go to END_FUNC

    move $a0, $t2           # Print the a[i]
    li $v0, 1
    syscall

    la $a0, strSpace
    li $v0, 4
    syscall

    addi $a1, $a1, 4        # Move to the next element
    j DISPLAY

END_FUNC:
    j $ra
```

Test case:



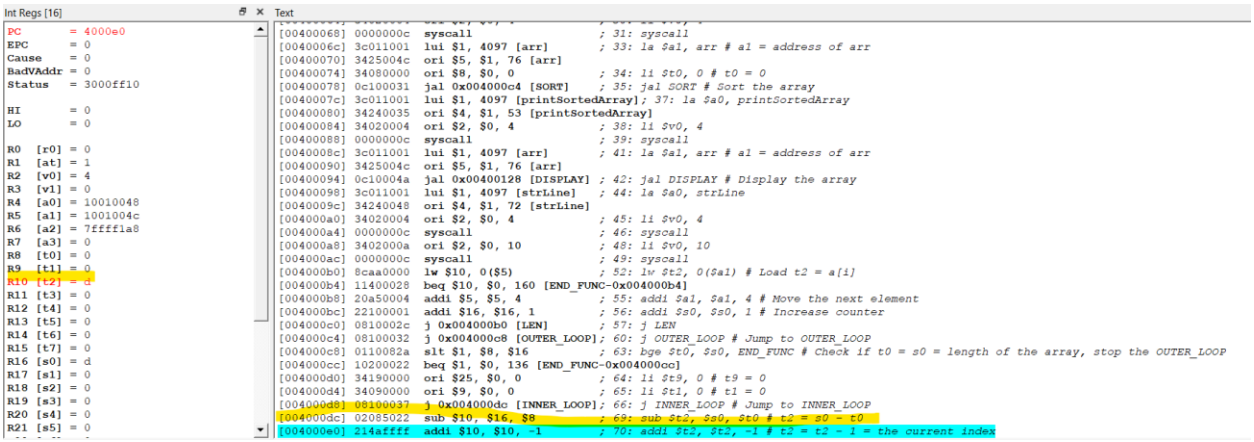
Single Step:

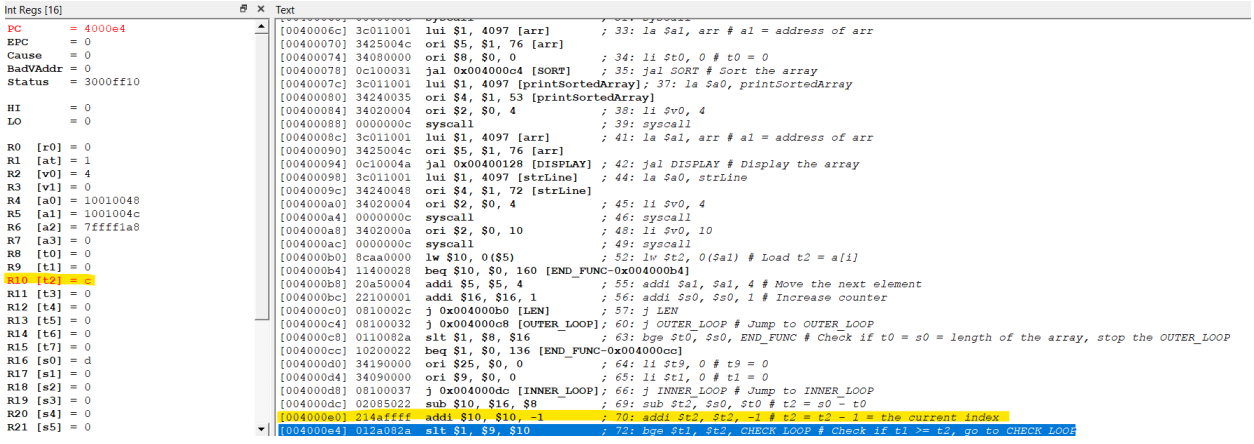
OUTER_LOOP:

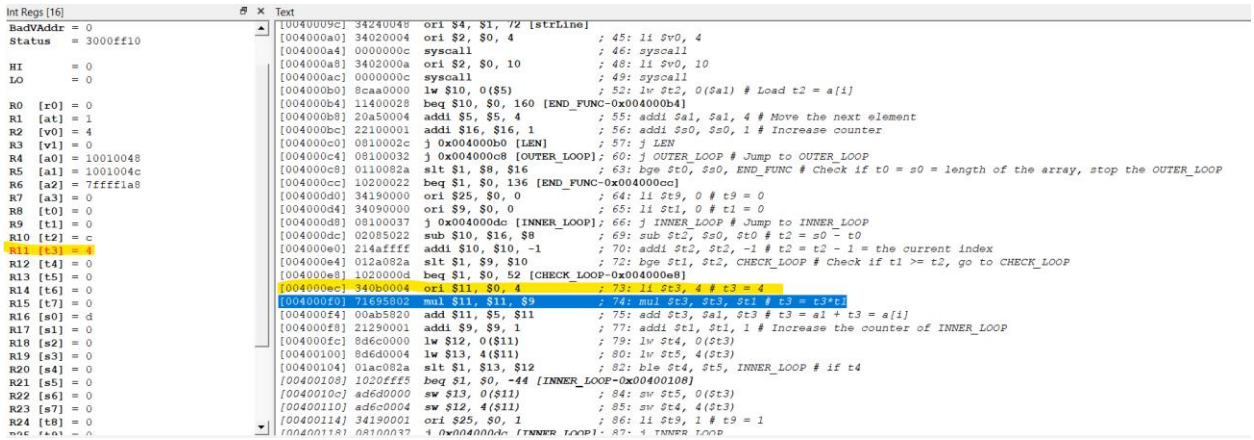


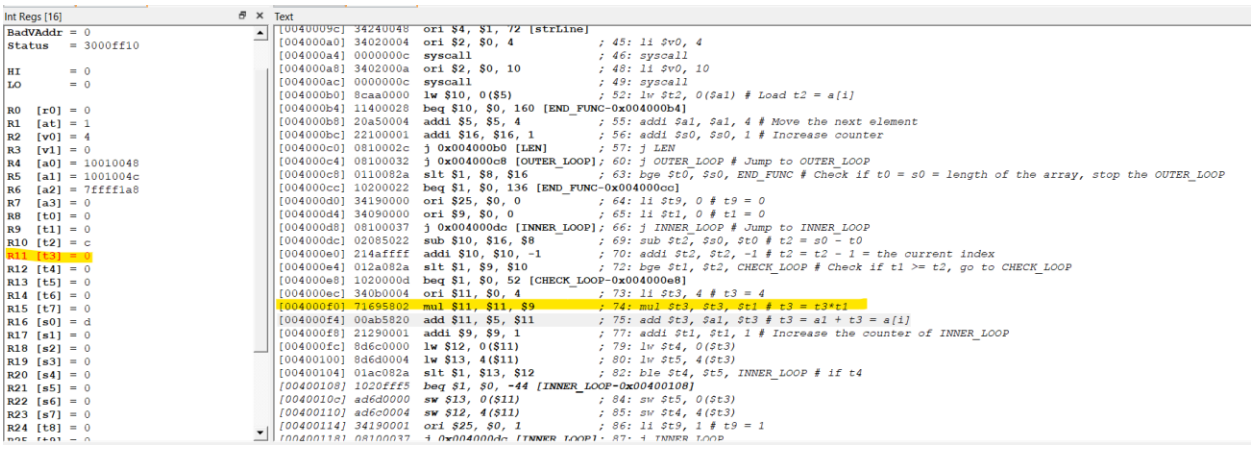
INNER_LOOP:

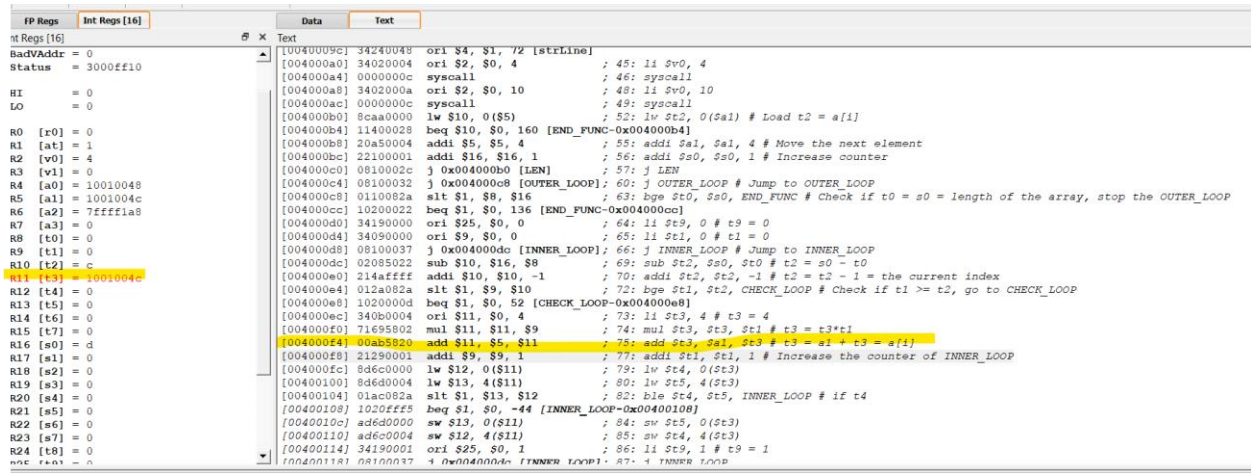
In the first loop inside INNER_LOOP:

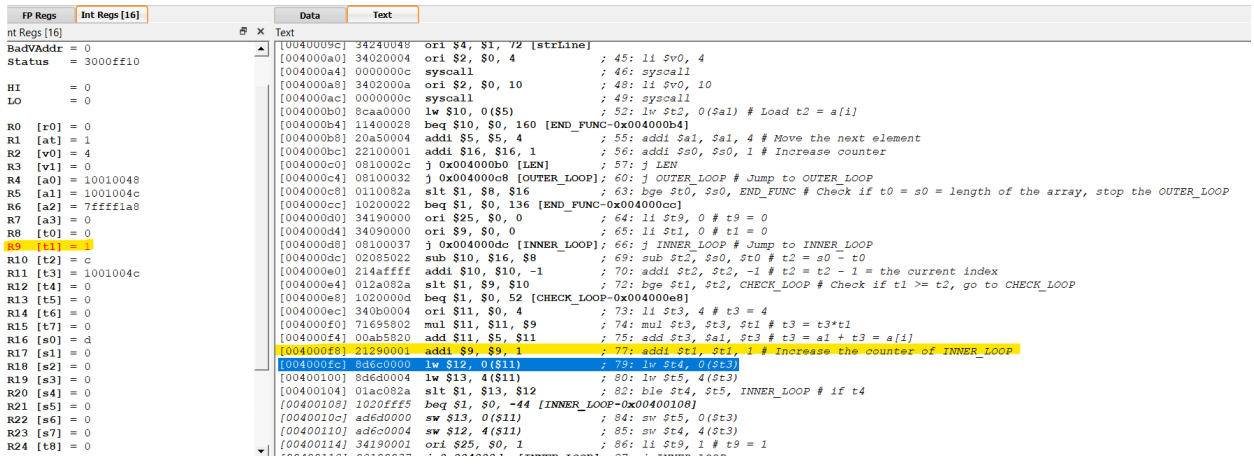


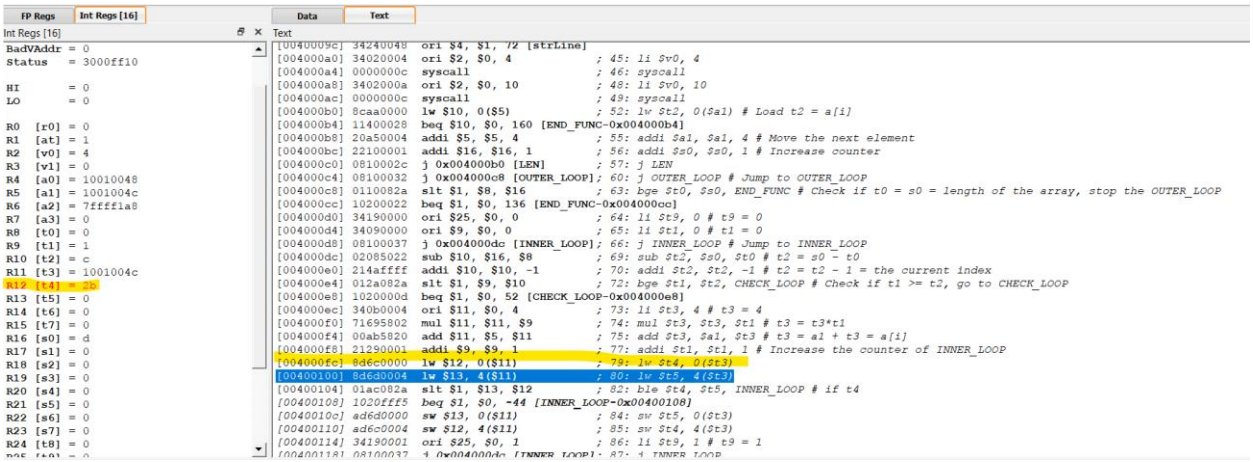


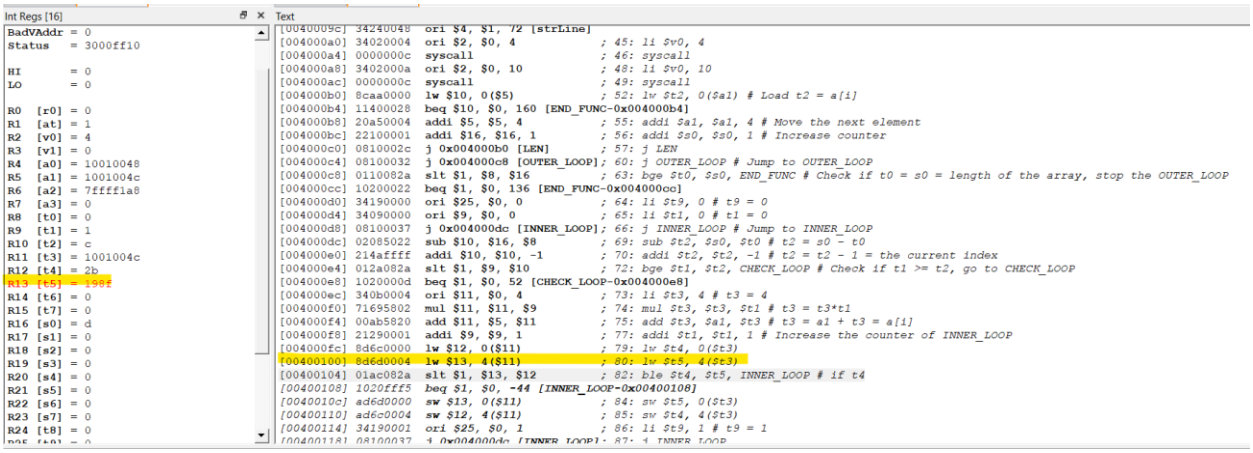


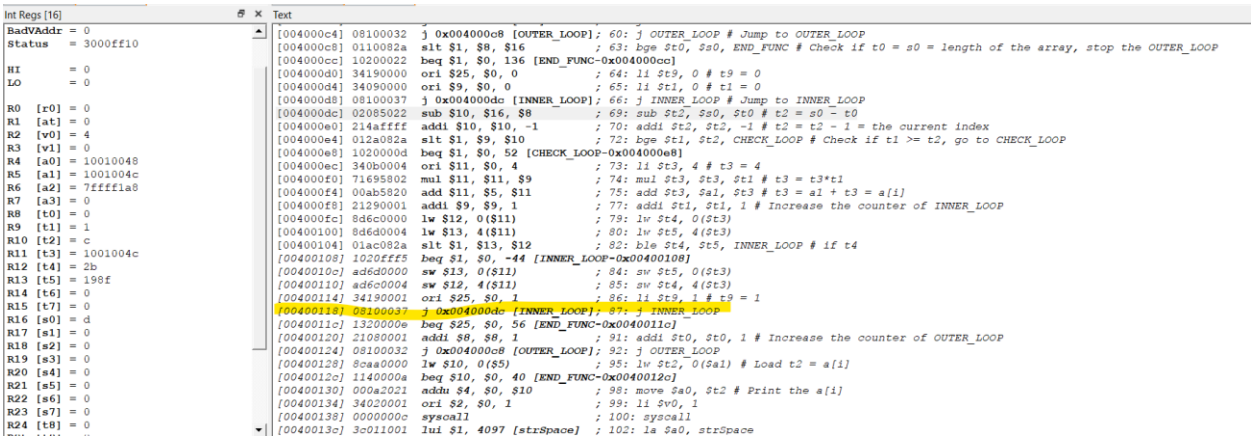

Int Regs [16]
Text
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 0
R0 [r0] = 0
R1 [at] = 1
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 10010048
R5 [a1] = 1001004c
R6 [a2] = 7ffff1a8
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = 0
R10 [t2] = c
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = d
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
[004000f0] 71695802 mul $11, $11, $9 ; 74: mul $t3, $t3, $t1 # t3 = t3*t1
[004000f4] 00ab5820 add $11, $5, $11 ; 75: add $t3, $a1, $t3 # t3 = a1 + t3 = a[i]
[004000f8] 21290001 addi $9, $9, 1 ; 77: addi $t1, $t1, 1 # Increase the counter of INNER_LOOP
[004000fc] 8d6c0000 lw $12, 0($11) ; 79: lw $t4, 0($t3)
[00400100] 8d6d0004 lw $13, 4($11) ; 80: lw $t5, 4($t3)
[00400104] 01ac082a slt $1, $13, $12 ; 82: ble $t4, $t5, INNER_LOOP # if t4
[00400108] 1020fff5 beq $1, $0, -44 [INNER_LOOP-0x00400108]
[0040010c] ad6d0000 sw $13, 0($11) ; 84: sw $t5, 0($t3)
[00400110] ad6c0004 sw $12, 4($11) ; 85: sw $t4, 4($t3)
[00400114] 34190001 ori $25, $0, 1 ; 86: li $t9, 1 # t9 = 1


FP Regs
Int Regs [16]
Data
Text
nt Regs [16]
Text
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 0
R0 [r0] = 0
R1 [at] = 1
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 10010048
R5 [a1] = 1001004c
R6 [a2] = 7ffff1a8
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = 0
R10 [t2] = c
R11 [t3] = 1001004c
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = d
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
[004000f0] 71695802 mul $11, $11, $9 ; 74: mul $t3, $t3, $t1 # t3 = t3*t1
[004000f4] 00ab5820 add $11, $5, $11 ; 75: add $t3, $a1, $t3 # t3 = a1 + t3 = a[i]
[004000f8] 21290001 addi $9, $9, 1 ; 77: addi $t1, $t1, 1 # Increase the counter of INNER_LOOP
[004000fc] 8d6c0000 lw $12, 0($11) ; 79: lw $t4, 0($t3)
[00400100] 8d6d0004 lw $13, 4($11) ; 80: lw $t5, 4($t3)
[00400104] 01ac082a slt $1, $13, $12 ; 82: ble $t4, $t5, INNER_LOOP # if t4
[00400108] 1020fff5 beq $1, $0, -44 [INNER_LOOP-0x00400108]
[0040010c] ad6d0000 sw $13, 0($11) ; 84: sw $t5, 0($t3)
[00400110] ad6c0004 sw $12, 4($11) ; 85: sw $t4, 4($t3)
[00400114] 34190001 ori $25, $0, 1 ; 86: li $t9, 1 # t9 = 1


FP Regs
Int Regs [16]
Data
Text
nt Regs [16]
Text
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 0
R0 [r0] = 0
R1 [at] = 1
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 10010048
R5 [a1] = 1001004c
R6 [a2] = 7ffff1a8
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = 1
R10 [t2] = c
R11 [t3] = 1001004c
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = d
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
[004000f0] 71695802 mul $11, $11, $9 ; 74: mul $t3, $t3, $t1 # t3 = t3*t1
[004000f4] 00ab5820 add $11, $5, $11 ; 75: add $t3, $a1, $t3 # t3 = a1 + t3 = a[i]
[004000f8] 21290001 addi $9, $9, 1 ; 77: addi $t1, $t1, 1 # Increase the counter of INNER_LOOP
[004000fc] 8d6c0000 lw $12, 0($11) ; 79: lw $t4, 0($t3)
[00400100] 8d6d0004 lw $13, 4($11) ; 80: lw $t5, 4($t3)
[00400104] 01ac082a slt $1, $13, $12 ; 82: ble $t4, $t5, INNER_LOOP # if t4
[00400108] 1020fff5 beq $1, $0, -44 [INNER_LOOP-0x00400108]
[0040010c] ad6d0000 sw $13, 0($11) ; 84: sw $t5, 0($t3)
[00400110] ad6c0004 sw $12, 4($11) ; 85: sw $t4, 4($t3)
[00400114] 34190001 ori $25, $0, 1 ; 86: li $t9, 1 # t9 = 1

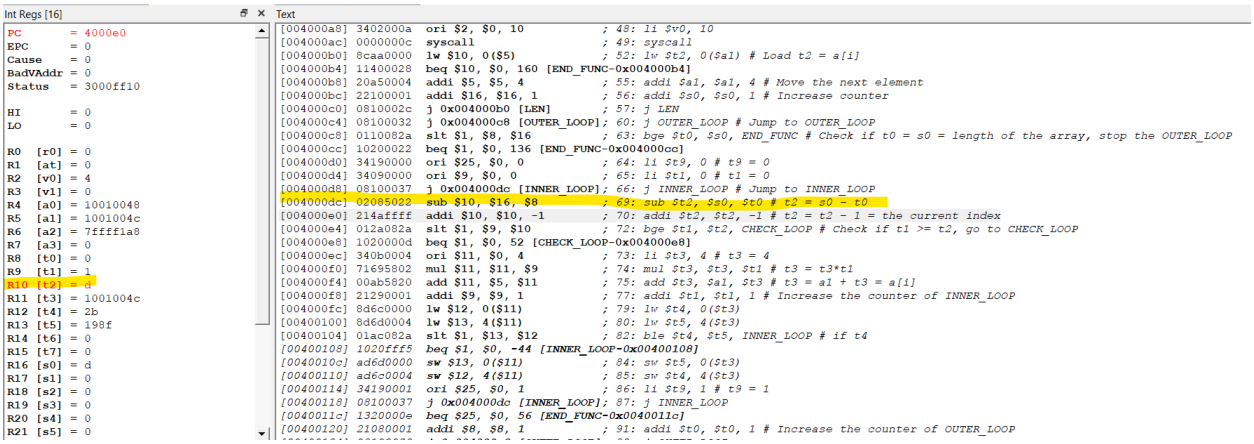


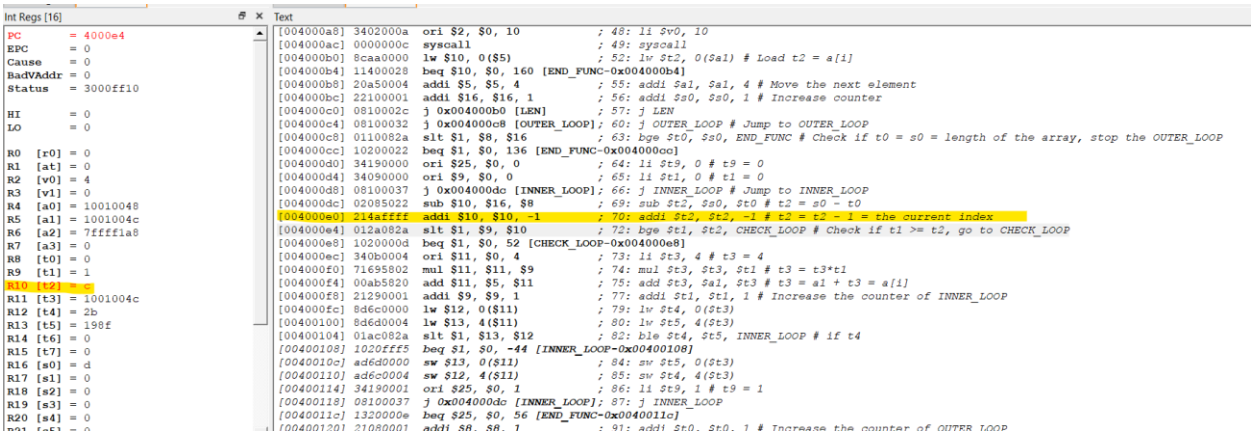


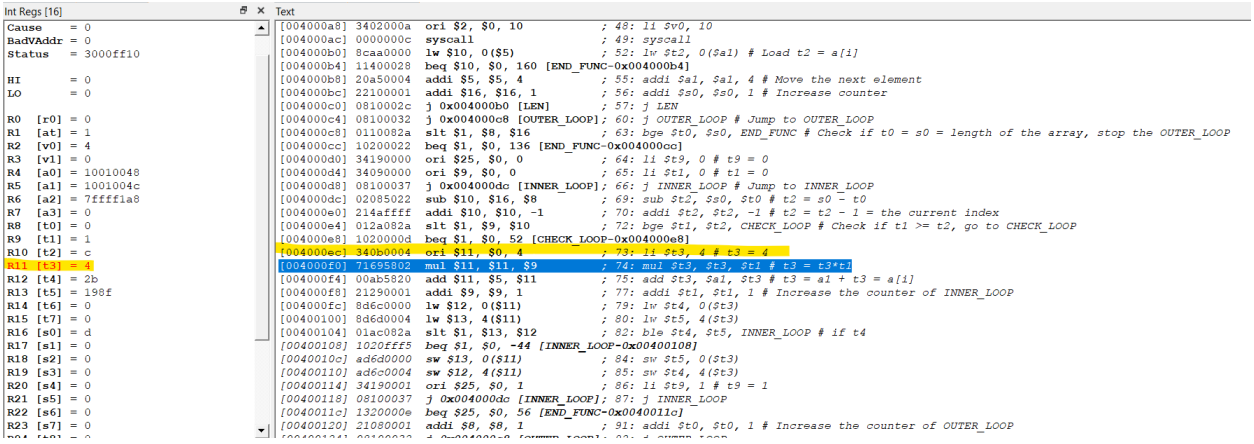


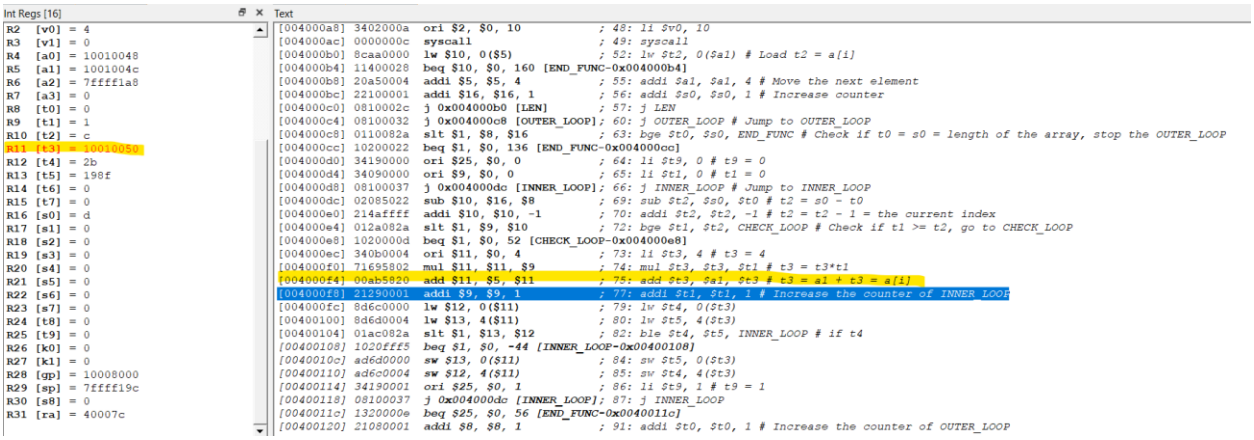
Those above steps happened when It did not sastified the condition for swapping.

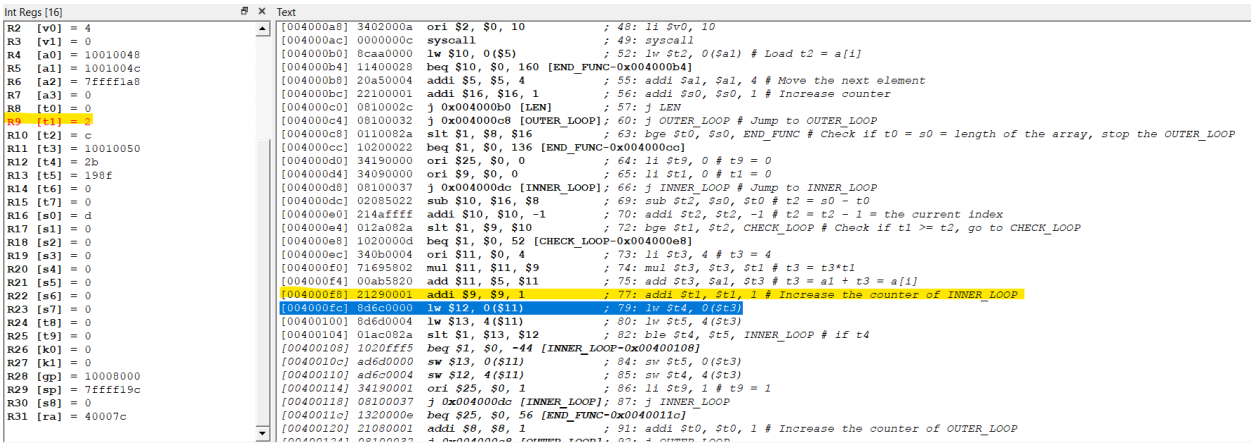
In the second loop inside INNER_LOOP:

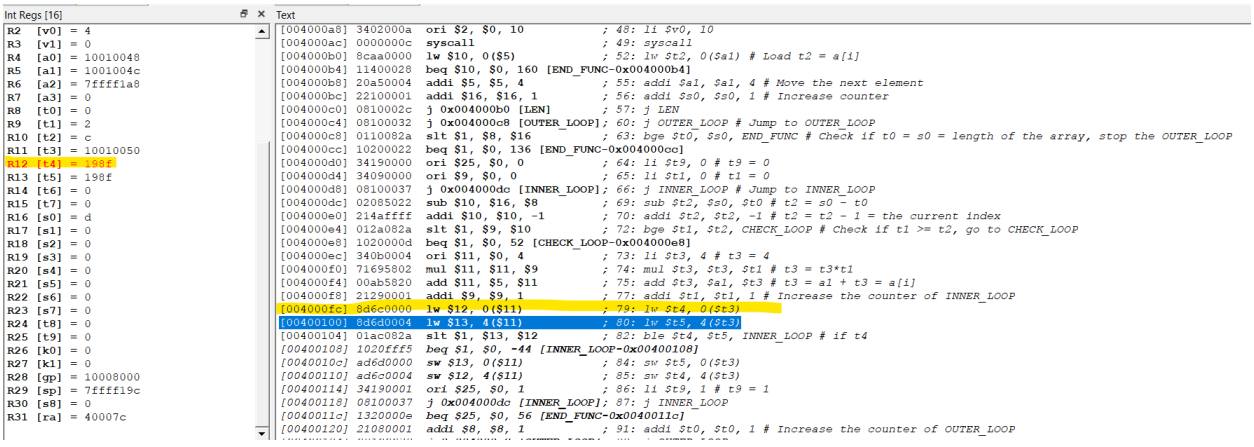


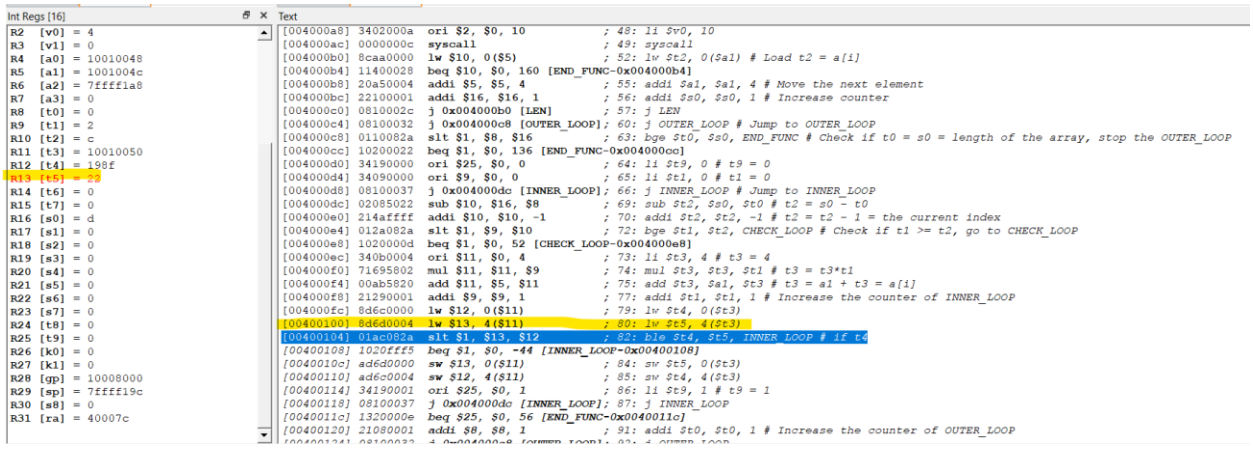


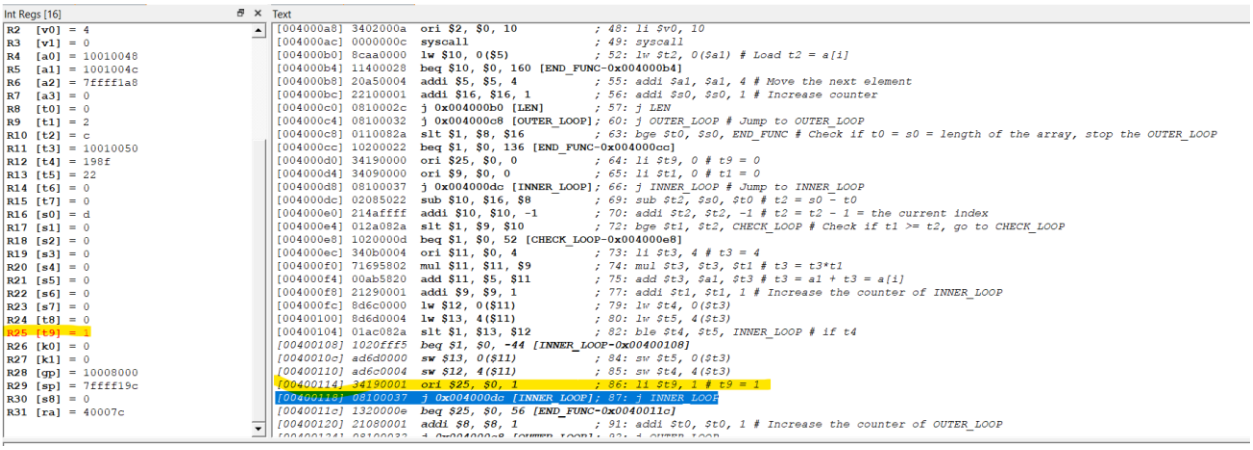



Int Regs [16]
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 10010048
R5 [a1] = 1001004c
R6 [a2] = 7ffff1a8
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = 1
R10 [t2] = c
R11 [t3] = 10010050
R12 [t4] = 2b
R13 [t5] = 198f
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = d
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff19c
R30 [s8] = 0
R31 [ra] = 40007c
Text
[004000a8] 3402000a ori $2, $0, 10 ; 48: li $v0, 10
[004000ac] 0000000c syscall ; 49: syscall
[004000b0] 8caa0000 lw $10, 0($5) ; 52: lw $t2, 0($a1) # Load t2 = a[i]
[004000b4] 11400028 beq $10, $0, 160 [END_FUNC-0x004000b4]
[004000b8] 20a50004 addi $5, $5, 4 ; 55: addi $a1, $a1, 4 # Move the next element
[004000bc] 22100001 addi $16, $16, 1 ; 56: addi $s0, $s0, 1 # Increase counter
[004000c0] 0810002c j 0x004000b0 [LEN] ; 57: j LEN
[004000c4] 08100032 j 0x004000c8 [OUTER_LOOP] ; 60: j OUTER_LOOP # Jump to OUTER_LOOP
[004000c8] 0110082a slt $1, $8, $16 ; 63: bge $t0, $s0, END_FUNC # Check if t0 = s0 = length of the array, stop the OUTER_LOOP
[004000cc] 10200022 beq $1, $0, 136 [END_FUNC-0x004000cc]
[004000d0] 34190000 ori $25, $0, 0 ; 64: li $t9, 0 # t9 = 0
[004000d4] 34090000 ori $9, $0, 0 ; 65: li $t1, 0 # t1 = 0
[004000d8] 08100037 j 0x004000dc [INNER_LOOP] ; 66: j INNER_LOOP # Jump to INNER_LOOP
[004000dc] 02085022 sub $10, $16, $8 ; 69: sub $t2, $s0, $t0 # t2 = s0 - t0
[004000e0] 214affff addi $10, $10, -1 ; 70: addi $t2, $t2, -1 # t2 = t2 - 1 = the current index
[004000e4] 012a082a slt $1, $9, $10 ; 72: bge $t1, $t2, CHECK_LOOP # Check if t1 >= t2, go to CHECK_LOOP
[004000e8] 1020000d beq $1, $0, 52 [CHECK_LOOP-0x004000e8]
[004000ec] 340b0004 ori $11, $0, 4 ; 73: li $t3, 4 # t3 = 4
[004000f0] 71695802 mul $11, $11, $9 ; 74: mul $t3, $t3, $t1 # t3 = t3*t1
[004000f4] 00ab5820 add $11, $5, $11 ; 75: add $t3, $a1, $t3 # t3 = a1 + t3 = a[i]
[004000f8] 21290001 addi $9, $9, 1 ; 77: addi $t1, $t1, 1 # Increase the counter of INNER_LOOP
[004000fc] 8d6c0000 lw $12, 0($11) ; 79: lw $t4, 0($t3)
[00400100] 8d6d0004 lw $13, 4($11) ; 80: lw $t5, 4($t3)
[00400104] 01ac082a slt $1, $13, $12 ; 82: ble $t4, $t5, INNER_LOOP # if t4
[00400108] 1020fff5 beq $1, $0, -44 [INNER_LOOP-0x00400108]
[0040010c] ad6d0000 sw $13, 0($11) ; 84: sw $t5, 0($t3)
[00400110] ad6c0004 sw $12, 4($11) ; 85: sw $t4, 4($t3)
[00400114] 34190001 ori $25, $0, 1 ; 86: li $t9, 1 # t9 = 1
[00400118] 08100037 j 0x004000dc [INNER_LOOP] ; 87: j INNER_LOOP
[0040011c] 1320000e beq $25, $0, 56 [END_FUNC-0x0040011c]
[00400120] 21080001 addi $8, $8, 1 ; 91: addi $t0, $t0, 1 # Increase the counter of OUTER_LOOP


Int Regs [16]
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 10010048
R5 [a1] = 1001004c
R6 [a2] = 7ffff1a8
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = 2
R10 [t2] = c
R11 [t3] = 10010050
R12 [t4] = 2b
R13 [t5] = 198f
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = d
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff19c
R30 [s8] = 0
R31 [ra] = 40007c
Text
[004000a8] 3402000a ori $2, $0, 10 ; 48: li $v0, 10
[004000ac] 0000000c syscall ; 49: syscall
[004000b0] 8caa0000 lw $10, 0($5) ; 52: lw $t2, 0($a1) # Load t2 = a[i]
[004000b4] 11400028 beq $10, $0, 160 [END_FUNC-0x004000b4]
[004000b8] 20a50004 addi $5, $5, 4 ; 55: addi $a1, $a1, 4 # Move the next element
[004000bc] 22100001 addi $16, $16, 1 ; 56: addi $s0, $s0, 1 # Increase counter
[004000c0] 0810002c j 0x004000b0 [LEN] ; 57: j LEN
[004000c4] 08100032 j 0x004000c8 [OUTER_LOOP] ; 60: j OUTER_LOOP # Jump to OUTER_LOOP
[004000c8] 0110082a slt $1, $8, $16 ; 63: bge $t0, $s0, END_FUNC # Check if t0 = s0 = length of the array, stop the OUTER_LOOP
[004000cc] 10200022 beq $1, $0, 136 [END_FUNC-0x004000cc]
[004000d0] 34190000 ori $25, $0, 0 ; 64: li $t9, 0 # t9 = 0
[004000d4] 34090000 ori $9, $0, 0 ; 65: li $t1, 0 # t1 = 0
[004000d8] 08100037 j 0x004000dc [INNER_LOOP] ; 66: j INNER_LOOP # Jump to INNER_LOOP
[004000dc] 02085022 sub $10, $16, $8 ; 69: sub $t2, $s0, $t0 # t2 = s0 - t0
[004000e0] 214affff addi $10, $10, -1 ; 70: addi $t2, $t2, -1 # t2 = t2 - 1 = the current index
[004000e4] 012a082a slt $1, $9, $10 ; 72: bge $t1, $t2, CHECK_LOOP # Check if t1 >= t2, go to CHECK_LOOP
[004000e8] 1020000d beq $1, $0, 52 [CHECK_LOOP-0x004000e8]
[004000ec] 340b0004 ori $11, $0, 4 ; 73: li $t3, 4 # t3 = 4
[004000f0] 71695802 mul $11, $11, $9 ; 74: mul $t3, $t3, $t1 # t3 = t3*t1
[004000f4] 00ab5820 add $11, $5, $11 ; 75: add $t3, $a1, $t3 # t3 = a1 + t3 = a[i]
[004000f8] 21290001 addi $9, $9, 1 ; 77: addi $t1, $t1, 1 # Increase the counter of INNER_LOOP
[004000fc] 8d6c0000 lw $12, 0($11) ; 79: lw $t4, 0($t3)
[00400100] 8d6d0004 lw $13, 4($11) ; 80: lw $t5, 4($t3)
[00400104] 01ac082a slt $1, $13, $12 ; 82: ble $t4, $t5, INNER_LOOP # if t4
[00400108] 1020fff5 beq $1, $0, -44 [INNER_LOOP-0x00400108]
[0040010c] ad6d0000 sw $13, 0($11) ; 84: sw $t5, 0($t3)
[00400110] ad6c0004 sw $12, 4($11) ; 85: sw $t4, 4($t3)
[00400114] 34190001 ori $25, $0, 1 ; 86: li $t9, 1 # t9 = 1
[00400118] 08100037 j 0x004000dc [INNER_LOOP] ; 87: j INNER_LOOP
[0040011c] 1320000e beq $25, $0, 56 [END_FUNC-0x0040011c]
[00400120] 21080001 addi $8, $8, 1 ; 91: addi $t0, $t0, 1 # Increase the counter of OUTER_LOOP

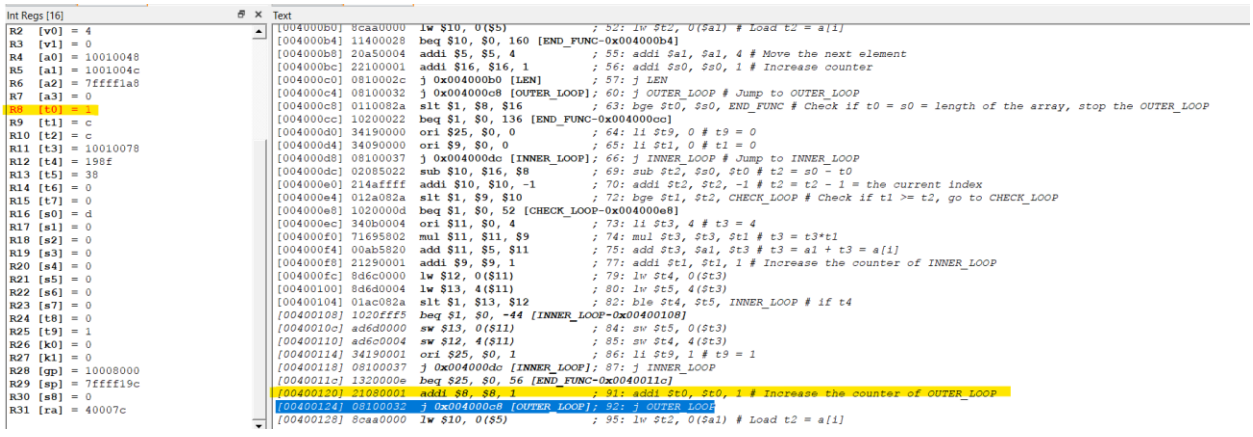

Int Regs [16]
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 10010048
R5 [a1] = 1001004c
R6 [a2] = 7ffff1a8
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = 2
R10 [t2] = c
R11 [t3] = 10010050
R12 [t4] = 198f
R13 [t5] = 198f
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = d
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff19c
R30 [s8] = 0
R31 [ra] = 40007c
Text
[004000a8] 3402000a ori $2, $0, 10 ; 48: li $v0, 10
[004000ac] 0000000c syscall ; 49: syscall
[004000b0] 8caa0000 lw $10, 0($5) ; 52: lw $t2, 0($a1) # Load t2 = a[i]
[004000b4] 11400028 beq $10, $0, 160 [END_FUNC-0x004000b4]
[004000b8] 20a50004 addi $5, $5, 4 ; 55: addi $a1, $a1, 4 # Move the next element
[004000bc] 22100001 addi $16, $16, 1 ; 56: addi $s0, $s0, 1 # Increase counter
[004000c0] 0810002c j 0x004000b0 [LEN] ; 57: j LEN
[004000c4] 08100032 j 0x004000c8 [OUTER_LOOP] ; 60: j OUTER_LOOP # Jump to OUTER_LOOP
[004000c8] 0110082a slt $1, $8, $16 ; 63: bge $t0, $s0, END_FUNC # Check if t0 = s0 = length of the array, stop the OUTER_LOOP
[004000cc] 10200022 beq $1, $0, 136 [END_FUNC-0x004000cc]
[004000d0] 34190000 ori $25, $0, 0 ; 64: li $t9, 0 # t9 = 0
[004000d4] 34090000 ori $9, $0, 0 ; 65: li $t1, 0 # t1 = 0
[004000d8] 08100037 j 0x004000dc [INNER_LOOP] ; 66: j INNER_LOOP # Jump to INNER_LOOP
[004000dc] 02085022 sub $10, $16, $8 ; 69: sub $t2, $s0, $t0 # t2 = s0 - t0
[004000e0] 214affff addi $10, $10, -1 ; 70: addi $t2, $t2, -1 # t2 = t2 - 1 = the current index
[004000e4] 012a082a slt $1, $9, $10 ; 72: bge $t1, $t2, CHECK_LOOP # Check if t1 >= t2, go to CHECK_LOOP
[004000e8] 1020000d beq $1, $0, 52 [CHECK_LOOP-0x004000e8]
[004000ec] 340b0004 ori $11, $0, 4 ; 73: li $t3, 4 # t3 = 4
[004000f0] 71695802 mul $11, $11, $9 ; 74: mul $t3, $t3, $t1 # t3 = t3*t1
[004000f4] 00ab5820 add $11, $5, $11 ; 75: add $t3, $a1, $t3 # t3 = a1 + t3 = a[i]
[004000f8] 21290001 addi $9, $9, 1 ; 77: addi $t1, $t1, 1 # Increase the counter of INNER_LOOP
[004000fc] 8d6c0000 lw $12, 0($11) ; 79: lw $t4, 0($t3)
[00400100] 8d6d0004 lw $13, 4($11) ; 80: lw $t5, 4($t3)
[00400104] 01ac082a slt $1, $13, $12 ; 82: ble $t4, $t5, INNER_LOOP # if t4
[00400108] 1020fff5 beq $1, $0, -44 [INNER_LOOP-0x00400108]
[0040010c] ad6d0000 sw $13, 0($11) ; 84: sw $t5, 0($t3)
[00400110] ad6c0004 sw $12, 4($11) ; 85: sw $t4, 4($t3)
[00400114] 34190001 ori $25, $0, 1 ; 86: li $t9, 1 # t9 = 1
[00400118] 08100037 j 0x004000dc [INNER_LOOP] ; 87: j INNER_LOOP
[0040011c] 1320000e beq $25, $0, 56 [END_FUNC-0x0040011c]
[00400120] 21080001 addi $8, $8, 1 ; 91: addi $t0, $t0, 1 # Increase the counter of OUTER_LOOP





Those above steps happened when It sastified the condition for swapping.

It will repeat the step until $t1 >= $t2.



When $t1 >= $t2, Increasing the counter $t0 = $t0 + 1. Then, It will jump back to OUTER_LOOP and reapeat those step until $t0 = $t0

Code(Quick_Sort):

```
.data
strNameId: .asciiz "Phan Tran Thanh Huy\nITCSIU22056\n"
printInitialArray: .asciiz "The initial array: "
printSortedArray: .asciiz "The sorted array: "
strLine: .asciiz "\n"
strSpace: .asciiz " "
arr: .word 43, 6441, 34, 54, 4232, 64, 526, 643, 6475, 423, 4236, 566, 56, 0
sortedArr: .space 1024

.text
.globl main

main:
    la $a0, strNameId
    li $v0, 4
    syscall

    la $a0, printInitialArray
    li $v0, 4
    syscall

    la $a1, arr          # $a1 = address of arr
    jal DISPLAY          # Print the array

    la $a0, strLine
    li $v0, 4
    syscall

    la $a0, arr          # $a0 = address of arr
    li $a1, 0            # Low index of the array which is 0
    li $a2, 11           # High index of the array which is 11
    jal QUICKSORT            # Sort the array by QuickSort

    la $a0, printSortedArray
    li $v0, 4
    syscall

    la $a1, arr          # $a1 = address of arr
    jal DISPLAY          # Print the array

    la $a0, strLine
    li $v0, 4
    syscall

    li $v0, 10
    syscall

SWAP:
    addi $sp, $sp, -16  # Allocate space on the stack

    sw $a0, 0($sp)      # Store array
    sw $a1, 4($sp)      # Store low
    sw $a2, 8($sp)      # Store High
    sw $t2, 12($sp)     # Store t2 for high

    sll $t1, $a1, 2     # $t1 = low * 4
    add $t1, $a0, $t1   # $t1 = arr + Low * 4
    lw $s3, 0($t1)      # $s3 = arr[$t1]

    sll $t2, $a2, 2     # $t2 = High * 4
    add $t2, $a0, $t2   # $t2 = arr + High * 4
    lw $s4, 0($t2)      # $s4 = arr[$t2]

    sw $s4, 0($t1)      # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
    sw $s3, 0($t2)      # arr[$t2] = $s3

    lw $t2, 12($sp)     # Load $t2 on the stack
    addi $sp, $sp, 16
    jr $ra
PARTITION:
    addi $sp, $sp, -16      # Allocate space on the stack

    sw $a0, 0($sp)          # Save array on the stack
    sw $a1, 4($sp)          # Save low on the stack
    sw $a2, 8($sp)          # Save high on the stack
    sw $ra, 12($sp)         # Save $ra on the stack

    move $s1, $a1       # $s1 = $a1
    move $s2, $a2       # $s2 = $a2

    sll $t1, $s2, 2     # t1 = high * 4
    add $t1, $a0, $t1   # t1 = arr + high * 4
    lw $t2, 0($t1)          # pivot = t2 = arr[high]

    addi $t3, $s1, -1   # t3 = i = low - 1
    move $t4, $s1       # t4 = j = low
    addi $t5, $s2, -1   # t5 = high - 1

    forloop:
        slt $t6, $t5, $t4   # Check $t6 = 1 if j > high - 1
                  #       $t6 = 0 if j <= high - 1
            bne $t6, $0, endfor # Check if j > high, go to endfor

            sll $t1, $t4, 2     # $t1 = low * 4
            add $t1, $t1, $a0   # $t1 = arr + low * 4
            lw $t7, 0($t1)      # $t7 = arr[j] = arr[low]

            slt $t8, $t7, $t2   # Check $t8 = 1 if arr[j] > pivot
                  #   $t8 = 0 if arr[j] < pivot
            bne $t8, $zero, endfif  # if array[j] <= pivot -> endIf

            addi $t3, $t3, 1    # Increase i++
            move $a1, $t3
            move $a2, $t4
            jal SWAP

        addi $t4, $t4, 1    # j++
        j forloop

    endfif:
            addi $t4, $t4, 1    # j++
            j forloop

    endfor:
            addi $a1, $t3, 1    # $a1 = i + 1
            move $a2, $s2       # $a2 = high
        add $v0, $zero, $a1 # $v0 = $a1
            jal SWAP                # Swap(arr[i+1], arr[high])

            lw $ra, 12($sp)     # Return address
            addi $sp, $sp, 16   # Restore the stack
            jr $ra

QUICKSORT:
        addi $sp, $sp, -16  # Allocate space on the stack

        sw $a0, 0($sp)      # save arr
        sw $a1, 4($sp)      # save low
        sw $a2, 8($sp)      # save high
        sw $ra, 12($sp)     # Return address

        move $t0, $a2       # $t0 = high

        slt $t1, $a1, $t0   # Check $t1 = 1 if low > high
                  #   $t1 = 0 if low <= high

        beq $t1, $zero, ENDIF   # if low <= high, go to ENDIF

        jal PARTITION

        move $s0, $v0       # pivot $s0 = $v0

            lw $a1, 4($sp)      # load low
            addi $a2, $s0, -1   # pivot - 1
            jal QUICKSORT       # quicksort(array, low, pivot - 1)

            addi $a1, $s0, 1    # pivot + 1
            lw $a2, 8($sp)      # load high
            jal QUICKSORT       # quicksort(array, pivot + 1, high)

ENDIF:

            lw $a0, 0($sp)      # Restore a0
            lw $a1, 4($sp)      # Restore a1
            lw $a2, 8($sp)      # Restore a2
            lw $ra, 12($sp)         # Restore return address
            addi $sp, $sp, 16       # Restore the stack
            jr $ra          # Return the caller

DISPLAY:
    lw $t2, 0($a1)          # Load $t2 = a[i]
    beq $t2, $0, END_FUNC       # Check if $t2 = 0, go to the END_FUNC

    move $a0, $t2           # Print $t2 = a[i]
    li $v0, 1
    syscall

    la $a0, strSpace
    li $v0, 4
    syscall

    addi $a1, $a1, 4        # Move to the next element
    j DISPLAY

END_FUNC:
    j $ra
```

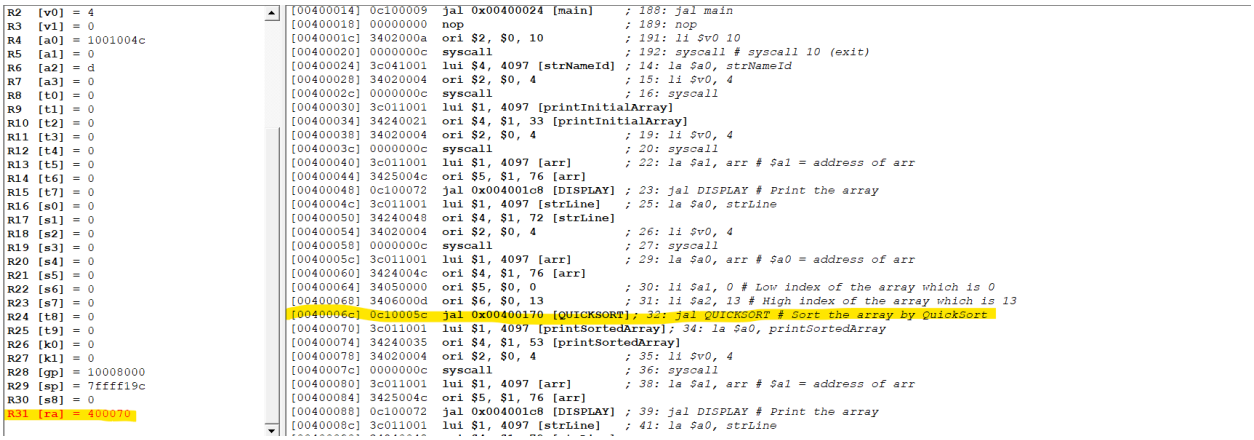
Test case

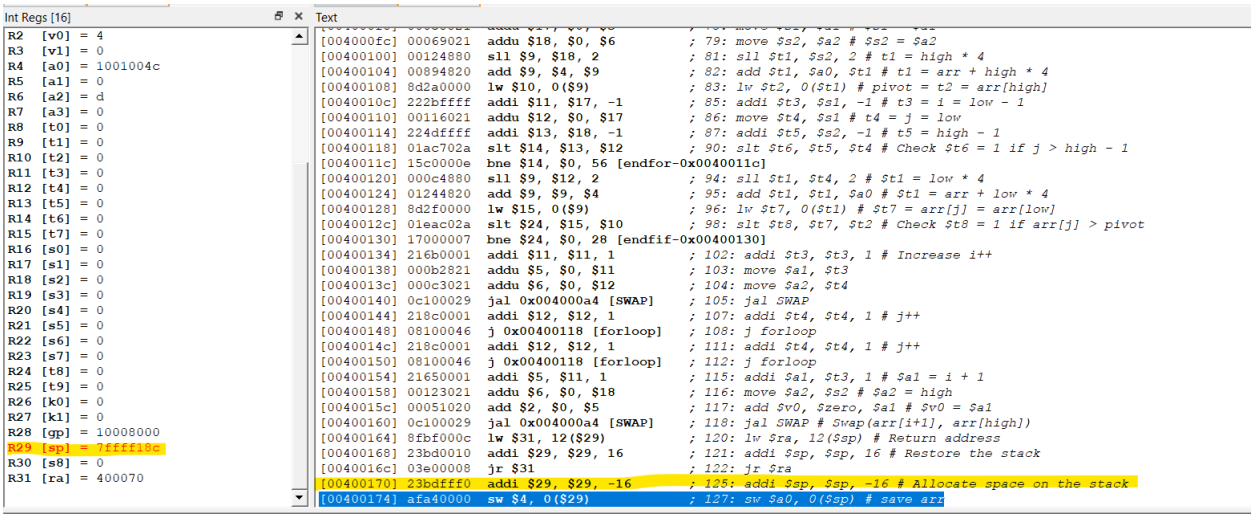
Console

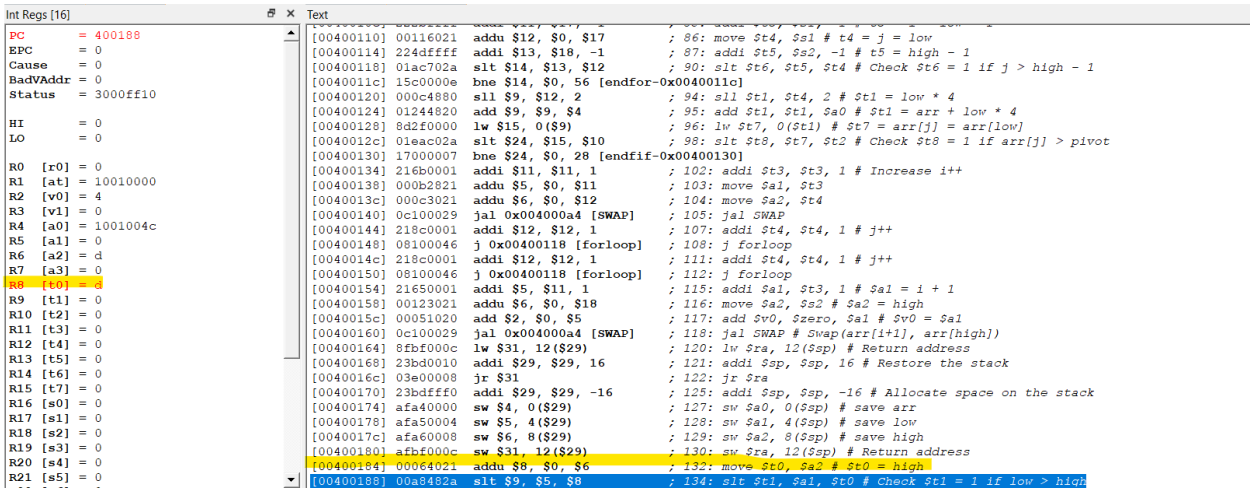
```
Phan Tran Thanh Huy
ITCSIU22056
The initial array: 43 6543 34 54 4232 64 526 643 6435 423 4236 566 56
The sorted array: 6543 6435 4236 4232 643 566 526 423 64 56 54 43 34
```

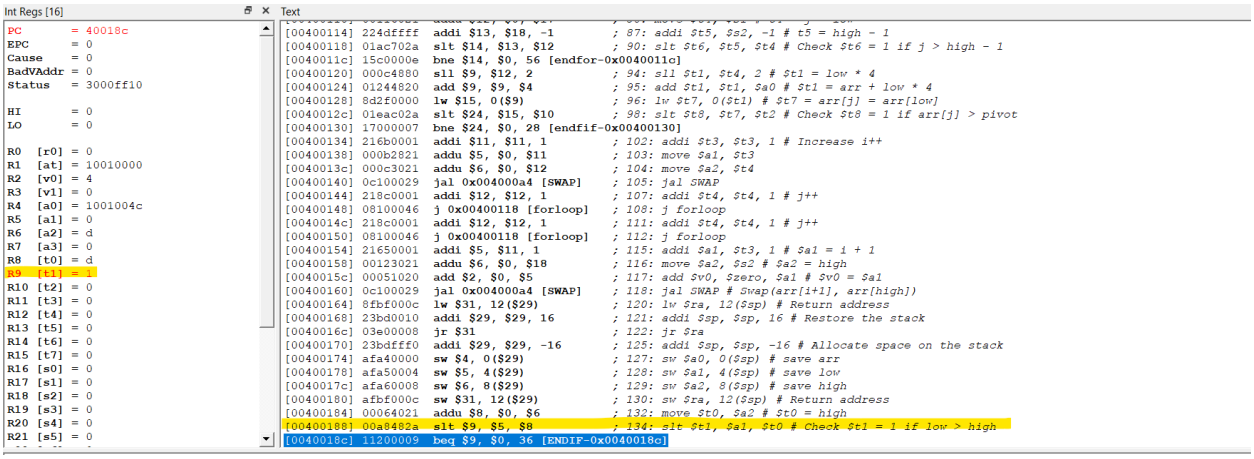
Single Step:

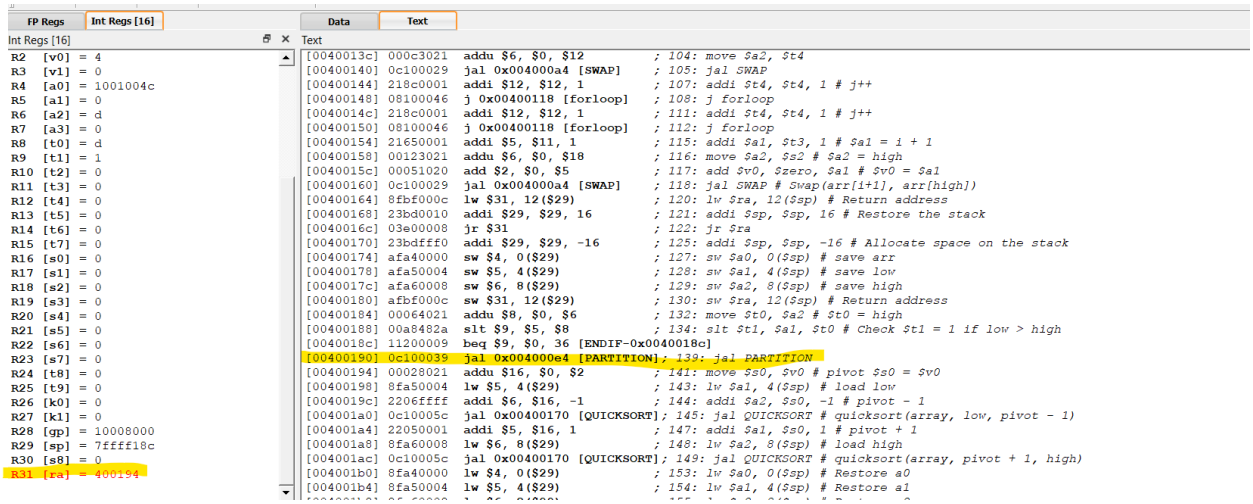
QuickSort (Array, Low, High)

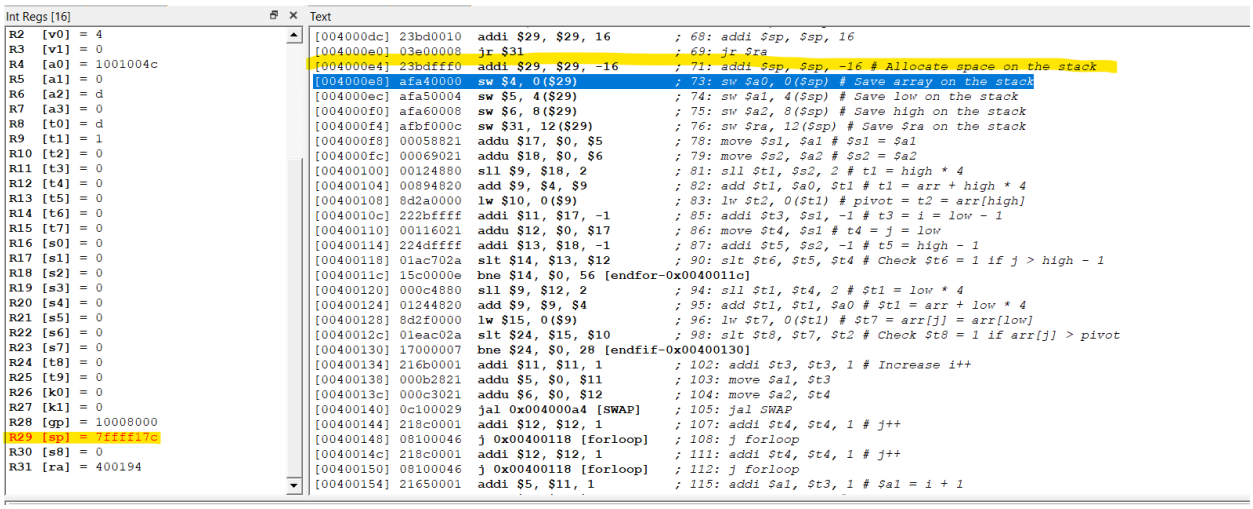


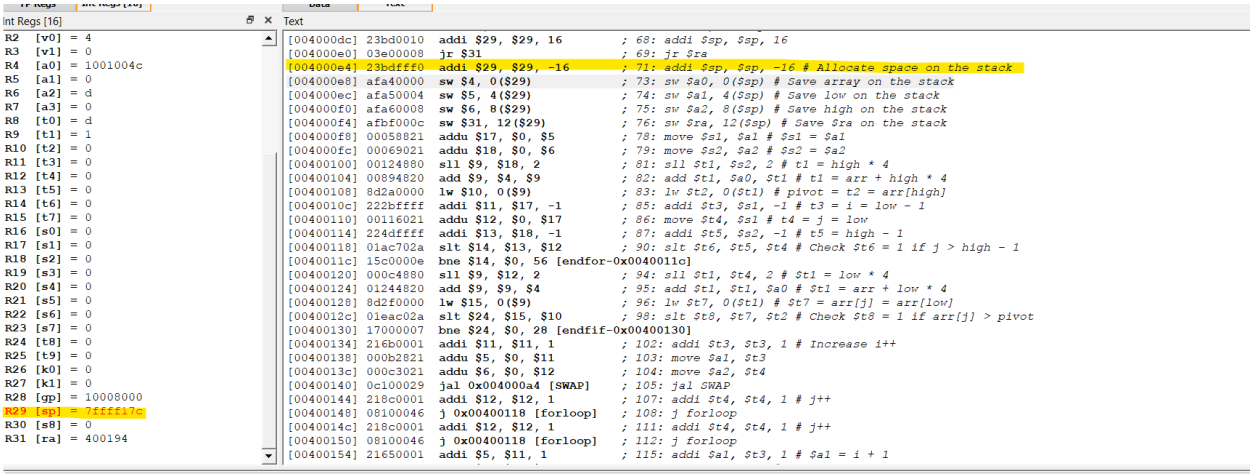


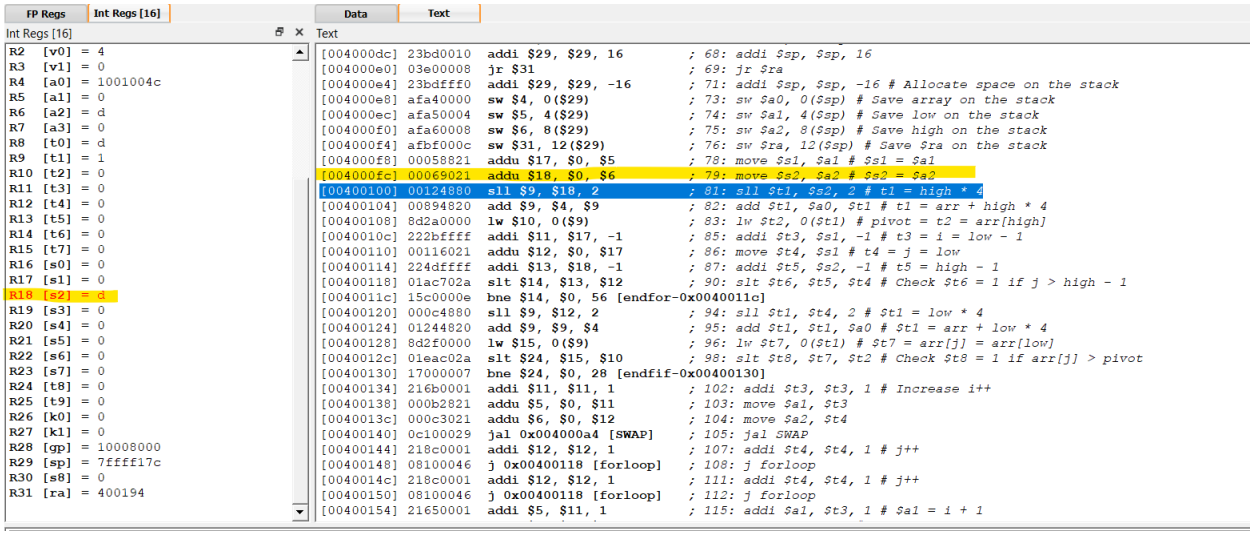

Int Regs [16]
PC = 400188
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 0
R0 [r0] = 0
R1 [at] = 10010000
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 1001004c
R5 [a1] = 0
R6 [a2] = d
R7 [a3] = 0
R8 [t0] = d
R9 [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
Text
[00400110] 00116021 addu $12, $0, $17 ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff addi $13, $18, -1 ; 87: addi $t5, $s2, -1 # t5 = high - 1
[00400118] 01ac702a slt $14, $13, $12 ; 90: slt $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
[0040011c] 15c0000e bne $14, $0, 56 [endfor-0x0040011c]
[00400120] 000c4880 sll $9, $12, 2 ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400124] 01244820 add $9, $9, $4 ; 95: add $t1, $t1, $a0 # $t1 = arr + low * 4
[00400128] 8d2f0000 lw $15, 0($9) ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
[0040012c] 01eac02a slt $24, $15, $10 ; 98: slt $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400130] 17000007 bne $24, $0, 28 [endfif-0x00400130]
[00400134] 216b0001 addi $11, $11, 1 ; 102: addi $t3, $t3, 1 # Increase i++
[00400138] 000b2821 addu $5, $0, $11 ; 103: move $a1, $t3
[0040013c] 000c3021 addu $6, $0, $12 ; 104: move $a2, $t4
[00400140] 0c100029 jal 0x004000a4 [SWAP] ; 105: jal SWAP
[00400144] 218c0001 addi $12, $12, 1 ; 107: addi $t4, $t4, 1 # j++
[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
[0040014c] 218c0001 addi $12, $12, 1 ; 111: addi $t4, $t4, 1 # j++
[00400150] 08100046 j 0x00400118 [forloop] ; 112: j forloop
[00400154] 21650001 addi $5, $11, 1 ; 115: addi $a1, $t3, 1 # $a1 = i + 1
[00400158] 00123021 addu $6, $0, $18 ; 116: move $a2, $s2 # $a2 = high
[0040015c] 00051020 add $2, $0, $5 ; 117: add $v0, $zero, $a1 # $v0 = $a1
[00400160] 0c100029 jal 0x004000a4 [SWAP] ; 118: jal SWAP # Swap(arr[i+1], arr[high])
[00400164] 8fbf000c lw $31, 12($29) ; 120: lw $ra, 12($sp) # Return address
[00400168] 23bd0010 addi $29, $29, 16 ; 121: addi $sp, $sp, 16 # Restore the stack
[0040016c] 03e00008 jr $31 ; 122: jr $ra
[00400170] 23bdfff0 addi $29, $29, -16 ; 125: addi $sp, $sp, -16 # Allocate space on the stack
[00400174] afa40000 sw $4, 0($29) ; 127: sw $a0, 0($sp) # save arr
[00400178] afa50004 sw $5, 4($29) ; 128: sw $a1, 4($sp) # save low
[0040017c] afa60008 sw $6, 8($29) ; 129: sw $a2, 8($sp) # save high
[00400180] afbf000c sw $31, 12($29) ; 130: sw $ra, 12($sp) # Return address
[00400184] 00064021 addu $8, $0, $6 ; 132: move $t0, $a2 # $t0 = high
[00400188] 00a8482a slt $9, $5, $8 ; 134: slt $t1, $a1, $t0 # Check $t1 = 1 if low > high


Int Regs [16]
PC = 40018c
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 0
R0 [r0] = 0
R1 [at] = 10010000
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 1001004c
R5 [a1] = 0
R6 [a2] = d
R7 [a3] = 0
R8 [t0] = d
R9 [t1] = 1
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
Text
[00400114] 224dffff addi $13, $18, -1 ; 87: addi $t5, $s2, -1 # t5 = high - 1
[00400118] 01ac702a slt $14, $13, $12 ; 90: slt $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
[0040011c] 15c0000e bne $14, $0, 56 [endfor-0x0040011c]
[00400120] 000c4880 sll $9, $12, 2 ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400124] 01244820 add $9, $9, $4 ; 95: add $t1, $t1, $a0 # $t1 = arr + low * 4
[00400128] 8d2f0000 lw $15, 0($9) ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
[0040012c] 01eac02a slt $24, $15, $10 ; 98: slt $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400130] 17000007 bne $24, $0, 28 [endfif-0x00400130]
[00400134] 216b0001 addi $11, $11, 1 ; 102: addi $t3, $t3, 1 # Increase i++
[00400138] 000b2821 addu $5, $0, $11 ; 103: move $a1, $t3
[0040013c] 000c3021 addu $6, $0, $12 ; 104: move $a2, $t4
[00400140] 0c100029 jal 0x004000a4 [SWAP] ; 105: jal SWAP
[00400144] 218c0001 addi $12, $12, 1 ; 107: addi $t4, $t4, 1 # j++
[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
[0040014c] 218c0001 addi $12, $12, 1 ; 111: addi $t4, $t4, 1 # j++
[00400150] 08100046 j 0x00400118 [forloop] ; 112: j forloop
[00400154] 21650001 addi $5, $11, 1 ; 115: addi $a1, $t3, 1 # $a1 = i + 1
[00400158] 00123021 addu $6, $0, $18 ; 116: move $a2, $s2 # $a2 = high
[0040015c] 00051020 add $2, $0, $5 ; 117: add $v0, $zero, $a1 # $v0 = $a1
[00400160] 0c100029 jal 0x004000a4 [SWAP] ; 118: jal SWAP # Swap(arr[i+1], arr[high])
[00400164] 8fbf000c lw $31, 12($29) ; 120: lw $ra, 12($sp) # Return address
[00400168] 23bd0010 addi $29, $29, 16 ; 121: addi $sp, $sp, 16 # Restore the stack
[0040016c] 03e00008 jr $31 ; 122: jr $ra
[00400170] 23bdfff0 addi $29, $29, -16 ; 125: addi $sp, $sp, -16 # Allocate space on the stack
[00400174] afa40000 sw $4, 0($29) ; 127: sw $a0, 0($sp) # save arr
[00400178] afa50004 sw $5, 4($29) ; 128: sw $a1, 4($sp) # save low
[0040017c] afa60008 sw $6, 8($29) ; 129: sw $a2, 8($sp) # save high
[00400180] afbf000c sw $31, 12($29) ; 130: sw $ra, 12($sp) # Return address
[00400184] 00064021 addu $8, $0, $6 ; 132: move $t0, $a2 # $t0 = high
[00400188] 00a8482a slt $9, $5, $8 ; 134: slt $t1, $a1, $t0 # Check $t1 = 1 if low > high
[0040018c] 11200009 beq $9, $0, 36 [ENDIF-0x0040018c]


FP Regs
Int Regs [16]
Data
Text
Int Regs [16]
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 1001004c
R5 [a1] = 0
R6 [a2] = d
R7 [a3] = 0
R8 [t0] = d
R9 [t1] = 1
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff18c
R30 [s8] = 0
R31 [ra] = 400194
Text
[0040013c] 000c3021 addu $6, $0, $12 ; 104: move $a2, $t4
[00400140] 0c100029 jal 0x004000a4 [SWAP] ; 105: jal SWAP
[00400144] 218c0001 addi $12, $12, 1 ; 107: addi $t4, $t4, 1 # j++
[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
[0040014c] 218c0001 addi $12, $12, 1 ; 111: addi $t4, $t4, 1 # j++
[00400150] 08100046 j 0x00400118 [forloop] ; 112: j forloop
[00400154] 21650001 addi $5, $11, 1 ; 115: addi $a1, $t3, 1 # $a1 = i + 1
[00400158] 00123021 addu $6, $0, $18 ; 116: move $a2, $s2 # $a2 = high
[0040015c] 00051020 add $2, $0, $5 ; 117: add $v0, $zero, $a1 # $v0 = $a1
[00400160] 0c100029 jal 0x004000a4 [SWAP] ; 118: jal SWAP # Swap(arr[i+1], arr[high])
[00400164] 8fbf000c lw $31, 12($29) ; 120: lw $ra, 12($sp) # Return address
[00400168] 23bd0010 addi $29, $29, 16 ; 121: addi $sp, $sp, 16 # Restore the stack
[0040016c] 03e00008 jr $31 ; 122: jr $ra
[00400170] 23bdfff0 addi $29, $29, -16 ; 125: addi $sp, $sp, -16 # Allocate space on the stack
[00400174] afa40000 sw $4, 0($29) ; 127: sw $a0, 0($sp) # save arr
[00400178] afa50004 sw $5, 4($29) ; 128: sw $a1, 4($sp) # save low
[0040017c] afa60008 sw $6, 8($29) ; 129: sw $a2, 8($sp) # save high
[00400180] afbf000c sw $31, 12($29) ; 130: sw $ra, 12($sp) # Return address
[00400184] 00064021 addu $8, $0, $6 ; 132: move $t0, $a2 # $t0 = high
[00400188] 00a8482a slt $9, $5, $8 ; 134: slt $t1, $a1, $t0 # Check $t1 = 1 if low > high
[0040018c] 11200009 beq $9, $0, 36 [ENDIF-0x0040018c]
[00400190] 0c100039 jal 0x004000e4 [PARTITION] ; 139: jal PARTITION
[00400194] 00028021 addu $16, $0, $2 ; 141: move $s0, $v0 # pivot $s0 = $v0
[00400198] 8fa50004 lw $5, 4($29) ; 143: lw $a1, 4($sp) # load low
[0040019c] 2206ffff addi $6, $16, -1 ; 144: addi $a2, $s0, -1 # pivot - 1
[004001a0] 0c10005c jal 0x00400170 [QUICKSORT] ; 145: jal QUICKSORT # quicksort(array, low, pivot - 1)
[004001a4] 22050001 addi $5, $16, 1 ; 147: addi $a1, $s0, 1 # pivot + 1
[004001a8] 8fa60008 lw $6, 8($29) ; 148: lw $a2, 8($sp) # load high
[004001ac] 0c10005c jal 0x00400170 [QUICKSORT] ; 149: jal QUICKSORT # quicksort(array, pivot + 1, high)
[004001b0] 8fa40000 lw $4, 0($29) ; 153: lw $a0, 0($sp) # Restore a0
[004001b4] 8fa50004 lw $5, 4($29) ; 154: lw $a1, 4($sp) # Restore a1


Int Regs [16]
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 1001004c
R5 [a1] = 0
R6 [a2] = d
R7 [a3] = 0
R8 [t0] = d
R9 [t1] = 1
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff17c
R30 [s8] = 0
R31 [ra] = 400194
Text
[004000dc] 23bd0010 addi $29, $29, 16 ; 68: addi $sp, $sp, 16
[004000e0] 03e00008 jr $31 ; 69: jr $ra
[004000e4] 23bdfff0 addi $29, $29, -16 ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000 sw $4, 0($29) ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004 sw $5, 4($29) ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008 sw $6, 8($29) ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c sw $31, 12($29) ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821 addu $17, $0, $5 ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021 addu $18, $0, $6 ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880 sll $9, $18, 2 ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820 add $9, $4, $9 ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000 lw $10, 0($9) ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
[0040010c] 222bffff addi $11, $17, -1 ; 85: addi $t3, $s1, -1 # t3 = i = low - 1
[00400110] 00116021 addu $12, $0, $17 ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff addi $13, $18, -1 ; 87: addi $t5, $s2, -1 # t5 = high - 1
[00400118] 01ac702a slt $14, $13, $12 ; 90: slt $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
[0040011c] 15c0000e bne $14, $0, 56 [endfor-0x0040011c]
[00400120] 000c4880 sll $9, $12, 2 ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400124] 01244820 add $9, $9, $4 ; 95: add $t1, $t1, $a0 # $t1 = arr + low * 4
[00400128] 8d2f0000 lw $15, 0($9) ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
[0040012c] 01eac02a slt $24, $15, $10 ; 98: slt $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400130] 17000007 bne $24, $0, 28 [endfif-0x00400130]
[00400134] 216b0001 addi $11, $11, 1 ; 102: addi $t3, $t3, 1 # Increase i++
[00400138] 000b2821 addu $5, $0, $11 ; 103: move $a1, $t3
[0040013c] 000c3021 addu $6, $0, $12 ; 104: move $a2, $t4
[00400140] 0c100029 jal 0x004000a4 [SWAP] ; 105: jal SWAP
[00400144] 218c0001 addi $12, $12, 1 ; 107: addi $t4, $t4, 1 # j++
[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
[0040014c] 218c0001 addi $12, $12, 1 ; 111: addi $t4, $t4, 1 # j++
[00400150] 08100046 j 0x00400118 [forloop] ; 112: j forloop
[00400154] 21650001 addi $5, $11, 1 ; 115: addi $a1, $t3, 1 # $a1 = i + 1


Int Regs [16]
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 1001004c
R5 [a1] = 0
R6 [a2] = d
R7 [a3] = 0
R8 [t0] = d
R9 [t1] = 1
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff17c
R30 [s8] = 0
R31 [ra] = 400194
Text
[004000dc] 23bd0010 addi $29, $29, 16 ; 68: addi $sp, $sp, 16
[004000e0] 03e00008 jr $31 ; 69: jr $ra
[004000e4] 23bdfff0 addi $29, $29, -16 ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000 sw $4, 0($29) ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004 sw $5, 4($29) ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008 sw $6, 8($29) ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c sw $31, 12($29) ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821 addu $17, $0, $5 ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021 addu $18, $0, $6 ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880 sll $9, $18, 2 ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820 add $9, $4, $9 ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000 lw $10, 0($9) ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
[0040010c] 222bffff addi $11, $17, -1 ; 85: addi $t3, $s1, -1 # t3 = i = low - 1
[00400110] 00116021 addu $12, $0, $17 ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff addi $13, $18, -1 ; 87: addi $t5, $s2, -1 # t5 = high - 1
[00400118] 01ac702a slt $14, $13, $12 ; 90: slt $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
[0040011c] 15c0000e bne $14, $0, 56 [endfor-0x0040011c]
[00400120] 000c4880 sll $9, $12, 2 ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400124] 01244820 add $9, $9, $4 ; 95: add $t1, $t1, $a0 # $t1 = arr + low * 4
[00400128] 8d2f0000 lw $15, 0($9) ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
[0040012c] 01eac02a slt $24, $15, $10 ; 98: slt $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400130] 17000007 bne $24, $0, 28 [endfif-0x00400130]
[00400134] 216b0001 addi $11, $11, 1 ; 102: addi $t3, $t3, 1 # Increase i++
[00400138] 000b2821 addu $5, $0, $11 ; 103: move $a1, $t3
[0040013c] 000c3021 addu $6, $0, $12 ; 104: move $a2, $t4
[00400140] 0c100029 jal 0x004000a4 [SWAP] ; 105: jal SWAP
[00400144] 218c0001 addi $12, $12, 1 ; 107: addi $t4, $t4, 1 # j++
[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
[0040014c] 218c0001 addi $12, $12, 1 ; 111: addi $t4, $t4, 1 # j++
[00400150] 08100046 j 0x00400118 [forloop] ; 112: j forloop
[00400154] 21650001 addi $5, $11, 1 ; 115: addi $a1, $t3, 1 # $a1 = i + 1


FP Regs
Int Regs [16]
Data
Text
Int Regs [16]
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 1001004c
R5 [a1] = 0
R6 [a2] = d
R7 [a3] = 0
R8 [t0] = d
R9 [t1] = 1
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff17c
R30 [s8] = 0
R31 [ra] = 400194
Text
[004000dc] 23bd0010 addi $29, $29, 16 ; 68: addi $sp, $sp, 16
[004000e0] 03e00008 jr $31 ; 69: jr $ra
[004000e4] 23bdfff0 addi $29, $29, -16 ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000 sw $4, 0($29) ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004 sw $5, 4($29) ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008 sw $6, 8($29) ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c sw $31, 12($29) ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821 addu $17, $0, $5 ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021 addu $18, $0, $6 ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880 sll $9, $18, 2 ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820 add $9, $4, $9 ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000 lw $10, 0($9) ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
[0040010c] 222bffff addi $11, $17, -1 ; 85: addi $t3, $s1, -1 # t3 = i = low - 1
[00400110] 00116021 addu $12, $0, $17 ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff addi $13, $18, -1 ; 87: addi $t5, $s2, -1 # t5 = high - 1
[00400118] 01ac702a slt $14, $13, $12 ; 90: slt $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
[0040011c] 15c0000e bne $14, $0, 56 [endfor-0x0040011c]
[00400120] 000c4880 sll $9, $12, 2 ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400124] 01244820 add $9, $9, $4 ; 95: add $t1, $t1, $a0 # $t1 = arr + low * 4
[00400128] 8d2f0000 lw $15, 0($9) ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
[0040012c] 01eac02a slt $24, $15, $10 ; 98: slt $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400130] 17000007 bne $24, $0, 28 [endfif-0x00400130]
[00400134] 216b0001 addi $11, $11, 1 ; 102: addi $t3, $t3, 1 # Increase i++
[00400138] 000b2821 addu $5, $0, $11 ; 103: move $a1, $t3
[0040013c] 000c3021 addu $6, $0, $12 ; 104: move $a2, $t4
[00400140] 0c100029 jal 0x004000a4 [SWAP] ; 105: jal SWAP
[00400144] 218c0001 addi $12, $12, 1 ; 107: addi $t4, $t4, 1 # j++
[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
[0040014c] 218c0001 addi $12, $12, 1 ; 111: addi $t4, $t4, 1 # j++
[00400150] 08100046 j 0x00400118 [forloop] ; 112: j forloop
[00400154] 21650001 addi $5, $11, 1 ; 115: addi $a1, $t3, 1 # $a1 = i + 1

Int Regs [16]

```
R2   [v0] = 4
R3   [v1] = 0
R4   [a0] = 1001004c
R5   [a1] = 0
R6   [a2] = d
R7   [a3] = 0
R8   [t0] = d
R9   [t1] = 34
R10  [t2] = 0
R11  [t3] = 0
R12  [t4] = 0
R13  [t5] = 0
R14  [t6] = 0
R15  [t7] = 0
R16  [s0] = 0
R17  [s1] = 0
R18  [s2] = d
R19  [s3] = 0
R20  [s4] = 0
R21  [s5] = 0
R22  [s6] = 0
R23  [s7] = 0
R24  [t8] = 0
R25  [t9] = 0
R26  [k0] = 0
R27  [k1] = 0
R28  [gp] = 10008000
R29  [sp] = 7ffff17c
R30  [s8] = 0
R31  [ra] = 400194
```

Text

```
[004000dc] 23bd0010  addi $29, $29, 16         ; 68: addi $sp, $sp, 16
[004000e0] 03e00008  jr $31                    ; 69: jr $ra
[004000e4] 23bdfff0  addi $29, $29, -16        ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000  sw $4, 0($29)             ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004  sw $5, 4($29)             ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008  sw $6, 8($29)             ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c  sw $31, 12($29)           ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821  addu $17, $0, $5          ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021  addu $18, $0, $6          ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880  sll $9, $18, 2            ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820  add $9, $4, $9            ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000  lw $10, 0($9)             ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
[0040010c] 222bffff  addi $11, $17, -1         ; 85: addi $t3, $s1, -1 # t3 = i = low - 1
[00400110] 00116021  addu $12, $0, $17         ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff  addi $13, $18, -1         ; 87: addi $t5, $s2, -1 # t5 = high - 1
[00400118] 01ac702a  slt $14, $13, $12         ; 90: slt $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
[0040011c] 15c0000e  bne $14, $0, 56 [endfor-0x0040011c]
[00400120] 000c4880  sll $9, $12, 2            ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400124] 01244820  add $9, $9, $4            ; 95: add $t1, $t1, $a0 # $t1 = arr + low * 4
[00400128] 8d2f0000  lw $15, 0($9)             ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
[0040012c] 01eac02a  slt $24, $15, $10         ; 98: slt $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400130] 17000007  bne $24, $0, 28 [endfif-0x00400130]
[00400134] 216b0001  addi $11, $11, 1          ; 102: addi $t3, $t3, 1 # Increase i++
[00400138] 000b2821  addu $5, $0, $11          ; 103: move $a1, $t3
[0040013c] 000c3021  addu $6, $0, $12          ; 104: move $a2, $t4
[00400140] 0c100029  jal 0x004000a4 [SWAP]     ; 105: jal SWAP
[00400144] 218c0001  addi $12, $12, 1          ; 107: addi $t4, $t4, 1 # j++
[00400148] 08100046  j 0x00400118 [forloop]    ; 108: j forloop
[0040014c] 218c0001  addi $12, $12, 1          ; 111: addi $t4, $t4, 1 # j++
[00400150] 08100046  j 0x00400118 [forloop]    ; 112: j forloop
[00400154] 21650001  addi $5, $11, 1           ; 115: addi $a1, $t3, 1 # $a1 = i + 1
```

Int Regs [16]

```
R2   [v0] = 4
R3   [v1] = 0
R4   [a0] = 1001004c
R5   [a1] = 0
R6   [a2] = d
R7   [a3] = 0
R8   [t0] = d
R9   [t1] = 10010080
R10  [t2] = 0
R11  [t3] = 0
R12  [t4] = 0
R13  [t5] = 0
R14  [t6] = 0
R15  [t7] = 0
R16  [s0] = 0
R17  [s1] = 0
R18  [s2] = d
R19  [s3] = 0
R20  [s4] = 0
R21  [s5] = 0
R22  [s6] = 0
R23  [s7] = 0
R24  [t8] = 0
R25  [t9] = 0
R26  [k0] = 0
R27  [k1] = 0
R28  [gp] = 10008000
R29  [sp] = 7ffff17c
R30  [s8] = 0
R31  [ra] = 400194
```

Text

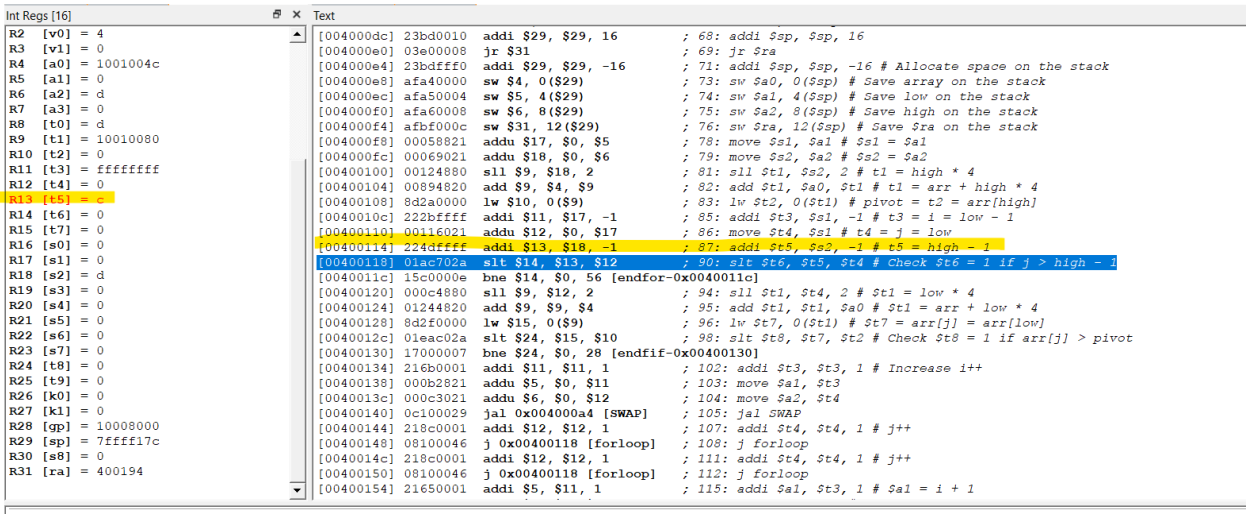
```
[004000dc] 23bd0010  addi $29, $29, 16         ; 68: addi $sp, $sp, 16
[004000e0] 03e00008  jr $31                    ; 69: jr $ra
[004000e4] 23bdfff0  addi $29, $29, -16        ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000  sw $4, 0($29)             ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004  sw $5, 4($29)             ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008  sw $6, 8($29)             ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c  sw $31, 12($29)           ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821  addu $17, $0, $5          ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021  addu $18, $0, $6          ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880  sll $9, $18, 2            ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820  add $9, $4, $9            ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000  lw $10, 0($9)             ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
[0040010c] 222bffff  addi $11, $17, -1         ; 85: addi $t3, $s1, -1 # t3 = i = low - 1
[00400110] 00116021  addu $12, $0, $17         ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff  addi $13, $18, -1         ; 87: addi $t5, $s2, -1 # t5 = high - 1
[00400118] 01ac702a  slt $14, $13, $12         ; 90: slt $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
[0040011c] 15c0000e  bne $14, $0, 56 [endfor-0x0040011c]
[00400120] 000c4880  sll $9, $12, 2            ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400124] 01244820  add $9, $9, $4            ; 95: add $t1, $t1, $a0 # $t1 = arr + low * 4
[00400128] 8d2f0000  lw $15, 0($9)             ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
[0040012c] 01eac02a  slt $24, $15, $10         ; 98: slt $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400130] 17000007  bne $24, $0, 28 [endfif-0x00400130]
[00400134] 216b0001  addi $11, $11, 1          ; 102: addi $t3, $t3, 1 # Increase i++
[00400138] 000b2821  addu $5, $0, $11          ; 103: move $a1, $t3
[0040013c] 000c3021  addu $6, $0, $12          ; 104: move $a2, $t4
[00400140] 0c100029  jal 0x004000a4 [SWAP]     ; 105: jal SWAP
[00400144] 218c0001  addi $12, $12, 1          ; 107: addi $t4, $t4, 1 # j++
[00400148] 08100046  j 0x00400118 [forloop]    ; 108: j forloop
[0040014c] 218c0001  addi $12, $12, 1          ; 111: addi $t4, $t4, 1 # j++
[00400150] 08100046  j 0x00400118 [forloop]    ; 112: j forloop
[00400154] 21650001  addi $5, $11, 1           ; 115: addi $a1, $t3, 1 # $a1 = i + 1
```

Int Regs [16]

```
R2   [v0] = 4
R3   [v1] = 0
R4   [a0] = 1001004c
R5   [a1] = 0
R6   [a2] = d
R7   [a3] = 0
R8   [t0] = d
R9   [t1] = 10010080
R10  [t2] = 0
R11  [t3] = ffffffff
R12  [t4] = 0
R13  [t5] = 0
R14  [t6] = 0
R15  [t7] = 0
R16  [s0] = 0
R17  [s1] = 0
R18  [s2] = d
R19  [s3] = 0
R20  [s4] = 0
R21  [s5] = 0
R22  [s6] = 0
R23  [s7] = 0
R24  [t8] = 0
R25  [t9] = 0
R26  [k0] = 0
R27  [k1] = 0
R28  [gp] = 10008000
R29  [sp] = 7ffff17c
R30  [s8] = 0
R31  [ra] = 400194
```

Text

```
[004000dc] 23bd0010  addi $29, $29, 16         ; 68: addi $sp, $sp, 16
[004000e0] 03e00008  jr $31                    ; 69: jr $ra
[004000e4] 23bdfff0  addi $29, $29, -16        ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000  sw $4, 0($29)             ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004  sw $5, 4($29)             ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008  sw $6, 8($29)             ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c  sw $31, 12($29)           ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821  addu $17, $0, $5          ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021  addu $18, $0, $6          ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880  sll $9, $18, 2            ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820  add $9, $4, $9            ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000  lw $10, 0($9)             ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
[0040010c] 222bffff  addi $11, $17, -1         ; 85: addi $t3, $s1, -1 # t3 = i = low - 1
[00400110] 00116021  addu $12, $0, $17         ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff  addi $13, $18, -1         ; 87: addi $t5, $s2, -1 # t5 = high - 1
[00400118] 01ac702a  slt $14, $13, $12         ; 90: slt $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
[0040011c] 15c0000e  bne $14, $0, 56 [endfor-0x0040011c]
[00400120] 000c4880  sll $9, $12, 2            ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400124] 01244820  add $9, $9, $4            ; 95: add $t1, $t1, $a0 # $t1 = arr + low * 4
[00400128] 8d2f0000  lw $15, 0($9)             ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
[0040012c] 01eac02a  slt $24, $15, $10         ; 98: slt $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400130] 17000007  bne $24, $0, 28 [endfif-0x00400130]
[00400134] 216b0001  addi $11, $11, 1          ; 102: addi $t3, $t3, 1 # Increase i++
[00400138] 000b2821  addu $5, $0, $11          ; 103: move $a1, $t3
[0040013c] 000c3021  addu $6, $0, $12          ; 104: move $a2, $t4
[00400140] 0c100029  jal 0x004000a4 [SWAP]     ; 105: jal SWAP
[00400144] 218c0001  addi $12, $12, 1          ; 107: addi $t4, $t4, 1 # j++
[00400148] 08100046  j 0x00400118 [forloop]    ; 108: j forloop
[0040014c] 218c0001  addi $12, $12, 1          ; 111: addi $t4, $t4, 1 # j++
[00400150] 08100046  j 0x00400118 [forloop]    ; 112: j forloop
[00400154] 21650001  addi $5, $11, 1           ; 115: addi $a1, $t3, 1 # $a1 = i + 1
```

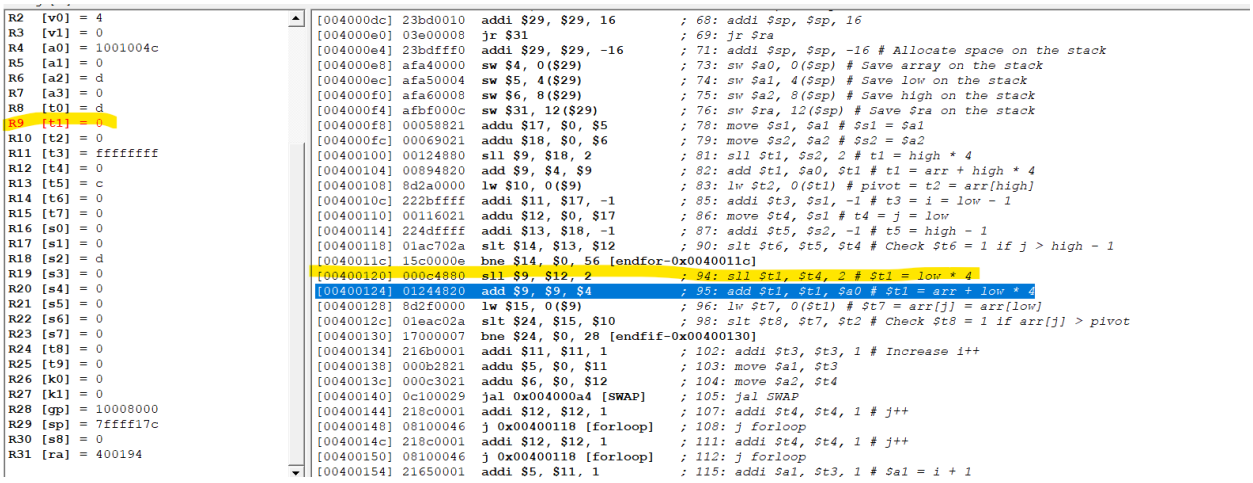


Int Regs [16]

```
R2  [v0] = 4
R3  [v1] = 0
R4  [a0] = 1001004c
R5  [a1] = 0
R6  [a2] = d
R7  [a3] = 0
R8  [t0] = d
R9  [t1] = 10010080
R10 [t2] = 0
R11 [t3] = ffffffff
R12 [t4] = 0
R13 [t5] = c
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff17c
R30 [s8] = 0
R31 [ra] = 400194
```

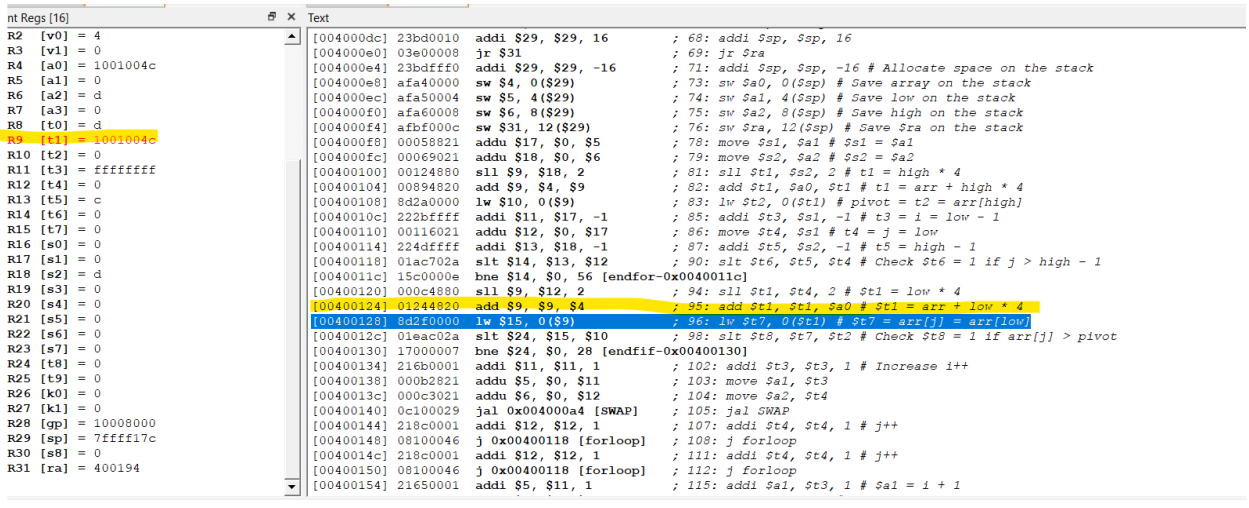
Text

```
[004000dc] 23bd0010  addi $29, $29, 16         ; 68: addi $sp, $sp, 16
[004000e0] 03e00008  jr $31                    ; 69: jr $ra
[004000e4] 23bdfff0  addi $29, $29, -16        ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000  sw $4, 0($29)             ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004  sw $5, 4($29)             ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008  sw $6, 8($29)             ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c  sw $31, 12($29)           ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821  addu $17, $0, $5          ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021  addu $18, $0, $6          ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880  sll $9, $18, 2            ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820  add $9, $4, $9            ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000  lw $10, 0($9)             ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
[0040010c] 222bffff  addi $11, $17, -1         ; 85: addi $t3, $s1, -1 # t3 = i = low - 1
[00400110] 00116021  addu $12, $0, $17         ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff  addi $13, $18, -1         ; 87: addi $t5, $s2, -1 # t5 = high - 1
[00400118] 01ac702a  slt $14, $13, $12         ; 90: slt $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
[0040011c] 15c0000e  bne $14, $0, 56 [endfor-0x0040011c]
[00400120] 000c4880  sll $9, $12, 2            ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400124] 01244820  add $9, $9, $4            ; 95: add $t1, $t1, $a0 # $t1 = arr + low * 4
[00400128] 8d2f0000  lw $15, 0($9)             ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
[0040012c] 01eac02a  slt $24, $15, $10         ; 98: slt $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400130] 17000007  bne $24, $0, 28 [endfif-0x00400130]
[00400134] 216b0001  addi $11, $11, 1          ; 102: addi $t3, $t3, 1 # Increase i++
[00400138] 000b2821  addu $5, $0, $11          ; 103: move $a1, $t3
[0040013c] 000c3021  addu $6, $0, $12          ; 104: move $a2, $t4
[00400140] 0c100029  jal 0x004000a4 [SWAP]     ; 105: jal SWAP
[00400144] 218c0001  addi $12, $12, 1          ; 107: addi $t4, $t4, 1 # j++
[00400148] 08100046  j 0x00400118 [forloop]    ; 108: j forloop
[0040014c] 218c0001  addi $12, $12, 1          ; 111: addi $t4, $t4, 1 # j++
[00400150] 08100046  j 0x00400118 [forloop]    ; 112: j forloop
[00400154] 21650001  addi $5, $11, 1           ; 115: addi $a1, $t3, 1 # $a1 = i + 1
```



```
R2  [v0] = 4
R3  [v1] = 0
R4  [a0] = 1001004c
R5  [a1] = 0
R6  [a2] = d
R7  [a3] = 0
R8  [t0] = d
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = ffffffff
R12 [t4] = 0
R13 [t5] = c
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff17c
R30 [s8] = 0
R31 [ra] = 400194
```

```
[004000dc] 23bd0010  addi $29, $29, 16         ; 68: addi $sp, $sp, 16
[004000e0] 03e00008  jr $31                    ; 69: jr $ra
[004000e4] 23bdfff0  addi $29, $29, -16        ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000  sw $4, 0($29)             ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004  sw $5, 4($29)             ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008  sw $6, 8($29)             ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c  sw $31, 12($29)           ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821  addu $17, $0, $5          ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021  addu $18, $0, $6          ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880  sll $9, $18, 2            ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820  add $9, $4, $9            ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000  lw $10, 0($9)             ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
[0040010c] 222bffff  addi $11, $17, -1         ; 85: addi $t3, $s1, -1 # t3 = i = low - 1
[00400110] 00116021  addu $12, $0, $17         ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff  addi $13, $18, -1         ; 87: addi $t5, $s2, -1 # t5 = high - 1
[00400118] 01ac702a  slt $14, $13, $12         ; 90: slt $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
[0040011c] 15c0000e  bne $14, $0, 56 [endfor-0x0040011c]
[00400120] 000c4880  sll $9, $12, 2            ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400124] 01244820  add $9, $9, $4            ; 95: add $t1, $t1, $a0 # $t1 = arr + low * 4
[00400128] 8d2f0000  lw $15, 0($9)             ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
[0040012c] 01eac02a  slt $24, $15, $10         ; 98: slt $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400130] 17000007  bne $24, $0, 28 [endfif-0x00400130]
[00400134] 216b0001  addi $11, $11, 1          ; 102: addi $t3, $t3, 1 # Increase i++
[00400138] 000b2821  addu $5, $0, $11          ; 103: move $a1, $t3
[0040013c] 000c3021  addu $6, $0, $12          ; 104: move $a2, $t4
[00400140] 0c100029  jal 0x004000a4 [SWAP]     ; 105: jal SWAP
[00400144] 218c0001  addi $12, $12, 1          ; 107: addi $t4, $t4, 1 # j++
[00400148] 08100046  j 0x00400118 [forloop]    ; 108: j forloop
[0040014c] 218c0001  addi $12, $12, 1          ; 111: addi $t4, $t4, 1 # j++
[00400150] 08100046  j 0x00400118 [forloop]    ; 112: j forloop
[00400154] 21650001  addi $5, $11, 1           ; 115: addi $a1, $t3, 1 # $a1 = i + 1
```

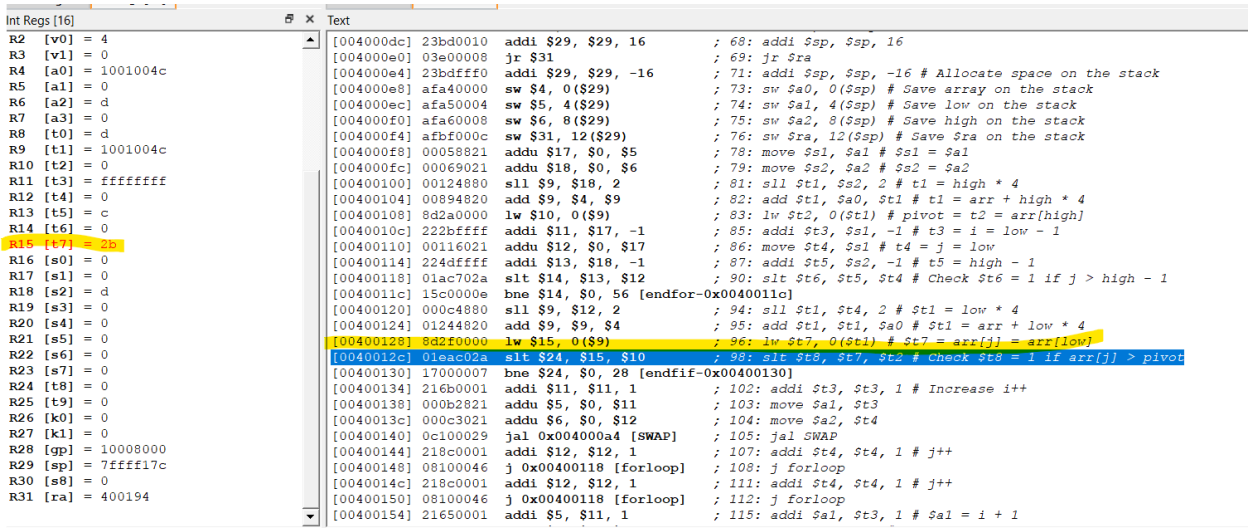


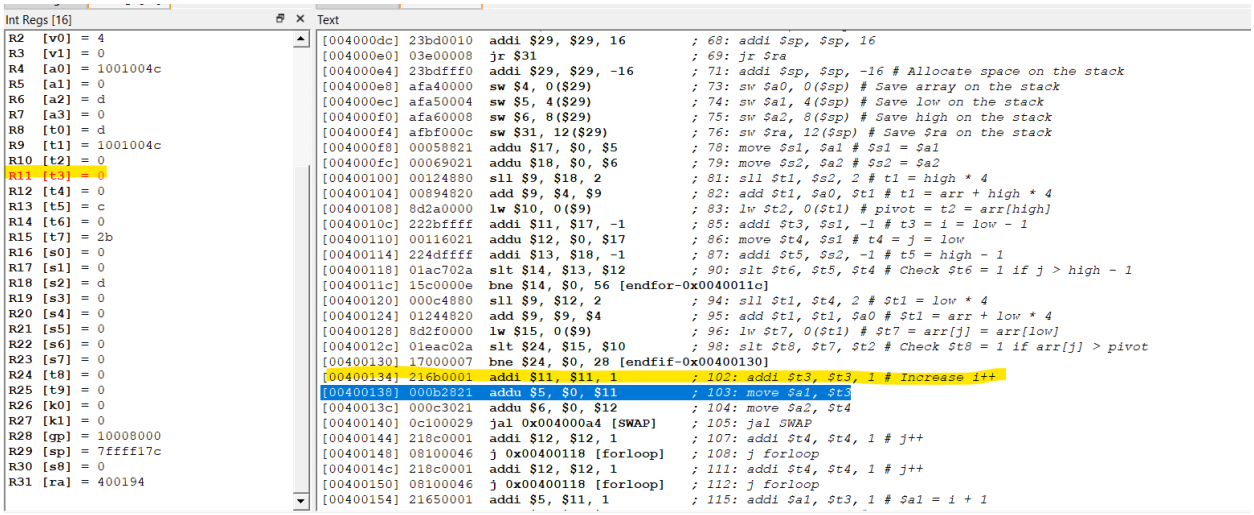
nt Regs [16]

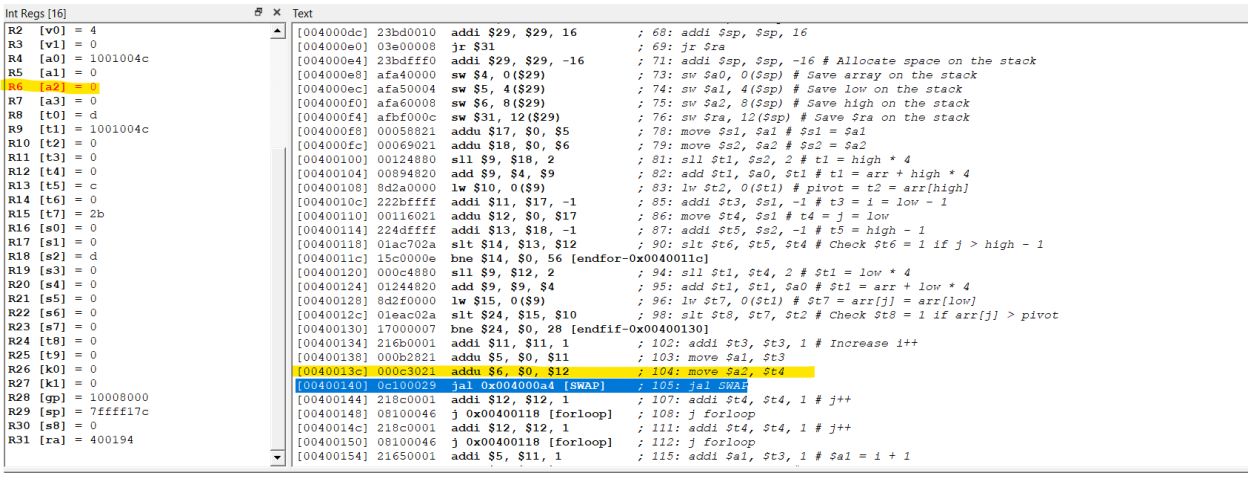
```
R2  [v0] = 4
R3  [v1] = 0
R4  [a0] = 1001004c
R5  [a1] = 0
R6  [a2] = d
R7  [a3] = 0
R8  [t0] = d
R9  [t1] = 1001004c
R10 [t2] = 0
R11 [t3] = ffffffff
R12 [t4] = 0
R13 [t5] = c
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff17c
R30 [s8] = 0
R31 [ra] = 400194
```

Text

```
[004000dc] 23bd0010  addi $29, $29, 16         ; 68: addi $sp, $sp, 16
[004000e0] 03e00008  jr $31                    ; 69: jr $ra
[004000e4] 23bdfff0  addi $29, $29, -16        ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000  sw $4, 0($29)             ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004  sw $5, 4($29)             ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008  sw $6, 8($29)             ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c  sw $31, 12($29)           ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821  addu $17, $0, $5          ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021  addu $18, $0, $6          ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880  sll $9, $18, 2            ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820  add $9, $4, $9            ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000  lw $10, 0($9)             ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
[0040010c] 222bffff  addi $11, $17, -1         ; 85: addi $t3, $s1, -1 # t3 = i = low - 1
[00400110] 00116021  addu $12, $0, $17         ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff  addi $13, $18, -1         ; 87: addi $t5, $s2, -1 # t5 = high - 1
[00400118] 01ac702a  slt $14, $13, $12         ; 90: slt $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
[0040011c] 15c0000e  bne $14, $0, 56 [endfor-0x0040011c]
[00400120] 000c4880  sll $9, $12, 2            ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400124] 01244820  add $9, $9, $4            ; 95: add $t1, $t1, $a0 # $t1 = arr + low * 4
[00400128] 8d2f0000  lw $15, 0($9)             ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
[0040012c] 01eac02a  slt $24, $15, $10         ; 98: slt $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400130] 17000007  bne $24, $0, 28 [endfif-0x00400130]
[00400134] 216b0001  addi $11, $11, 1          ; 102: addi $t3, $t3, 1 # Increase i++
[00400138] 000b2821  addu $5, $0, $11          ; 103: move $a1, $t3
[0040013c] 000c3021  addu $6, $0, $12          ; 104: move $a2, $t4
[00400140] 0c100029  jal 0x004000a4 [SWAP]     ; 105: jal SWAP
[00400144] 218c0001  addi $12, $12, 1          ; 107: addi $t4, $t4, 1 # j++
[00400148] 08100046  j 0x00400118 [forloop]    ; 108: j forloop
[0040014c] 218c0001  addi $12, $12, 1          ; 111: addi $t4, $t4, 1 # j++
[00400150] 08100046  j 0x00400118 [forloop]    ; 112: j forloop
[00400154] 21650001  addi $5, $11, 1           ; 115: addi $a1, $t3, 1 # $a1 = i + 1
```


Int Regs [16]
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 1001004c
R5 [a1] = 0
R6 [a2] = d
R7 [a3] = 0
R8 [t0] = d
R9 [t1] = 1001004c
R10 [t2] = 0
R11 [t3] = ffffffff
R12 [t4] = 0
R13 [t5] = c
R14 [t6] = 0
R15 [t7] = 2b
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff17c
R30 [s8] = 0
R31 [ra] = 400194
Text
[004000dc] 23bd0010 addi $29, $29, 16 ; 68: addi $sp, $sp, 16
[004000e0] 03e00008 jr $31 ; 69: jr $ra
[004000e4] 23bdfff0 addi $29, $29, -16 ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000 sw $4, 0($29) ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004 sw $5, 4($29) ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008 sw $6, 8($29) ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c sw $31, 12($29) ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821 addu $17, $0, $5 ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021 addu $18, $0, $6 ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880 sll $9, $18, 2 ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820 add $9, $4, $9 ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000 lw $10, 0($9) ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
[0040010c] 222bffff addi $11, $17, -1 ; 85: addi $t3, $s1, -1 # t3 = i = low - 1
[00400110] 00116021 addu $12, $0, $17 ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff addi $13, $18, -1 ; 87: addi $t5, $s2, -1 # t5 = high - 1
[00400118] 01ac702a slt $14, $13, $12 ; 90: slt $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
[0040011c] 15c0000e bne $14, $0, 56 [endfor-0x0040011c]
[00400120] 000c4880 sll $9, $12, 2 ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400124] 01244820 add $9, $9, $4 ; 95: add $t1, $t1, $a0 # $t1 = arr + low * 4
[00400128] 8d2f0000 lw $15, 0($9) ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
[0040012c] 01eac02a slt $24, $15, $10 ; 98: slt $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400130] 17000007 bne $24, $0, 28 [endfif-0x00400130]
[00400134] 216b0001 addi $11, $11, 1 ; 102: addi $t3, $t3, 1 # Increase i++
[00400138] 000b2821 addu $5, $0, $11 ; 103: move $a1, $t3
[0040013c] 000c3021 addu $6, $0, $12 ; 104: move $a2, $t4
[00400140] 0c100029 jal 0x004000a4 [SWAP] ; 105: jal SWAP
[00400144] 218c0001 addi $12, $12, 1 ; 107: addi $t4, $t4, 1 # j++
[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
[0040014c] 218c0001 addi $12, $12, 1 ; 111: addi $t4, $t4, 1 # j++
[00400150] 08100046 j 0x00400118 [forloop] ; 112: j forloop
[00400154] 21650001 addi $5, $11, 1 ; 115: addi $a1, $t3, 1 # $a1 = i + 1


Int Regs [16]
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 1001004c
R5 [a1] = 0
R6 [a2] = d
R7 [a3] = 0
R8 [t0] = d
R9 [t1] = 1001004c
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = c
R14 [t6] = 0
R15 [t7] = 2b
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff17c
R30 [s8] = 0
R31 [ra] = 400194
Text
[004000dc] 23bd0010 addi $29, $29, 16 ; 68: addi $sp, $sp, 16
[004000e0] 03e00008 jr $31 ; 69: jr $ra
[004000e4] 23bdfff0 addi $29, $29, -16 ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000 sw $4, 0($29) ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004 sw $5, 4($29) ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008 sw $6, 8($29) ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c sw $31, 12($29) ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821 addu $17, $0, $5 ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021 addu $18, $0, $6 ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880 sll $9, $18, 2 ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820 add $9, $4, $9 ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000 lw $10, 0($9) ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
[0040010c] 222bffff addi $11, $17, -1 ; 85: addi $t3, $s1, -1 # t3 = i = low - 1
[00400110] 00116021 addu $12, $0, $17 ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff addi $13, $18, -1 ; 87: addi $t5, $s2, -1 # t5 = high - 1
[00400118] 01ac702a slt $14, $13, $12 ; 90: slt $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
[0040011c] 15c0000e bne $14, $0, 56 [endfor-0x0040011c]
[00400120] 000c4880 sll $9, $12, 2 ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400124] 01244820 add $9, $9, $4 ; 95: add $t1, $t1, $a0 # $t1 = arr + low * 4
[00400128] 8d2f0000 lw $15, 0($9) ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
[0040012c] 01eac02a slt $24, $15, $10 ; 98: slt $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400130] 17000007 bne $24, $0, 28 [endfif-0x00400130]
[00400134] 216b0001 addi $11, $11, 1 ; 102: addi $t3, $t3, 1 # Increase i++
[00400138] 000b2821 addu $5, $0, $11 ; 103: move $a1, $t3
[0040013c] 000c3021 addu $6, $0, $12 ; 104: move $a2, $t4
[00400140] 0c100029 jal 0x004000a4 [SWAP] ; 105: jal SWAP
[00400144] 218c0001 addi $12, $12, 1 ; 107: addi $t4, $t4, 1 # j++
[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
[0040014c] 218c0001 addi $12, $12, 1 ; 111: addi $t4, $t4, 1 # j++
[00400150] 08100046 j 0x00400118 [forloop] ; 112: j forloop
[00400154] 21650001 addi $5, $11, 1 ; 115: addi $a1, $t3, 1 # $a1 = i + 1


Int Regs [16]
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 1001004c
R5 [a1] = 0
R6 [a2] = 0
R7 [a3] = 0
R8 [t0] = d
R9 [t1] = 1001004c
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = c
R14 [t6] = 0
R15 [t7] = 2b
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff17c
R30 [s8] = 0
R31 [ra] = 400194
Text
[004000dc] 23bd0010 addi $29, $29, 16 ; 68: addi $sp, $sp, 16
[004000e0] 03e00008 jr $31 ; 69: jr $ra
[004000e4] 23bdfff0 addi $29, $29, -16 ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000 sw $4, 0($29) ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004 sw $5, 4($29) ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008 sw $6, 8($29) ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c sw $31, 12($29) ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821 addu $17, $0, $5 ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021 addu $18, $0, $6 ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880 sll $9, $18, 2 ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820 add $9, $4, $9 ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000 lw $10, 0($9) ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
[0040010c] 222bffff addi $11, $17, -1 ; 85: addi $t3, $s1, -1 # t3 = i = low - 1
[00400110] 00116021 addu $12, $0, $17 ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff addi $13, $18, -1 ; 87: addi $t5, $s2, -1 # t5 = high - 1
[00400118] 01ac702a slt $14, $13, $12 ; 90: slt $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
[0040011c] 15c0000e bne $14, $0, 56 [endfor-0x0040011c]
[00400120] 000c4880 sll $9, $12, 2 ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400124] 01244820 add $9, $9, $4 ; 95: add $t1, $t1, $a0 # $t1 = arr + low * 4
[00400128] 8d2f0000 lw $15, 0($9) ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
[0040012c] 01eac02a slt $24, $15, $10 ; 98: slt $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400130] 17000007 bne $24, $0, 28 [endfif-0x00400130]
[00400134] 216b0001 addi $11, $11, 1 ; 102: addi $t3, $t3, 1 # Increase i++
[00400138] 000b2821 addu $5, $0, $11 ; 103: move $a1, $t3
[0040013c] 000c3021 addu $6, $0, $12 ; 104: move $a2, $t4
[00400140] 0c100029 jal 0x004000a4 [SWAP] ; 105: jal SWAP
[00400144] 218c0001 addi $12, $12, 1 ; 107: addi $t4, $t4, 1 # j++
[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
[0040014c] 218c0001 addi $12, $12, 1 ; 111: addi $t4, $t4, 1 # j++
[00400150] 08100046 j 0x00400118 [forloop] ; 112: j forloop
[00400154] 21650001 addi $5, $11, 1 ; 115: addi $a1, $t3, 1 # $a1 = i + 1

Int Regs [16]

```
R2  [v0] = 4
R3  [v1] = 0
R4  [a0] = 1001004c
R5  [a1] = 0
R6  [a2] = 0
R7  [a3] = 0
R8  [t0] = d
R9  [t1] = 1001004c
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = c
R14 [t6] = 0
R15 [t7] = 2b
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff17c
R30 [s8] = 0
R31 [ra] = 400144
```

Text

```
[004000cc] 8d540000  lw $20, 0($10)          ; 62: lw $s4, 0($t2) # $s4 = arr[$t2]
[004000d0] ad340000  sw $20, 0($9)           ; 64: sw $s4, 0($t1) # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
[004000d4] ad530000  sw $19, 0($10)          ; 65: sw $s3, 0($t2) # arr[$t2] = $s3
[004000d8] 8faa000c  lw $10, 12($29)         ; 67: lw $t2, 12($sp) # Load $t2 on the stack
[004000dc] 23bd0010  addi $29, $29, 16       ; 68: addi $sp, $sp, 16
[004000e0] 03e00008  jr $31                  ; 69: jr $ra
[004000e4] 23bdfff0  addi $29, $29, -16      ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000  sw $4, 0($29)           ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004  sw $5, 4($29)           ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008  sw $6, 8($29)           ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c  sw $31, 12($29)         ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821  addu $17, $0, $5        ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021  addu $18, $0, $6        ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880  sll $9, $18, 2          ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820  add $9, $4, $9          ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000  lw $10, 0($9)           ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
[0040010c] 222bffff  addi $11, $17, -1       ; 85: addi $t3, $s1, -1 # t3 = i = low - 1
[00400110] 00116021  addu $12, $0, $17       ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff  addi $13, $18, -1       ; 87: addi $t5, $s2, -1 # t5 = high - 1
[00400118] 01ac702a  slt $14, $13, $12       ; 90: slt $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
[0040011c] 15c0000e  bne $14, $0, 56 [endfor-0x0040011c]
[00400120] 000c4880  sll $9, $12, 2          ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400124] 01244820  add $9, $9, $4          ; 95: add $t1, $t1, $a0 # $t1 = arr + low * 4
[00400128] 8d2f0000  lw $15, 0($9)           ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
[0040012c] 01eac02a  slt $24, $15, $10       ; 98: slt $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400130] 17000007  bne $24, $0, 28 [endfif-0x00400130]
[00400134] 216b0001  addi $11, $11, 1        ; 102: addi $t3, $t3, 1 # Increase i++
[00400138] 000b2821  addu $5, $0, $11        ; 103: move $a1, $t3
[0040013c] 000c3021  addu $6, $0, $12        ; 104: move $a2, $t4
[00400140] 0c100029  jal 0x004000a4 [SWAP]   ; 105: jal SWAP
[00400144] 218c0001  addi $12, $12, 1        ; 107: addi $t4, $t4, 1 # j++
```

Int Regs [16]

```
R2  [v0] = 4
R3  [v1] = 0
R4  [a0] = 1001004c
R5  [a1] = 0
R6  [a2] = 0
R7  [a3] = 0
R8  [t0] = d
R9  [t1] = 1001004c
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = c
R14 [t6] = 0
R15 [t7] = 2b
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff16c
R30 [s8] = 0
R31 [ra] = 400144
```

Text

```
[0040009c] 3402000a  ori $2, $0, 10           ; 45: li $v0, 10
[004000a0] 0000000c  syscall                  ; 46: syscall
[004000a4] 23bdfff0  addi $29, $29, -16       ; 49: addi $sp, $sp, -16 # Allocate space on the stack
[004000a8] afa40000  sw $4, 0($29)            ; 51: sw $a0, 0($sp) # Store array
[004000ac] afa50004  sw $5, 4($29)            ; 52: sw $a1, 4($sp) # Store Low
[004000b0] afa60008  sw $6, 8($29)            ; 53: sw $a2, 8($sp) # Store High
[004000b4] afaa000c  sw $10, 12($29)          ; 54: sw $t2, 12($sp) # Store t2 for high
[004000b8] 00054880  sll $9, $5, 2            ; 56: sll $t1, $a1, 2 # $t1 = Low * 4
[004000bc] 00894820  add $9, $4, $9           ; 57: add $t1, $a0, $t1 # $t1 = arr + Low * 4
[004000c0] 8d330000  lw $19, 0($9)            ; 58: lw $s3, 0($t1) # $s3 = arr[$t1]
[004000c4] 00065080  sll $10, $6, 2           ; 60: sll $t2, $a2, 2 # $t2 = High * 4
[004000c8] 008a5020  add $10, $4, $10         ; 61: add $t2, $a0, $t2 # $t2 = arr + High * 4
[004000cc] 8d540000  lw $20, 0($10)           ; 62: lw $s4, 0($t2) # $s4 = arr[$t2]
[004000d0] ad340000  sw $20, 0($9)            ; 64: sw $s4, 0($t1) # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
[004000d4] ad530000  sw $19, 0($10)           ; 65: sw $s3, 0($t2) # arr[$t2] = $s3
[004000d8] 8faa000c  lw $10, 12($29)          ; 67: lw $t2, 12($sp) # Load $t2 on the stack
[004000dc] 23bd0010  addi $29, $29, 16        ; 68: addi $sp, $sp, 16
[004000e0] 03e00008  jr $31                   ; 69: jr $ra
[004000e4] 23bdfff0  addi $29, $29, -16       ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000  sw $4, 0($29)            ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004  sw $5, 4($29)            ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008  sw $6, 8($29)            ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c  sw $31, 12($29)          ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821  addu $17, $0, $5         ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021  addu $18, $0, $6         ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880  sll $9, $18, 2           ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820  add $9, $4, $9           ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000  lw $10, 0($9)            ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
[0040010c] 222bffff  addi $11, $17, -1        ; 85: addi $t3, $s1, -1 # t3 = i = low - 1
[00400110] 00116021  addu $12, $0, $17        ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff  addi $13, $18, -1        ; 87: addi $t5, $s2, -1 # t5 = high - 1
```

Int Regs [16]

```
R2  [v0] = 4
R3  [v1] = 0
R4  [a0] = 1001004c
R5  [a1] = 0
R6  [a2] = 0
R7  [a3] = 0
R8  [t0] = d
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = c
R14 [t6] = 0
R15 [t7] = 2b
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff16c
R30 [s8] = 0
R31 [ra] = 400144
```

Text

```
[0040009c] 3402000a  ori $2, $0, 10           ; 45: li $v0, 10
[004000a0] 0000000c  syscall                  ; 46: syscall
[004000a4] 23bdfff0  addi $29, $29, -16       ; 49: addi $sp, $sp, -16 # Allocate space on the stack
[004000a8] afa40000  sw $4, 0($29)            ; 51: sw $a0, 0($sp) # Store array
[004000ac] afa50004  sw $5, 4($29)            ; 52: sw $a1, 4($sp) # Store Low
[004000b0] afa60008  sw $6, 8($29)            ; 53: sw $a2, 8($sp) # Store High
[004000b4] afaa000c  sw $10, 12($29)          ; 54: sw $t2, 12($sp) # Store t2 for high
[004000b8] 00054880  sll $9, $5, 2            ; 56: sll $t1, $a1, 2 # $t1 = Low * 4
[004000bc] 00894820  add $9, $4, $9           ; 57: add $t1, $a0, $t1 # $t1 = arr + Low * 4
[004000c0] 8d330000  lw $19, 0($9)            ; 58: lw $s3, 0($t1) # $s3 = arr[$t1]
[004000c4] 00065080  sll $10, $6, 2           ; 60: sll $t2, $a2, 2 # $t2 = High * 4
[004000c8] 008a5020  add $10, $4, $10         ; 61: add $t2, $a0, $t2 # $t2 = arr + High * 4
[004000cc] 8d540000  lw $20, 0($10)           ; 62: lw $s4, 0($t2) # $s4 = arr[$t2]
[004000d0] ad340000  sw $20, 0($9)            ; 64: sw $s4, 0($t1) # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
[004000d4] ad530000  sw $19, 0($10)           ; 65: sw $s3, 0($t2) # arr[$t2] = $s3
[004000d8] 8faa000c  lw $10, 12($29)          ; 67: lw $t2, 12($sp) # Load $t2 on the stack
[004000dc] 23bd0010  addi $29, $29, 16        ; 68: addi $sp, $sp, 16
[004000e0] 03e00008  jr $31                   ; 69: jr $ra
[004000e4] 23bdfff0  addi $29, $29, -16       ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000  sw $4, 0($29)            ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004  sw $5, 4($29)            ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008  sw $6, 8($29)            ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c  sw $31, 12($29)          ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821  addu $17, $0, $5         ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021  addu $18, $0, $6         ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880  sll $9, $18, 2           ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820  add $9, $4, $9           ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000  lw $10, 0($9)            ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
[0040010c] 222bffff  addi $11, $17, -1        ; 85: addi $t3, $s1, -1 # t3 = i = low - 1
[00400110] 00116021  addu $12, $0, $17        ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff  addi $13, $18, -1        ; 87: addi $t5, $s2, -1 # t5 = high - 1
```

Int Regs [16]

```
R2  [v0] = 4
R3  [v1] = 0
R4  [a0] = 1001004c
R5  [a1] = 0
R6  [a2] = 0
R7  [a3] = 0
R8  [t0] = d
R9  [t1] = 1001004c
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = c
R14 [t6] = 0
R15 [t7] = 2b
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff16c
R30 [s8] = 0
R31 [ra] = 400144
```

Text

```
[0040009c] 3402000a  ori $2, $0, 10          ; 45: li $v0, 10
[004000a0] 0000000c  syscall                 ; 46: syscall
[004000a4] 23bdfff0  addi $29, $29, -16      ; 49: addi $sp, $sp, -16 # Allocate space on the stack
[004000a8] afa40000  sw $4, 0($29)           ; 51: sw $a0, 0($sp) # Store array
[004000ac] afa50004  sw $5, 4($29)           ; 52: sw $a1, 4($sp) # Store Low
[004000b0] afa60008  sw $6, 8($29)           ; 53: sw $a2, 8($sp) # Store High
[004000b4] afaa000c  sw $10, 12($29)         ; 54: sw $t2, 12($sp) # Store t2 for high
[004000b8] 00054880  sll $9, $5, 2           ; 56: sll $t1, $a1, 2 # $t1 = Low * 4
[004000bc] 00894820  add $9, $4, $9          ; 57: add $t1, $a0, $t1 # $t1 = arr + Low * 4
[004000c0] 8d330000  lw $19, 0($9)           ; 58: lw $s3, 0($t1) # $s3 = arr[$t1]
[004000c4] 00065080  sll $10, $6, 2          ; 60: sll $t2, $a2, 2 # $t2 = High * 4
[004000c8] 008a5020  add $10, $4, $10        ; 61: add $t2, $a0, $t2 # $t2 = arr + High * 4
[004000cc] 8d540000  lw $20, 0($10)          ; 62: lw $s4, 0($t2) # $s4 = arr[$t2]
[004000d0] ad340000  sw $20, 0($9)           ; 64: sw $s4, 0($t1) # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
[004000d4] ad530000  sw $19, 0($10)          ; 65: sw $s3, 0($t2) # arr[$t2] = $s3
[004000d8] 8faa000c  lw $10, 12($29)         ; 67: lw $t2, 12($sp) # Load $t2 on the stack
[004000dc] 23bd0010  addi $29, $29, 16       ; 68: addi $sp, $sp, 16
[004000e0] 03e00008  jr $31                  ; 69: jr $ra
[004000e4] 23bdfff0  addi $29, $29, -16      ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000  sw $4, 0($29)           ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004  sw $5, 4($29)           ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008  sw $6, 8($29)           ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c  sw $31, 12($29)         ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821  addu $17, $0, $5        ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021  addu $18, $0, $6        ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880  sll $9, $18, 2          ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820  add $9, $4, $9          ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000  lw $10, 0($9)           ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
[0040010c] 222bffff  addi $11, $17, -1       ; 85: addi $t3, $s1, -1 # t3 = i = low - 1
[00400110] 00116021  addu $12, $0, $17       ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff  addi $13, $18, -1       ; 87: addi $t5, $s2, -1 # t5 = high - 1
```


Int Regs [16]

```
R2  [v0] = 4
R3  [v1] = 0
R4  [a0] = 1001004c
R5  [a1] = 0
R6  [a2] = 0
R7  [a3] = 0
R8  [t0] = d
R9  [t1] = 1001004c
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = c
R14 [t6] = 0
R15 [t7] = 2b
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 2b
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff16c
R30 [s8] = 0
R31 [ra] = 400144
```

Text

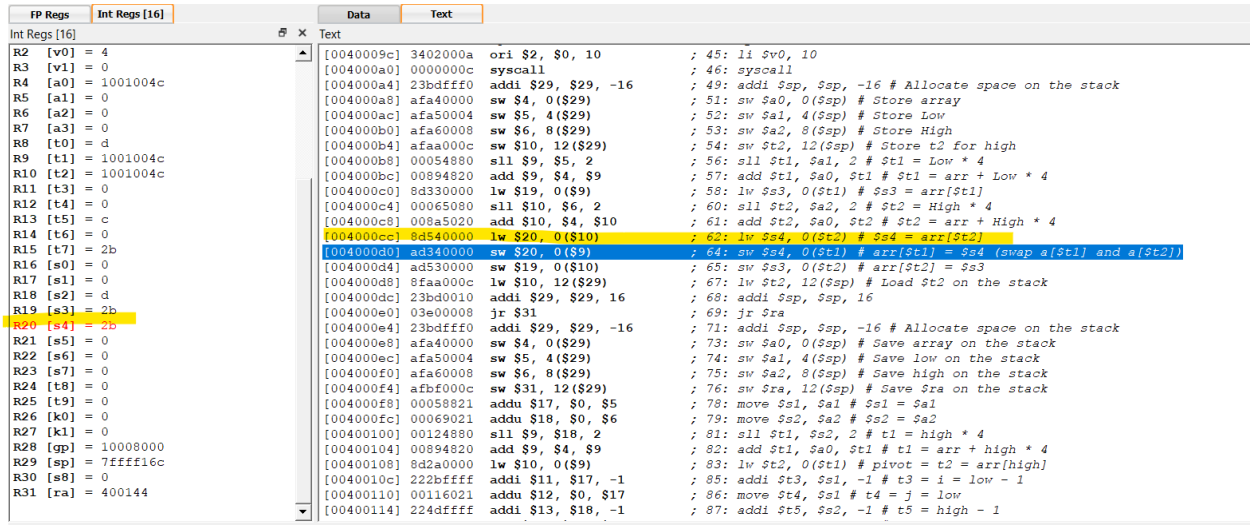
```
[0040009c] 3402000a  ori $2, $0, 10          ; 45: li $v0, 10
[004000a0] 0000000c  syscall                 ; 46: syscall
[004000a4] 23bdfff0  addi $29, $29, -16      ; 49: addi $sp, $sp, -16 # Allocate space on the stack
[004000a8] afa40000  sw $4, 0($29)           ; 51: sw $a0, 0($sp) # Store array
[004000ac] afa50004  sw $5, 4($29)           ; 52: sw $a1, 4($sp) # Store Low
[004000b0] afa60008  sw $6, 8($29)           ; 53: sw $a2, 8($sp) # Store High
[004000b4] afaa000c  sw $10, 12($29)         ; 54: sw $t2, 12($sp) # Store t2 for high
[004000b8] 00054880  sll $9, $5, 2           ; 56: sll $t1, $a1, 2 # $t1 = Low * 4
[004000bc] 00894820  add $9, $4, $9          ; 57: add $t1, $a0, $t1 # $t1 = arr + Low * 4
[004000c0] 8d330000  lw $19, 0($9)           ; 58: lw $s3, 0($t1) # $s3 = arr[$t1]
[004000c4] 00065080  sll $10, $6, 2          ; 60: sll $t2, $a2, 2 # $t2 = High * 4
[004000c8] 008a5020  add $10, $4, $10        ; 61: add $t2, $a0, $t2 # $t2 = arr + High * 4
[004000cc] 8d540000  lw $20, 0($10)          ; 62: lw $s4, 0($t2) # $s4 = arr[$t2]
[004000d0] ad340000  sw $20, 0($9)           ; 64: sw $s4, 0($t1) # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
[004000d4] ad530000  sw $19, 0($10)          ; 65: sw $s3, 0($t2) # arr[$t2] = $s3
[004000d8] 8faa000c  lw $10, 12($29)         ; 67: lw $t2, 12($sp) # Load $t2 on the stack
[004000dc] 23bd0010  addi $29, $29, 16       ; 68: addi $sp, $sp, 16
[004000e0] 03e00008  jr $31                  ; 69: jr $ra
[004000e4] 23bdfff0  addi $29, $29, -16      ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000  sw $4, 0($29)           ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004  sw $5, 4($29)           ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008  sw $6, 8($29)           ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c  sw $31, 12($29)         ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821  addu $17, $0, $5        ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021  addu $18, $0, $6        ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880  sll $9, $18, 2          ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820  add $9, $4, $9          ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000  lw $10, 0($9)           ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
[0040010c] 222bffff  addi $11, $17, -1       ; 85: addi $t3, $s1, -1 # t3 = i = low - 1
[00400110] 00116021  addu $12, $0, $17       ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff  addi $13, $18, -1       ; 87: addi $t5, $s2, -1 # t5 = high - 1
```


Int Regs [16]

```
R2  [v0] = 4
R3  [v1] = 0
R4  [a0] = 1001004c
R5  [a1] = 0
R6  [a2] = 0
R7  [a3] = 0
R8  [t0] = d
R9  [t1] = 1001004c
R10 [t2] = 1001004c
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = c
R14 [t6] = 0
R15 [t7] = 2b
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 2b
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff16c
R30 [s8] = 0
R31 [ra] = 400144
```

Text

```
[0040009c] 3402000a  ori $2, $0, 10          ; 45: li $v0, 10
[004000a0] 0000000c  syscall                 ; 46: syscall
[004000a4] 23bdfff0  addi $29, $29, -16      ; 49: addi $sp, $sp, -16 # Allocate space on the stack
[004000a8] afa40000  sw $4, 0($29)           ; 51: sw $a0, 0($sp) # Store array
[004000ac] afa50004  sw $5, 4($29)           ; 52: sw $a1, 4($sp) # Store Low
[004000b0] afa60008  sw $6, 8($29)           ; 53: sw $a2, 8($sp) # Store High
[004000b4] afaa000c  sw $10, 12($29)         ; 54: sw $t2, 12($sp) # Store t2 for high
[004000b8] 00054880  sll $9, $5, 2           ; 56: sll $t1, $a1, 2 # $t1 = Low * 4
[004000bc] 00894820  add $9, $4, $9          ; 57: add $t1, $a0, $t1 # $t1 = arr + Low * 4
[004000c0] 8d330000  lw $19, 0($9)           ; 58: lw $s3, 0($t1) # $s3 = arr[$t1]
[004000c4] 00065080  sll $10, $6, 2          ; 60: sll $t2, $a2, 2 # $t2 = High * 4
[004000c8] 008a5020  add $10, $4, $10        ; 61: add $t2, $a0, $t2 # $t2 = arr + High * 4
[004000cc] 8d540000  lw $20, 0($10)          ; 62: lw $s4, 0($t2) # $s4 = arr[$t2]
[004000d0] ad340000  sw $20, 0($9)           ; 64: sw $s4, 0($t1) # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
[004000d4] ad530000  sw $19, 0($10)          ; 65: sw $s3, 0($t2) # arr[$t2] = $s3
[004000d8] 8faa000c  lw $10, 12($29)         ; 67: lw $t2, 12($sp) # Load $t2 on the stack
[004000dc] 23bd0010  addi $29, $29, 16       ; 68: addi $sp, $sp, 16
[004000e0] 03e00008  jr $31                  ; 69: jr $ra
[004000e4] 23bdfff0  addi $29, $29, -16      ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000  sw $4, 0($29)           ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004  sw $5, 4($29)           ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008  sw $6, 8($29)           ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c  sw $31, 12($29)         ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821  addu $17, $0, $5        ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021  addu $18, $0, $6        ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880  sll $9, $18, 2          ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820  add $9, $4, $9          ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000  lw $10, 0($9)           ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
[0040010c] 222bffff  addi $11, $17, -1       ; 85: addi $t3, $s1, -1 # t3 = i = low - 1
[00400110] 00116021  addu $12, $0, $17       ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff  addi $13, $18, -1       ; 87: addi $t5, $s2, -1 # t5 = high - 1
```



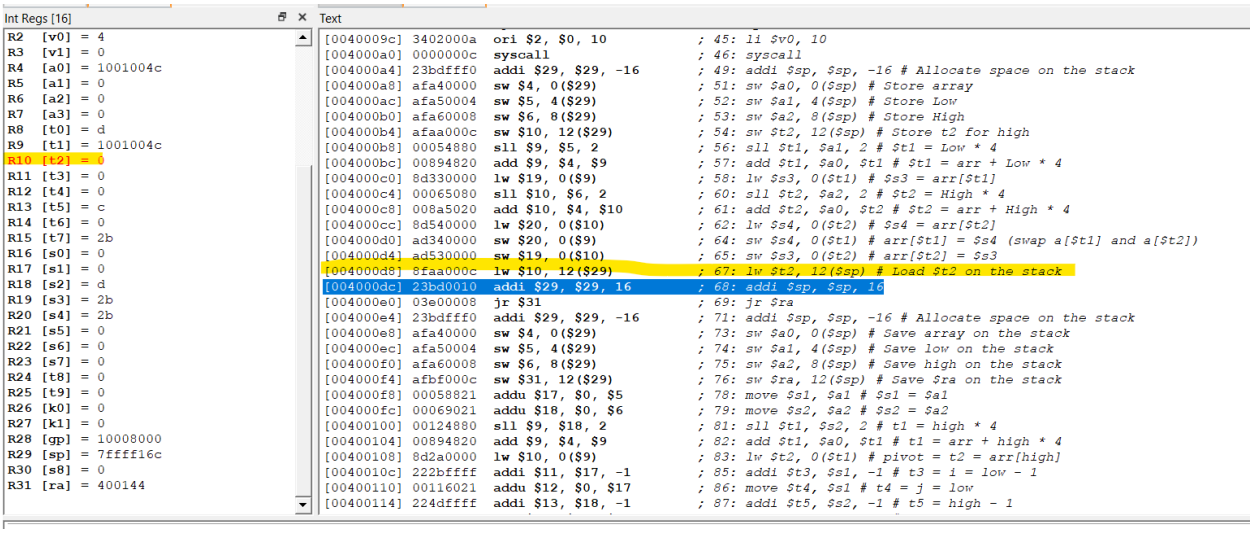
FP Regs | Int Regs [16] | Data | Text

Int Regs [16]

```
R2  [v0] = 4
R3  [v1] = 0
R4  [a0] = 1001004c
R5  [a1] = 0
R6  [a2] = 0
R7  [a3] = 0
R8  [t0] = d
R9  [t1] = 1001004c
R10 [t2] = 1001004c
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = c
R14 [t6] = 0
R15 [t7] = 2b
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 2b
R20 [s4] = 2b
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff16c
R30 [s8] = 0
R31 [ra] = 400144
```

Text

```
[0040009c] 3402000a  ori $2, $0, 10           ; 45: li $v0, 10
[004000a0] 0000000c  syscall                  ; 46: syscall
[004000a4] 23bdfff0  addi $29, $29, -16       ; 49: addi $sp, $sp, -16 # Allocate space on the stack
[004000a8] afa40000  sw $4, 0($29)            ; 51: sw $a0, 0($sp) # Store array
[004000ac] afa50004  sw $5, 4($29)            ; 52: sw $a1, 4($sp) # Store Low
[004000b0] afa60008  sw $6, 8($29)            ; 53: sw $a2, 8($sp) # Store High
[004000b4] afaa000c  sw $10, 12($29)          ; 54: sw $t2, 12($sp) # Store t2 for high
[004000b8] 00054880  sll $9, $5, 2            ; 56: sll $t1, $a1, 2 # $t1 = Low * 4
[004000bc] 00894820  add $9, $4, $9           ; 57: add $t1, $a0, $t1 # $t1 = arr + Low * 4
[004000c0] 8d330000  lw $19, 0($9)            ; 58: lw $s3, 0($t1) # $s3 = arr[$t1]
[004000c4] 00065080  sll $10, $6, 2           ; 60: sll $t2, $a2, 2 # $t2 = High * 4
[004000c8] 008a5020  add $10, $4, $10         ; 61: add $t2, $a0, $t2 # $t2 = arr + High * 4
[004000cc] 8d540000  lw $20, 0($10)           ; 62: lw $s4, 0($t2) # $s4 = arr[$t2]
[004000d0] ad340000  sw $20, 0($9)            ; 64: sw $s4, 0($t1) # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
[004000d4] ad530000  sw $19, 0($10)           ; 65: sw $s3, 0($t2) # arr[$t2] = $s3
[004000d8] 8faa000c  lw $10, 12($29)          ; 67: lw $t2, 12($sp) # Load $t2 on the stack
[004000dc] 23bd0010  addi $29, $29, 16        ; 68: addi $sp, $sp, 16
[004000e0] 03e00008  jr $31                   ; 69: jr $ra
[004000e4] 23bdfff0  addi $29, $29, -16       ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000  sw $4, 0($29)            ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004  sw $5, 4($29)            ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008  sw $6, 8($29)            ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c  sw $31, 12($29)          ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821  addu $17, $0, $5         ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021  addu $18, $0, $6         ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880  sll $9, $18, 2           ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820  add $9, $4, $9           ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000  lw $10, 0($9)            ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
[0040010c] 222bffff  addi $11, $17, -1        ; 85: addi $t3, $s1, -1 # t3 = i = low - 1
[00400110] 00116021  addu $12, $0, $17        ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff  addi $13, $18, -1        ; 87: addi $t5, $s2, -1 # t5 = high - 1
```

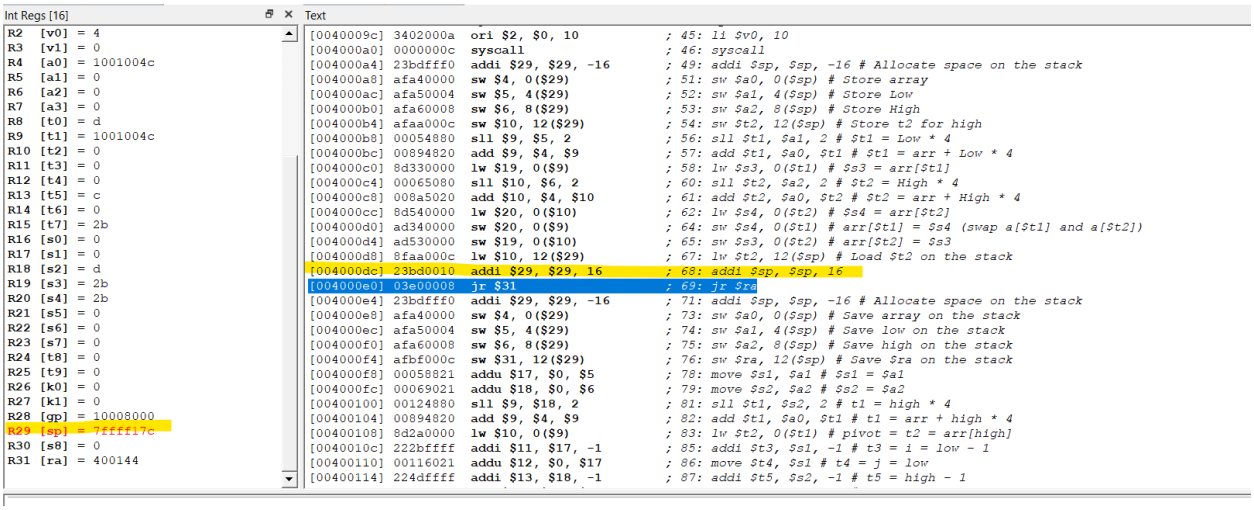


Int Regs [16]

```
R2  [v0] = 4
R3  [v1] = 0
R4  [a0] = 1001004c
R5  [a1] = 0
R6  [a2] = 0
R7  [a3] = 0
R8  [t0] = d
R9  [t1] = 1001004c
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = c
R14 [t6] = 0
R15 [t7] = 2b
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 2b
R20 [s4] = 2b
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff16c
R30 [s8] = 0
R31 [ra] = 400144
```

Text

```
[0040009c] 3402000a  ori $2, $0, 10           ; 45: li $v0, 10
[004000a0] 0000000c  syscall                  ; 46: syscall
[004000a4] 23bdfff0  addi $29, $29, -16       ; 49: addi $sp, $sp, -16 # Allocate space on the stack
[004000a8] afa40000  sw $4, 0($29)            ; 51: sw $a0, 0($sp) # Store array
[004000ac] afa50004  sw $5, 4($29)            ; 52: sw $a1, 4($sp) # Store Low
[004000b0] afa60008  sw $6, 8($29)            ; 53: sw $a2, 8($sp) # Store High
[004000b4] afaa000c  sw $10, 12($29)          ; 54: sw $t2, 12($sp) # Store t2 for high
[004000b8] 00054880  sll $9, $5, 2            ; 56: sll $t1, $a1, 2 # $t1 = Low * 4
[004000bc] 00894820  add $9, $4, $9           ; 57: add $t1, $a0, $t1 # $t1 = arr + Low * 4
[004000c0] 8d330000  lw $19, 0($9)            ; 58: lw $s3, 0($t1) # $s3 = arr[$t1]
[004000c4] 00065080  sll $10, $6, 2           ; 60: sll $t2, $a2, 2 # $t2 = High * 4
[004000c8] 008a5020  add $10, $4, $10         ; 61: add $t2, $a0, $t2 # $t2 = arr + High * 4
[004000cc] 8d540000  lw $20, 0($10)           ; 62: lw $s4, 0($t2) # $s4 = arr[$t2]
[004000d0] ad340000  sw $20, 0($9)            ; 64: sw $s4, 0($t1) # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
[004000d4] ad530000  sw $19, 0($10)           ; 65: sw $s3, 0($t2) # arr[$t2] = $s3
[004000d8] 8faa000c  lw $10, 12($29)          ; 67: lw $t2, 12($sp) # Load $t2 on the stack
[004000dc] 23bd0010  addi $29, $29, 16        ; 68: addi $sp, $sp, 16
[004000e0] 03e00008  jr $31                   ; 69: jr $ra
[004000e4] 23bdfff0  addi $29, $29, -16       ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000  sw $4, 0($29)            ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004  sw $5, 4($29)            ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008  sw $6, 8($29)            ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c  sw $31, 12($29)          ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821  addu $17, $0, $5         ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021  addu $18, $0, $6         ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880  sll $9, $18, 2           ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820  add $9, $4, $9           ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000  lw $10, 0($9)            ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
[0040010c] 222bffff  addi $11, $17, -1        ; 85: addi $t3, $s1, -1 # t3 = i = low - 1
[00400110] 00116021  addu $12, $0, $17        ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff  addi $13, $18, -1        ; 87: addi $t5, $s2, -1 # t5 = high - 1
```

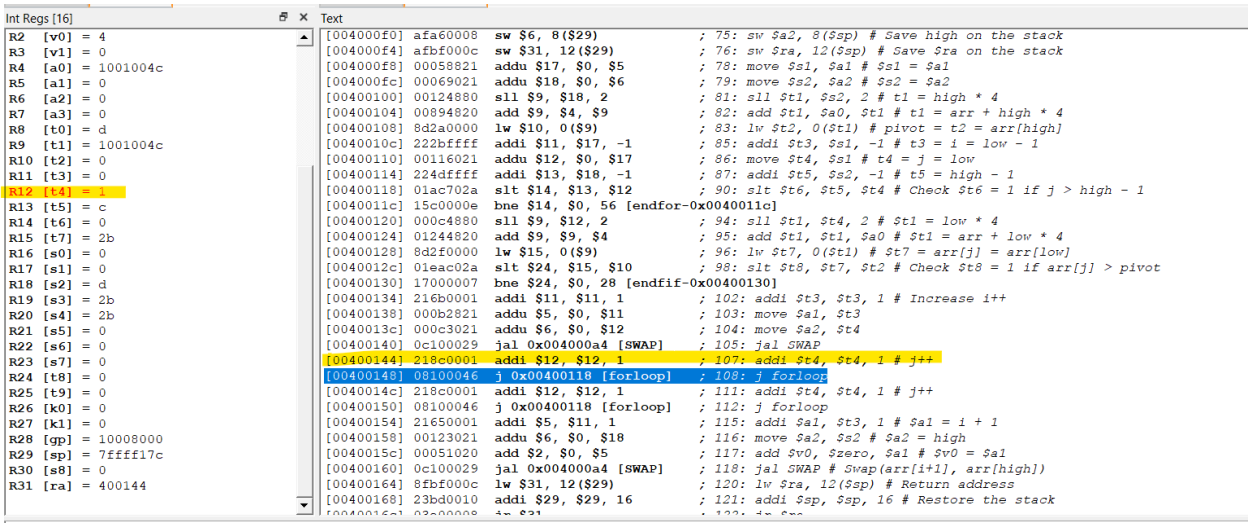


Int Regs [16]

```
R2  [v0] = 4
R3  [v1] = 0
R4  [a0] = 1001004c
R5  [a1] = 0
R6  [a2] = 0
R7  [a3] = 0
R8  [t0] = d
R9  [t1] = 1001004c
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = c
R14 [t6] = 0
R15 [t7] = 2b
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 2b
R20 [s4] = 2b
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff17c
R30 [s8] = 0
R31 [ra] = 400144
```

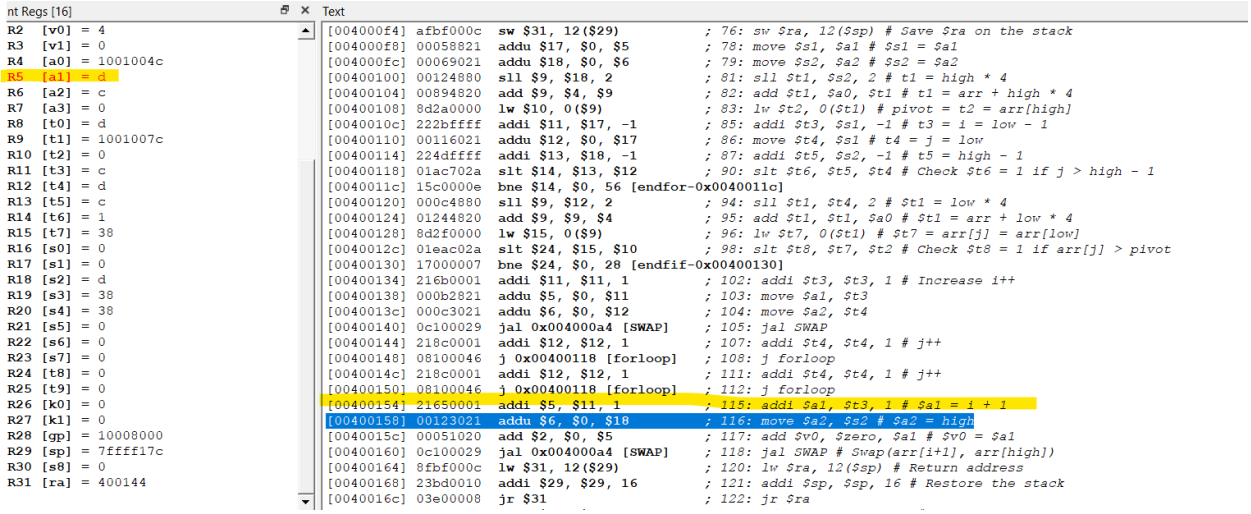
Text

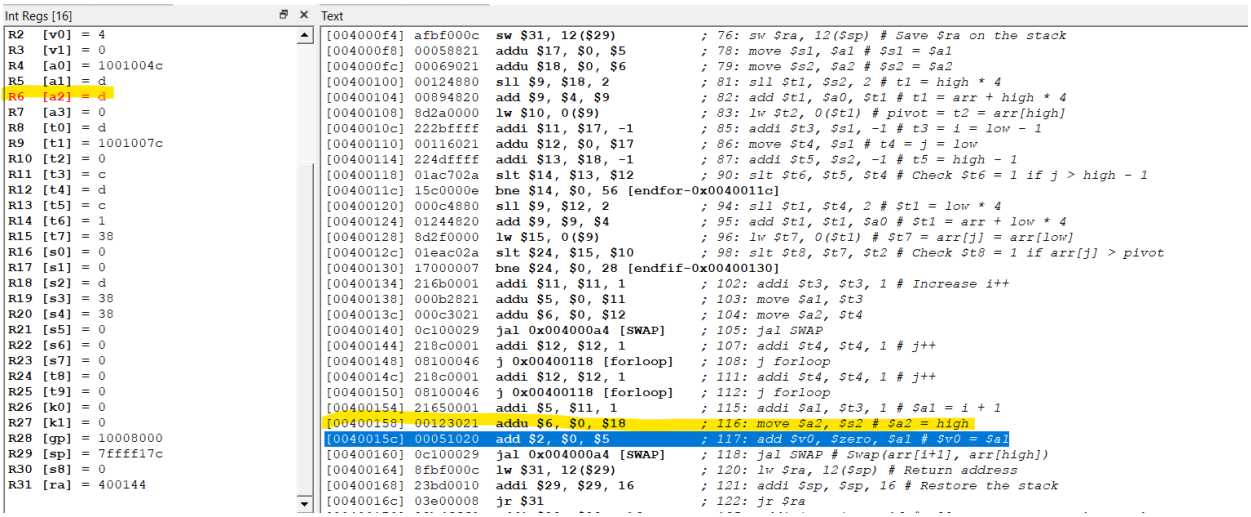
```
[0040009c] 3402000a  ori $2, $0, 10           ; 45: li $v0, 10
[004000a0] 0000000c  syscall                  ; 46: syscall
[004000a4] 23bdfff0  addi $29, $29, -16       ; 49: addi $sp, $sp, -16 # Allocate space on the stack
[004000a8] afa40000  sw $4, 0($29)            ; 51: sw $a0, 0($sp) # Store array
[004000ac] afa50004  sw $5, 4($29)            ; 52: sw $a1, 4($sp) # Store Low
[004000b0] afa60008  sw $6, 8($29)            ; 53: sw $a2, 8($sp) # Store High
[004000b4] afaa000c  sw $10, 12($29)          ; 54: sw $t2, 12($sp) # Store t2 for high
[004000b8] 00054880  sll $9, $5, 2            ; 56: sll $t1, $a1, 2 # $t1 = Low * 4
[004000bc] 00894820  add $9, $4, $9           ; 57: add $t1, $a0, $t1 # $t1 = arr + Low * 4
[004000c0] 8d330000  lw $19, 0($9)            ; 58: lw $s3, 0($t1) # $s3 = arr[$t1]
[004000c4] 00065080  sll $10, $6, 2           ; 60: sll $t2, $a2, 2 # $t2 = High * 4
[004000c8] 008a5020  add $10, $4, $10         ; 61: add $t2, $a0, $t2 # $t2 = arr + High * 4
[004000cc] 8d540000  lw $20, 0($10)           ; 62: lw $s4, 0($t2) # $s4 = arr[$t2]
[004000d0] ad340000  sw $20, 0($9)            ; 64: sw $s4, 0($t1) # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
[004000d4] ad530000  sw $19, 0($10)           ; 65: sw $s3, 0($t2) # arr[$t2] = $s3
[004000d8] 8faa000c  lw $10, 12($29)          ; 67: lw $t2, 12($sp) # Load $t2 on the stack
[004000dc] 23bd0010  addi $29, $29, 16        ; 68: addi $sp, $sp, 16
[004000e0] 03e00008  jr $31                   ; 69: jr $ra
[004000e4] 23bdfff0  addi $29, $29, -16       ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000  sw $4, 0($29)            ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004  sw $5, 4($29)            ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008  sw $6, 8($29)            ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c  sw $31, 12($29)          ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821  addu $17, $0, $5         ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021  addu $18, $0, $6         ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880  sll $9, $18, 2           ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820  add $9, $4, $9           ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000  lw $10, 0($9)            ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
[0040010c] 222bffff  addi $11, $17, -1        ; 85: addi $t3, $s1, -1 # t3 = i = low - 1
[00400110] 00116021  addu $12, $0, $17        ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff  addi $13, $18, -1        ; 87: addi $t5, $s2, -1 # t5 = high - 1
```

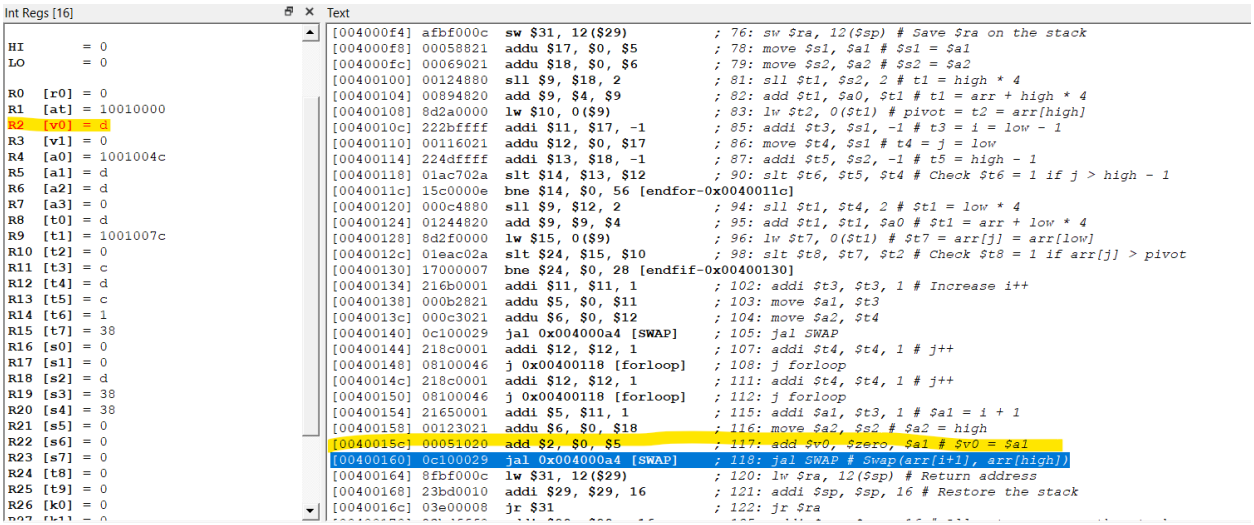


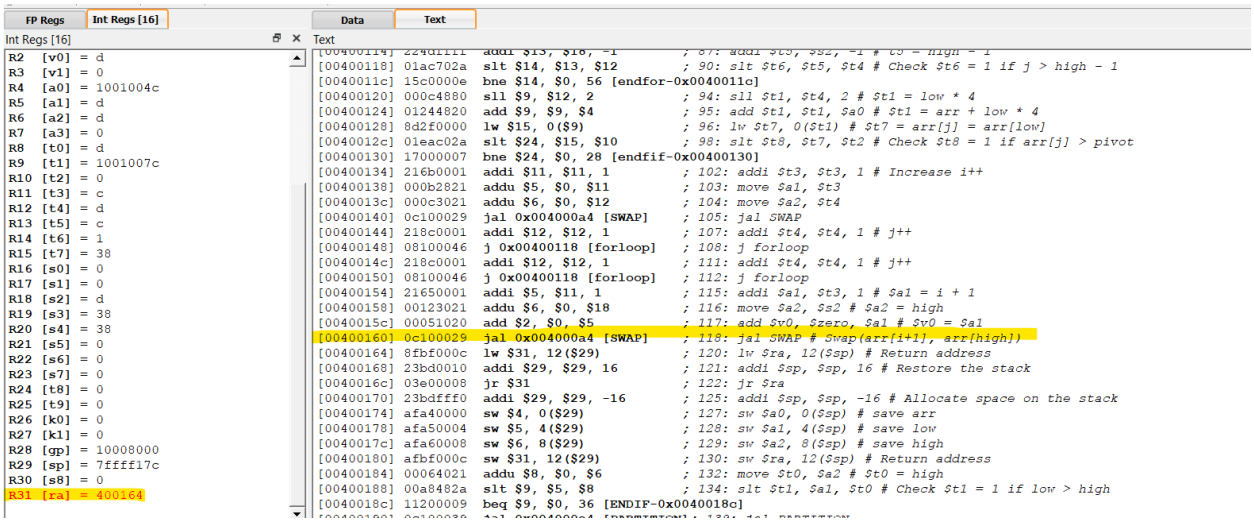
It will repeat the same steps until $t5 <= $t4 (or j <= high – 1)

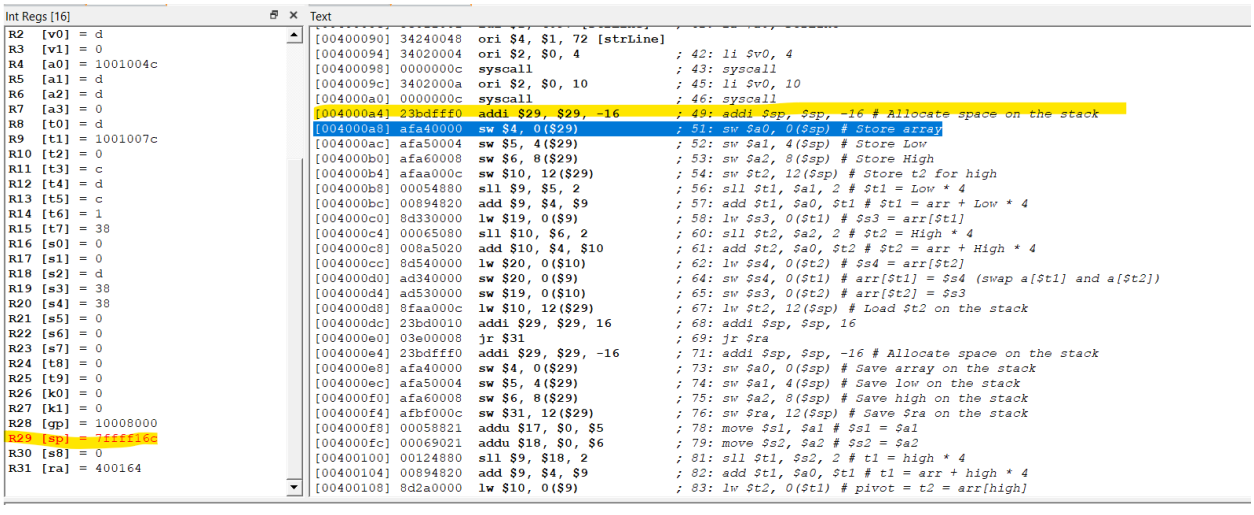
After that, It will jump to endfor if It does not satisfy the condition.






Int Regs [16]
Text
HI = 0
LO = 0
R0 [r0] = 0
R1 [at] = 10010000
R2 [v0] = d
R3 [v1] = 0
R4 [a0] = 1001004c
R5 [a1] = d
R6 [a2] = d
R7 [a3] = 0
R8 [t0] = d
R9 [t1] = 1001007c
R10 [t2] = 0
R11 [t3] = c
R12 [t4] = d
R13 [t5] = c
R14 [t6] = 1
R15 [t7] = 38
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 38
R20 [s4] = 38
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
[004000f4] afbf000c sw $31, 12($29) ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821 addu $17, $0, $5 ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021 addu $18, $0, $6 ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880 sll $9, $18, 2 ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820 add $9, $4, $9 ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000 lw $10, 0($9) ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
[0040010c] 222bffff addi $11, $17, -1 ; 85: addi $t3, $s1, -1 # t3 = i = low - 1
[00400110] 00116021 addu $12, $0, $17 ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff addi $13, $18, -1 ; 87: addi $t5, $s2, -1 # t5 = high - 1
[00400118] 01ac702a slt $14, $13, $12 ; 90: slt $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
[0040011c] 15c0000e bne $14, $0, 56 [endfor-0x0040011c]
[00400120] 000c4880 sll $9, $12, 2 ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400124] 01244820 add $9, $9, $4 ; 95: add $t1, $t1, $a0 # $t1 = arr + low * 4
[00400128] 8d2f0000 lw $15, 0($9) ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
[0040012c] 01eac02a slt $24, $15, $10 ; 98: slt $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400130] 17000007 bne $24, $0, 28 [endfif-0x00400130]
[00400134] 216b0001 addi $11, $11, 1 ; 102: addi $t3, $t3, 1 # Increase i++
[00400138] 000b2821 addu $5, $0, $11 ; 103: move $a1, $t3
[0040013c] 000c3021 addu $6, $0, $12 ; 104: move $a2, $t4
[00400140] 0c100029 jal 0x004000a4 [SWAP] ; 105: jal SWAP
[00400144] 218c0001 addi $12, $12, 1 ; 107: addi $t4, $t4, 1 # j++
[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
[0040014c] 218c0001 addi $12, $12, 1 ; 111: addi $t4, $t4, 1 # j++
[00400150] 08100046 j 0x00400118 [forloop] ; 112: j forloop
[00400154] 21650001 addi $5, $11, 1 ; 115: addi $a1, $t3, 1 # $a1 = i + 1
[00400158] 00123021 addu $6, $0, $18 ; 116: move $a2, $s2 # $a2 = high
[0040015c] 00051020 add $2, $0, $5 ; 117: add $v0, $zero, $a1 # $v0 = $a1
[00400160] 0c100029 jal 0x004000a4 [SWAP] ; 118: jal SWAP # Swap(arr[i+1], arr[high])
[00400164] 8fbf000c lw $31, 12($29) ; 120: lw $ra, 12($sp) # Return address
[00400168] 23bd0010 addi $29, $29, 16 ; 121: addi $sp, $sp, 16 # Restore the stack
[0040016c] 03e00008 jr $31 ; 122: jr $ra


FP Regs
Int Regs [16]
Data
Text
Int Regs [16]
Text
R2 [v0] = d
R3 [v1] = 0
R4 [a0] = 1001004c
R5 [a1] = d
R6 [a2] = d
R7 [a3] = 0
R8 [t0] = d
R9 [t1] = 1001007c
R10 [t2] = 0
R11 [t3] = c
R12 [t4] = d
R13 [t5] = c
R14 [t6] = 1
R15 [t7] = 38
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 38
R20 [s4] = 38
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff17c
R30 [s8] = 0
R31 [ra] = 400164
[00400118] 01ac702a slt $14, $13, $12 ; 90: slt $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
[0040011c] 15c0000e bne $14, $0, 56 [endfor-0x0040011c]
[00400120] 000c4880 sll $9, $12, 2 ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400124] 01244820 add $9, $9, $4 ; 95: add $t1, $t1, $a0 # $t1 = arr + low * 4
[00400128] 8d2f0000 lw $15, 0($9) ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
[0040012c] 01eac02a slt $24, $15, $10 ; 98: slt $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400130] 17000007 bne $24, $0, 28 [endfif-0x00400130]
[00400134] 216b0001 addi $11, $11, 1 ; 102: addi $t3, $t3, 1 # Increase i++
[00400138] 000b2821 addu $5, $0, $11 ; 103: move $a1, $t3
[0040013c] 000c3021 addu $6, $0, $12 ; 104: move $a2, $t4
[00400140] 0c100029 jal 0x004000a4 [SWAP] ; 105: jal SWAP
[00400144] 218c0001 addi $12, $12, 1 ; 107: addi $t4, $t4, 1 # j++
[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
[0040014c] 218c0001 addi $12, $12, 1 ; 111: addi $t4, $t4, 1 # j++
[00400150] 08100046 j 0x00400118 [forloop] ; 112: j forloop
[00400154] 21650001 addi $5, $11, 1 ; 115: addi $a1, $t3, 1 # $a1 = i + 1
[00400158] 00123021 addu $6, $0, $18 ; 116: move $a2, $s2 # $a2 = high
[0040015c] 00051020 add $2, $0, $5 ; 117: add $v0, $zero, $a1 # $v0 = $a1
[00400160] 0c100029 jal 0x004000a4 [SWAP] ; 118: jal SWAP # Swap(arr[i+1], arr[high])
[00400164] 8fbf000c lw $31, 12($29) ; 120: lw $ra, 12($sp) # Return address
[00400168] 23bd0010 addi $29, $29, 16 ; 121: addi $sp, $sp, 16 # Restore the stack
[0040016c] 03e00008 jr $31 ; 122: jr $ra
[00400170] 23bdfff0 addi $29, $29, -16 ; 125: addi $sp, $sp, -16 # Allocate space on the stack
[00400174] afa40000 sw $4, 0($29) ; 127: sw $a0, 0($sp) # save arr
[00400178] afa50004 sw $5, 4($29) ; 128: sw $a1, 4($sp) # save low
[0040017c] afa60008 sw $6, 8($29) ; 129: sw $a2, 8($sp) # save high
[00400180] afbf000c sw $31, 12($29) ; 130: sw $ra, 12($sp) # Return address
[00400184] 00064021 addu $8, $0, $6 ; 132: move $t0, $a2 # $t0 = high
[00400188] 00a8482a slt $9, $5, $8 ; 134: slt $t1, $a1, $t0 # Check $t1 = 1 if low > high
[0040018c] 11200009 beq $9, $0, 36 [ENDIF-0x0040018c]


Int Regs [16]
Text
R2 [v0] = d
R3 [v1] = 0
R4 [a0] = 1001004c
R5 [a1] = d
R6 [a2] = d
R7 [a3] = 0
R8 [t0] = d
R9 [t1] = 1001007c
R10 [t2] = 0
R11 [t3] = c
R12 [t4] = d
R13 [t5] = c
R14 [t6] = 1
R15 [t7] = 38
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 38
R20 [s4] = 38
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff16c
R30 [s8] = 0
R31 [ra] = 400164
[00400090] 34240048 ori $4, $1, 72 [strLine]
[00400094] 34020004 ori $2, $0, 4 ; 42: li $v0, 4
[00400098] 0000000c syscall ; 43: syscall
[0040009c] 3402000a ori $2, $0, 10 ; 45: li $v0, 10
[004000a0] 0000000c syscall ; 46: syscall
[004000a4] 23bdfff0 addi $29, $29, -16 ; 49: addi $sp, $sp, -16 # Allocate space on the stack
[004000a8] afa40000 sw $4, 0($29) ; 51: sw $a0, 0($sp) # Store array
[004000ac] afa50004 sw $5, 4($29) ; 52: sw $a1, 4($sp) # Store Low
[004000b0] afa60008 sw $6, 8($29) ; 53: sw $a2, 8($sp) # Store High
[004000b4] afaa000c sw $10, 12($29) ; 54: sw $t2, 12($sp) # Store t2 for high
[004000b8] 00054880 sll $9, $5, 2 ; 56: sll $t1, $a1, 2 # $t1 = Low * 4
[004000bc] 00894820 add $9, $4, $9 ; 57: add $t1, $a0, $t1 # $t1 = arr + Low * 4
[004000c0] 8d330000 lw $19, 0($9) ; 58: lw $s3, 0($t1) # $s3 = arr[$t1]
[004000c4] 00065080 sll $10, $6, 2 ; 60: sll $t2, $a2, 2 # $t2 = High * 4
[004000c8] 008a5020 add $10, $4, $10 ; 61: add $t2, $a0, $t2 # $t2 = arr + High * 4
[004000cc] 8d540000 lw $20, 0($10) ; 62: lw $s4, 0($t2) # $s4 = arr[$t2]
[004000d0] ad340000 sw $20, 0($9) ; 64: sw $s4, 0($t1) # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
[004000d4] ad530000 sw $19, 0($10) ; 65: sw $s3, 0($t2) # arr[$t2] = $s3
[004000d8] 8faa000c lw $10, 12($29) ; 67: lw $t2, 12($sp) # Load $t2 on the stack
[004000dc] 23bd0010 addi $29, $29, 16 ; 68: addi $sp, $sp, 16
[004000e0] 03e00008 jr $31 ; 69: jr $ra
[004000e4] 23bdfff0 addi $29, $29, -16 ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000 sw $4, 0($29) ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004 sw $5, 4($29) ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008 sw $6, 8($29) ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c sw $31, 12($29) ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821 addu $17, $0, $5 ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021 addu $18, $0, $6 ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880 sll $9, $18, 2 ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820 add $9, $4, $9 ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000 lw $10, 0($9) ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]

Int Regs [16]

```
R2  [v0] = d
R3  [v1] = 0
R4  [a0] = 1001004c
R5  [a1] = d
R6  [a2] = d
R7  [a3] = 0
R8  [t0] = d
R9  [t1] = 34
R10 [t2] = 0
R11 [t3] = c
R12 [t4] = d
R13 [t5] = c
R14 [t6] = 1
R15 [t7] = 38
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 38
R20 [s4] = 38
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff16c
R30 [s8] = 0
R31 [ra] = 400164
```

Text

```
[00400090] 34240048  ori $4, $1, 72 [strLine]
[00400094] 34020004  ori $2, $0, 4            ; 42: li $v0, 4
[00400098] 0000000c  syscall                  ; 43: syscall
[0040009c] 3402000a  ori $2, $0, 10           ; 45: li $v0, 10
[004000a0] 0000000c  syscall                  ; 46: syscall
[004000a4] 23bdfff0  addi $29, $29, -16       ; 49: addi $sp, $sp, -16 # Allocate space on the stack
[004000a8] afa40000  sw $4, 0($29)            ; 51: sw $a0, 0($sp) # Store array
[004000ac] afa50004  sw $5, 4($29)            ; 52: sw $a1, 4($sp) # Store Low
[004000b0] afa60008  sw $6, 8($29)            ; 53: sw $a2, 8($sp) # Store High
[004000b4] afaa000c  sw $10, 12($29)          ; 54: sw $t2, 12($sp) # Store t2 for high
[004000b8] 00054880  sll $9, $5, 2            ; 56: sll $t1, $a1, 2 # $t1 = Low * 4
[004000bc] 00894820  add $9, $4, $9           ; 57: add $t1, $a0, $t1 # $t1 = arr + Low * 4
[004000c0] 8d330000  lw $19, 0($9)            ; 58: lw $s3, 0($t1) # $s3 = arr[$t1]
[004000c4] 00065080  sll $10, $6, 2           ; 60: sll $t2, $a2, 2 # $t2 = High * 4
[004000c8] 008a5020  add $10, $4, $10         ; 61: add $t2, $a0, $t2 # $t2 = arr + High * 4
[004000cc] 8d540000  lw $20, 0($10)           ; 62: lw $s4, 0($t2) # $s4 = arr[$t2]
[004000d0] ad340000  sw $20, 0($9)            ; 64: sw $s4, 0($t1) # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
[004000d4] ad530000  sw $19, 0($10)           ; 65: sw $s3, 0($t2) # arr[$t2] = $s3
[004000d8] 8faa000c  lw $10, 12($29)          ; 67: lw $t2, 12($sp) # Load $t2 on the stack
[004000dc] 23bd0010  addi $29, $29, 16        ; 68: addi $sp, $sp, 16
[004000e0] 03e00008  jr $31                   ; 69: jr $ra
[004000e4] 23bdfff0  addi $29, $29, -16       ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000  sw $4, 0($29)            ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004  sw $5, 4($29)            ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008  sw $6, 8($29)            ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c  sw $31, 12($29)          ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821  addu $17, $0, $5         ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021  addu $18, $0, $6         ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880  sll $9, $18, 2           ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820  add $9, $4, $9           ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000  lw $10, 0($9)            ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
```

Int Regs [16]

```
R2  [v0] = d
R3  [v1] = 0
R4  [a0] = 1001004c
R5  [a1] = d
R6  [a2] = d
R7  [a3] = 0
R8  [t0] = d
R9  [t1] = 10010080
R10 [t2] = 0
R11 [t3] = c
R12 [t4] = d
R13 [t5] = c
R14 [t6] = 1
R15 [t7] = 38
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 38
R20 [s4] = 38
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff16c
R30 [s8] = 0
R31 [ra] = 400164
```

Text

```
[00400090] 34240048  ori $4, $1, 72 [strLine]
[00400094] 34020004  ori $2, $0, 4            ; 42: li $v0, 4
[00400098] 0000000c  syscall                  ; 43: syscall
[0040009c] 3402000a  ori $2, $0, 10           ; 45: li $v0, 10
[004000a0] 0000000c  syscall                  ; 46: syscall
[004000a4] 23bdfff0  addi $29, $29, -16       ; 49: addi $sp, $sp, -16 # Allocate space on the stack
[004000a8] afa40000  sw $4, 0($29)            ; 51: sw $a0, 0($sp) # Store array
[004000ac] afa50004  sw $5, 4($29)            ; 52: sw $a1, 4($sp) # Store Low
[004000b0] afa60008  sw $6, 8($29)            ; 53: sw $a2, 8($sp) # Store High
[004000b4] afaa000c  sw $10, 12($29)          ; 54: sw $t2, 12($sp) # Store t2 for high
[004000b8] 00054880  sll $9, $5, 2            ; 56: sll $t1, $a1, 2 # $t1 = Low * 4
[004000bc] 00894820  add $9, $4, $9           ; 57: add $t1, $a0, $t1 # $t1 = arr + Low * 4
[004000c0] 8d330000  lw $19, 0($9)            ; 58: lw $s3, 0($t1) # $s3 = arr[$t1]
[004000c4] 00065080  sll $10, $6, 2           ; 60: sll $t2, $a2, 2 # $t2 = High * 4
[004000c8] 008a5020  add $10, $4, $10         ; 61: add $t2, $a0, $t2 # $t2 = arr + High * 4
[004000cc] 8d540000  lw $20, 0($10)           ; 62: lw $s4, 0($t2) # $s4 = arr[$t2]
[004000d0] ad340000  sw $20, 0($9)            ; 64: sw $s4, 0($t1) # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
[004000d4] ad530000  sw $19, 0($10)           ; 65: sw $s3, 0($t2) # arr[$t2] = $s3
[004000d8] 8faa000c  lw $10, 12($29)          ; 67: lw $t2, 12($sp) # Load $t2 on the stack
[004000dc] 23bd0010  addi $29, $29, 16        ; 68: addi $sp, $sp, 16
[004000e0] 03e00008  jr $31                   ; 69: jr $ra
[004000e4] 23bdfff0  addi $29, $29, -16       ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000  sw $4, 0($29)            ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004  sw $5, 4($29)            ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008  sw $6, 8($29)            ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c  sw $31, 12($29)          ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821  addu $17, $0, $5         ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021  addu $18, $0, $6         ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880  sll $9, $18, 2           ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820  add $9, $4, $9           ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000  lw $10, 0($9)            ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
```

Int Regs [16]

```
R2  [v0] = d
R3  [v1] = 0
R4  [a0] = 1001004c
R5  [a1] = d
R6  [a2] = d
R7  [a3] = 0
R8  [t0] = d
R9  [t1] = 10010080
R10 [t2] = 0
R11 [t3] = c
R12 [t4] = d
R13 [t5] = c
R14 [t6] = 1
R15 [t7] = 38
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 0
R20 [s4] = 38
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff16c
R30 [s8] = 0
R31 [ra] = 400164
```

Text

```
[00400090] 34240048  ori $4, $1, 72 [strLine]
[00400094] 34020004  ori $2, $0, 4            ; 42: li $v0, 4
[00400098] 0000000c  syscall                  ; 43: syscall
[0040009c] 3402000a  ori $2, $0, 10           ; 45: li $v0, 10
[004000a0] 0000000c  syscall                  ; 46: syscall
[004000a4] 23bdfff0  addi $29, $29, -16       ; 49: addi $sp, $sp, -16 # Allocate space on the stack
[004000a8] afa40000  sw $4, 0($29)            ; 51: sw $a0, 0($sp) # Store array
[004000ac] afa50004  sw $5, 4($29)            ; 52: sw $a1, 4($sp) # Store Low
[004000b0] afa60008  sw $6, 8($29)            ; 53: sw $a2, 8($sp) # Store High
[004000b4] afaa000c  sw $10, 12($29)          ; 54: sw $t2, 12($sp) # Store t2 for high
[004000b8] 00054880  sll $9, $5, 2            ; 56: sll $t1, $a1, 2 # $t1 = Low * 4
[004000bc] 00894820  add $9, $4, $9           ; 57: add $t1, $a0, $t1 # $t1 = arr + Low * 4
[004000c0] 8d330000  lw $19, 0($9)            ; 58: lw $s3, 0($t1) # $s3 = arr[$t1]
[004000c4] 00065080  sll $10, $6, 2           ; 60: sll $t2, $a2, 2 # $t2 = High * 4
[004000c8] 008a5020  add $10, $4, $10         ; 61: add $t2, $a0, $t2 # $t2 = arr + High * 4
[004000cc] 8d540000  lw $20, 0($10)           ; 62: lw $s4, 0($t2) # $s4 = arr[$t2]
[004000d0] ad340000  sw $20, 0($9)            ; 64: sw $s4, 0($t1) # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
[004000d4] ad530000  sw $19, 0($10)           ; 65: sw $s3, 0($t2) # arr[$t2] = $s3
[004000d8] 8faa000c  lw $10, 12($29)          ; 67: lw $t2, 12($sp) # Load $t2 on the stack
[004000dc] 23bd0010  addi $29, $29, 16        ; 68: addi $sp, $sp, 16
[004000e0] 03e00008  jr $31                   ; 69: jr $ra
[004000e4] 23bdfff0  addi $29, $29, -16       ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000  sw $4, 0($29)            ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004  sw $5, 4($29)            ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008  sw $6, 8($29)            ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c  sw $31, 12($29)          ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821  addu $17, $0, $5         ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021  addu $18, $0, $6         ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880  sll $9, $18, 2           ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820  add $9, $4, $9           ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000  lw $10, 0($9)            ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
```

Int Regs [16]

```
R2  [v0] = d
R3  [v1] = 0
R4  [a0] = 1001004c
R5  [a1] = d
R6  [a2] = d
R7  [a3] = 0
R8  [t0] = d
R9  [t1] = 10010080
R10 [t2] = 34
R11 [t3] = c
R12 [t4] = d
R13 [t5] = c
R14 [t6] = 1
R15 [t7] = 38
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 0
R20 [s4] = 38
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff16c
R30 [s8] = 0
R31 [ra] = 400164
```

Text

```
[00400090] 34240048  ori $4, $1, 72 [strLine]
[00400094] 34020004  ori $2, $0, 4            ; 42: li $v0, 4
[00400098] 0000000c  syscall                  ; 43: syscall
[0040009c] 3402000a  ori $2, $0, 10           ; 45: li $v0, 10
[004000a0] 0000000c  syscall                  ; 46: syscall
[004000a4] 23bdfff0  addi $29, $29, -16       ; 49: addi $sp, $sp, -16 # Allocate space on the stack
[004000a8] afa40000  sw $4, 0($29)            ; 51: sw $a0, 0($sp) # Store array
[004000ac] afa50004  sw $5, 4($29)            ; 52: sw $a1, 4($sp) # Store Low
[004000b0] afa60008  sw $6, 8($29)            ; 53: sw $a2, 8($sp) # Store High
[004000b4] afaa000c  sw $10, 12($29)          ; 54: sw $t2, 12($sp) # Store t2 for high
[004000b8] 00054880  sll $9, $5, 2            ; 56: sll $t1, $a1, 2 # $t1 = Low * 4
[004000bc] 00894820  add $9, $4, $9           ; 57: add $t1, $a0, $t1 # $t1 = arr + Low * 4
[004000c0] 8d330000  lw $19, 0($9)            ; 58: lw $s3, 0($t1) # $s3 = arr[$t1]
[004000c4] 00065080  sll $10, $6, 2           ; 60: sll $t2, $a2, 2 # $t2 = High * 4
[004000c8] 008a5020  add $10, $4, $10         ; 61: add $t2, $a0, $t2 # $t2 = arr + High * 4
[004000cc] 8d540000  lw $20, 0($10)           ; 62: lw $s4, 0($t2) # $s4 = arr[$t2]
[004000d0] ad340000  sw $20, 0($9)            ; 64: sw $s4, 0($t1) # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
[004000d4] ad530000  sw $19, 0($10)           ; 65: sw $s3, 0($t2) # arr[$t2] = $s3
[004000d8] 8faa000c  lw $10, 12($29)          ; 67: lw $t2, 12($sp) # Load $t2 on the stack
[004000dc] 23bd0010  addi $29, $29, 16        ; 68: addi $sp, $sp, 16
[004000e0] 03e00008  jr $31                   ; 69: jr $ra
[004000e4] 23bdfff0  addi $29, $29, -16       ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000  sw $4, 0($29)            ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004  sw $5, 4($29)            ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008  sw $6, 8($29)            ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c  sw $31, 12($29)          ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821  addu $17, $0, $5         ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021  addu $18, $0, $6         ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880  sll $9, $18, 2           ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820  add $9, $4, $9           ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000  lw $10, 0($9)            ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
```

Int Regs [16]

```
R2  [v0] = d
R3  [v1] = 0
R4  [a0] = 1001004c
R5  [a1] = d
R6  [a2] = d
R7  [a3] = 0
R8  [t0] = d
R9  [t1] = 10010080
R10 [t2] = 10010080
R11 [t3] = c
R12 [t4] = d
R13 [t5] = c
R14 [t6] = 1
R15 [t7] = 38
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 0
R20 [s4] = 38
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff16c
R30 [s8] = 0
R31 [ra] = 400164
```

Text

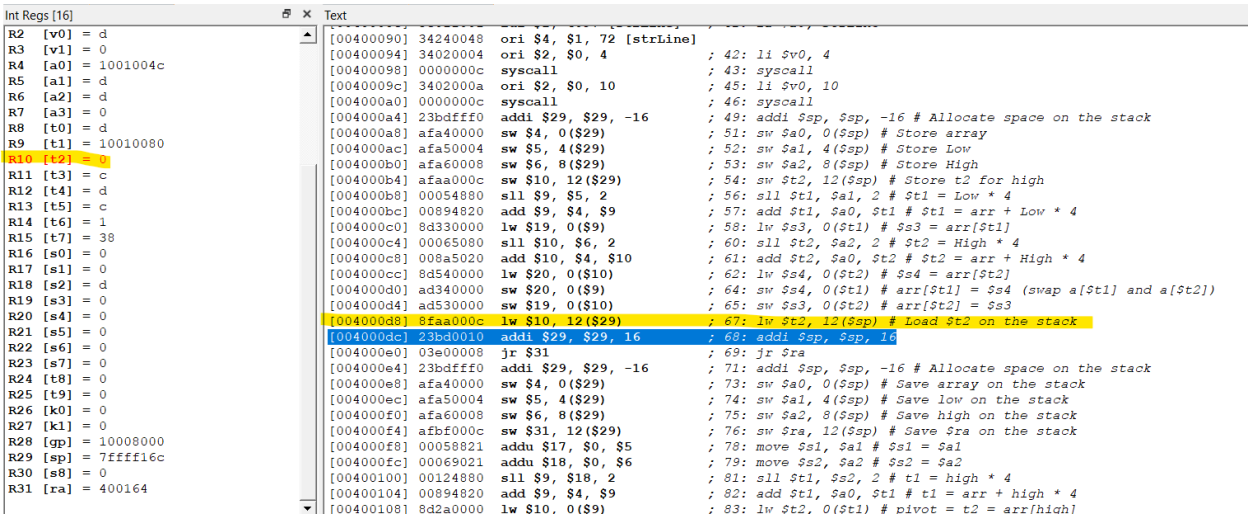
```
[00400090] 34240048  ori $4, $1, 72 [strLine]
[00400094] 34020004  ori $2, $0, 4            ; 42: li $v0, 4
[00400098] 0000000c  syscall                  ; 43: syscall
[0040009c] 3402000a  ori $2, $0, 10           ; 45: li $v0, 10
[004000a0] 0000000c  syscall                  ; 46: syscall
[004000a4] 23bdfff0  addi $29, $29, -16       ; 49: addi $sp, $sp, -16 # Allocate space on the stack
[004000a8] afa40000  sw $4, 0($29)            ; 51: sw $a0, 0($sp) # Store array
[004000ac] afa50004  sw $5, 4($29)            ; 52: sw $a1, 4($sp) # Store Low
[004000b0] afa60008  sw $6, 8($29)            ; 53: sw $a2, 8($sp) # Store High
[004000b4] afaa000c  sw $10, 12($29)          ; 54: sw $t2, 12($sp) # Store t2 for high
[004000b8] 00054880  sll $9, $5, 2            ; 56: sll $t1, $a1, 2 # $t1 = Low * 4
[004000bc] 00894820  add $9, $4, $9           ; 57: add $t1, $a0, $t1 # $t1 = arr + Low * 4
[004000c0] 8d330000  lw $19, 0($9)            ; 58: lw $s3, 0($t1) # $s3 = arr[$t1]
[004000c4] 00065080  sll $10, $6, 2           ; 60: sll $t2, $a2, 2 # $t2 = High * 4
[004000c8] 008a5020  add $10, $4, $10         ; 61: add $t2, $a0, $t2 # $t2 = arr + High * 4
[004000cc] 8d540000  lw $20, 0($10)           ; 62: lw $s4, 0($t2) # $s4 = arr[$t2]
[004000d0] ad340000  sw $20, 0($9)            ; 64: sw $s4, 0($t1) # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
[004000d4] ad530000  sw $19, 0($10)           ; 65: sw $s3, 0($t2) # arr[$t2] = $s3
[004000d8] 8faa000c  lw $10, 12($29)          ; 67: lw $t2, 12($sp) # Load $t2 on the stack
[004000dc] 23bd0010  addi $29, $29, 16        ; 68: addi $sp, $sp, 16
[004000e0] 03e00008  jr $31                   ; 69: jr $ra
[004000e4] 23bdfff0  addi $29, $29, -16       ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000  sw $4, 0($29)            ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004  sw $5, 4($29)            ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008  sw $6, 8($29)            ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c  sw $31, 12($29)          ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821  addu $17, $0, $5         ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021  addu $18, $0, $6         ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880  sll $9, $18, 2           ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820  add $9, $4, $9           ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000  lw $10, 0($9)            ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
```

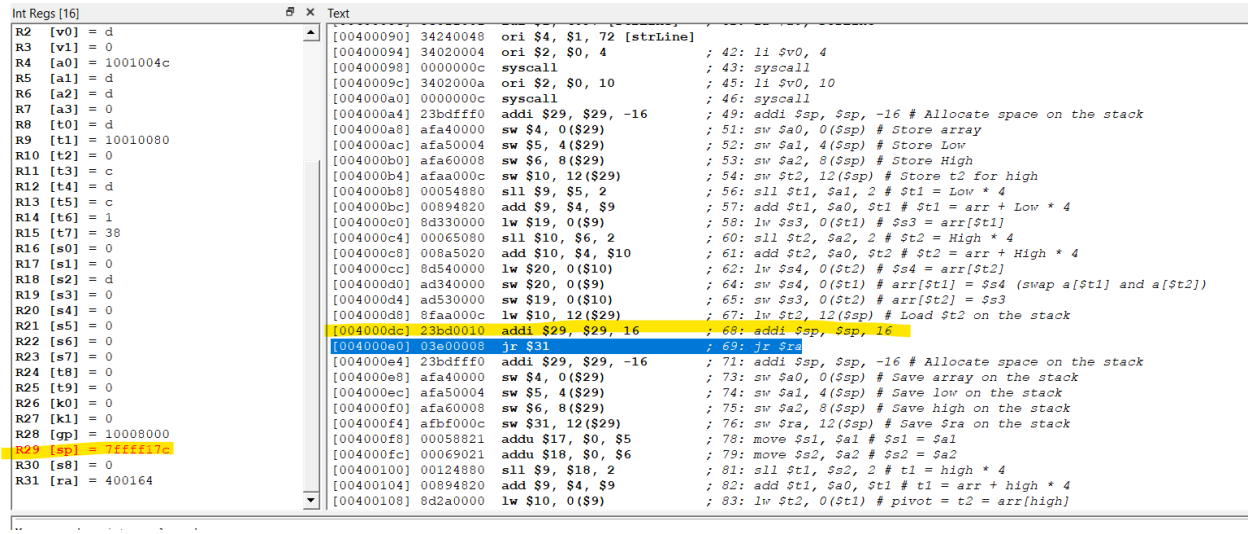
Int Regs [16]

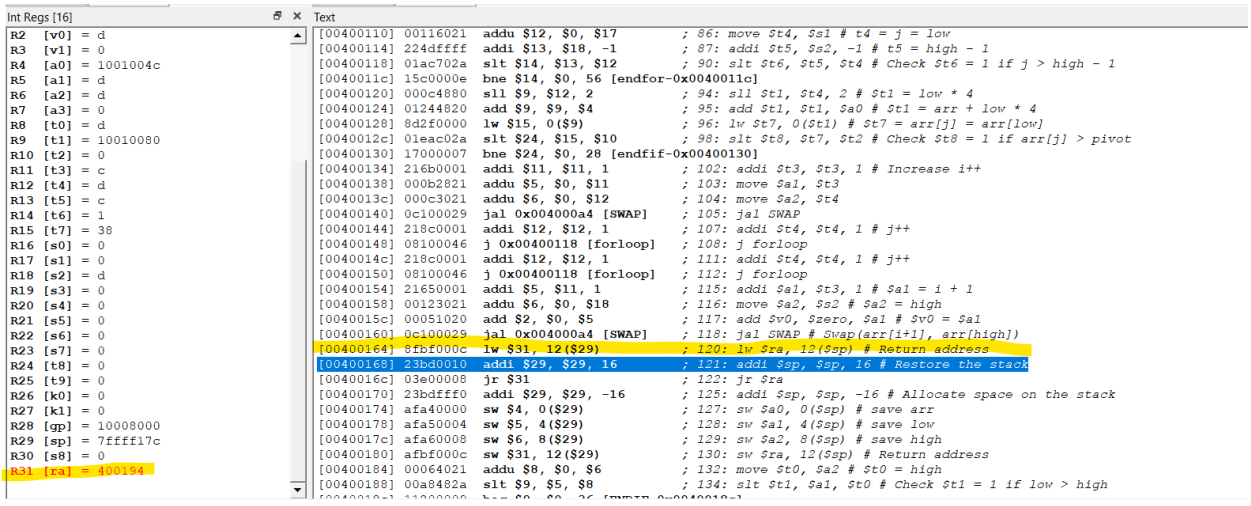
```
R2  [v0] = d
R3  [v1] = 0
R4  [a0] = 1001004c
R5  [a1] = d
R6  [a2] = d
R7  [a3] = 0
R8  [t0] = d
R9  [t1] = 10010080
R10 [t2] = 10010080
R11 [t3] = c
R12 [t4] = d
R13 [t5] = c
R14 [t6] = 1
R15 [t7] = 38
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff16c
R30 [s8] = 0
R31 [ra] = 400164
```

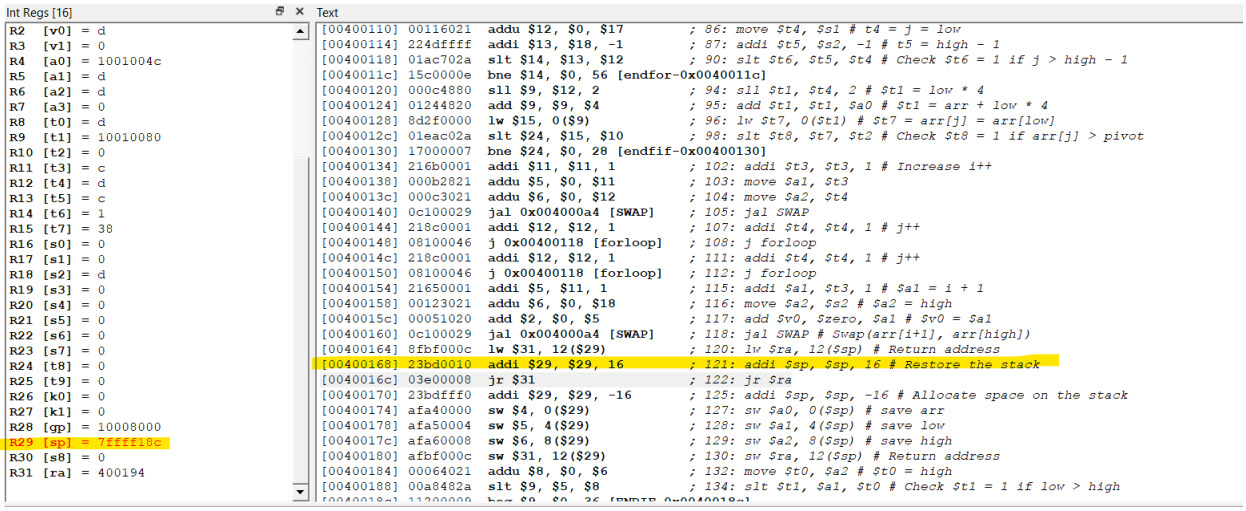
Text

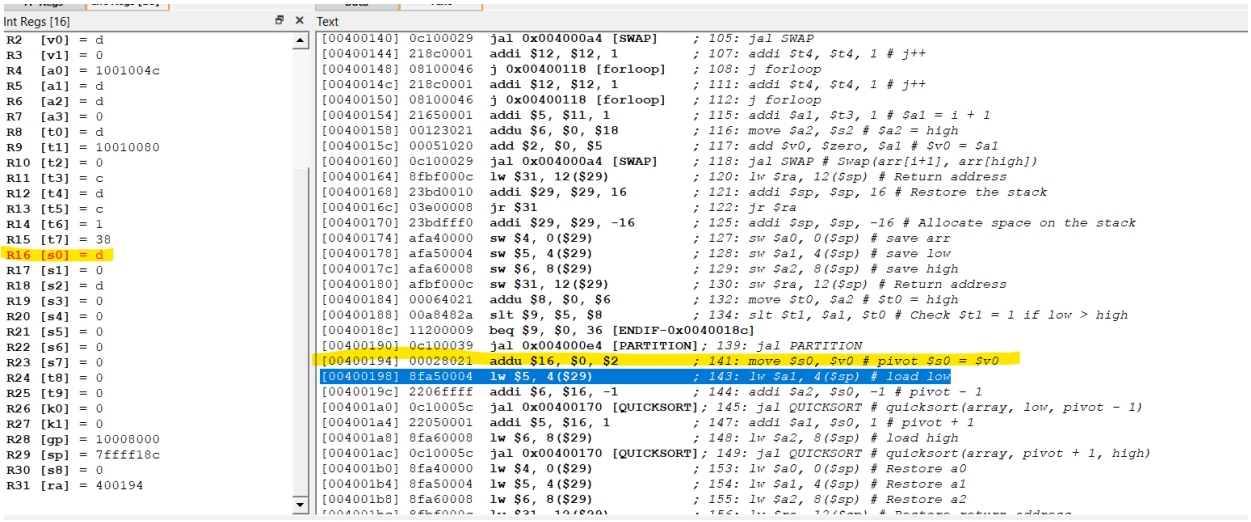
```
[00400090] 34240048  ori $4, $1, 72 [strLine]
[00400094] 34020004  ori $2, $0, 4            ; 42: li $v0, 4
[00400098] 0000000c  syscall                  ; 43: syscall
[0040009c] 3402000a  ori $2, $0, 10           ; 45: li $v0, 10
[004000a0] 0000000c  syscall                  ; 46: syscall
[004000a4] 23bdfff0  addi $29, $29, -16       ; 49: addi $sp, $sp, -16 # Allocate space on the stack
[004000a8] afa40000  sw $4, 0($29)            ; 51: sw $a0, 0($sp) # Store array
[004000ac] afa50004  sw $5, 4($29)            ; 52: sw $a1, 4($sp) # Store Low
[004000b0] afa60008  sw $6, 8($29)            ; 53: sw $a2, 8($sp) # Store High
[004000b4] afaa000c  sw $10, 12($29)          ; 54: sw $t2, 12($sp) # Store t2 for high
[004000b8] 00054880  sll $9, $5, 2            ; 56: sll $t1, $a1, 2 # $t1 = Low * 4
[004000bc] 00894820  add $9, $4, $9           ; 57: add $t1, $a0, $t1 # $t1 = arr + Low * 4
[004000c0] 8d330000  lw $19, 0($9)            ; 58: lw $s3, 0($t1) # $s3 = arr[$t1]
[004000c4] 00065080  sll $10, $6, 2           ; 60: sll $t2, $a2, 2 # $t2 = High * 4
[004000c8] 008a5020  add $10, $4, $10         ; 61: add $t2, $a0, $t2 # $t2 = arr + High * 4
[004000cc] 8d540000  lw $20, 0($10)           ; 62: lw $s4, 0($t2) # $s4 = arr[$t2]
[004000d0] ad340000  sw $20, 0($9)            ; 64: sw $s4, 0($t1) # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
[004000d4] ad530000  sw $19, 0($10)           ; 65: sw $s3, 0($t2) # arr[$t2] = $s3
[004000d8] 8faa000c  lw $10, 12($29)          ; 67: lw $t2, 12($sp) # Load $t2 on the stack
[004000dc] 23bd0010  addi $29, $29, 16        ; 68: addi $sp, $sp, 16
[004000e0] 03e00008  jr $31                   ; 69: jr $ra
[004000e4] 23bdfff0  addi $29, $29, -16       ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000  sw $4, 0($29)            ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004  sw $5, 4($29)            ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008  sw $6, 8($29)            ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c  sw $31, 12($29)          ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821  addu $17, $0, $5         ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021  addu $18, $0, $6         ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880  sll $9, $18, 2           ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820  add $9, $4, $9           ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000  lw $10, 0($9)            ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
```

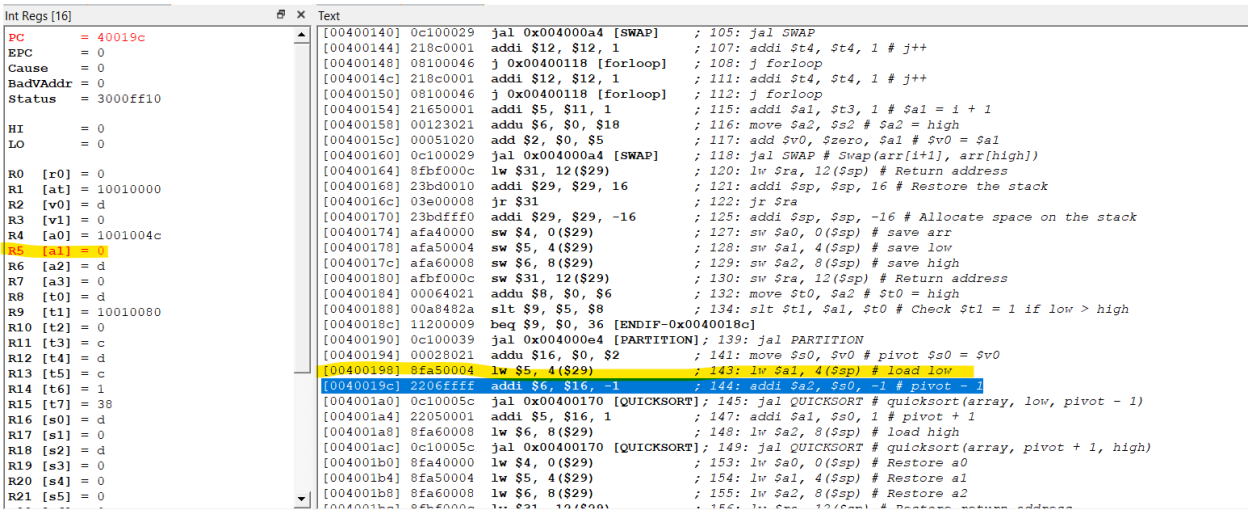

Int Regs [16]
R2 [v0] = d
R3 [v1] = 0
R4 [a0] = 1001004c
R5 [a1] = d
R6 [a2] = d
R7 [a3] = 0
R8 [t0] = d
R9 [t1] = 10010080
R10 [t2] = 0
R11 [t3] = c
R12 [t4] = d
R13 [t5] = c
R14 [t6] = 1
R15 [t7] = 38
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff16c
R30 [s8] = 0
R31 [ra] = 400164
Text
[00400090] 34240048 ori $4, $1, 72 [strLine]
[00400094] 34020004 ori $2, $0, 4 ; 42: li $v0, 4
[00400098] 0000000c syscall ; 43: syscall
[0040009c] 3402000a ori $2, $0, 10 ; 45: li $v0, 10
[004000a0] 0000000c syscall ; 46: syscall
[004000a4] 23bdfff0 addi $29, $29, -16 ; 49: addi $sp, $sp, -16 # Allocate space on the stack
[004000a8] afa40000 sw $4, 0($29) ; 51: sw $a0, 0($sp) # Store array
[004000ac] afa50004 sw $5, 4($29) ; 52: sw $a1, 4($sp) # Store Low
[004000b0] afa60008 sw $6, 8($29) ; 53: sw $a2, 8($sp) # Store High
[004000b4] afaa000c sw $10, 12($29) ; 54: sw $t2, 12($sp) # Store t2 for high
[004000b8] 00054880 sll $9, $5, 2 ; 56: sll $t1, $a1, 2 # $t1 = Low * 4
[004000bc] 00894820 add $9, $4, $9 ; 57: add $t1, $a0, $t1 # $t1 = arr + Low * 4
[004000c0] 8d330000 lw $19, 0($9) ; 58: lw $s3, 0($t1) # $s3 = arr[$t1]
[004000c4] 00065080 sll $10, $6, 2 ; 60: sll $t2, $a2, 2 # $t2 = High * 4
[004000c8] 008a5020 add $10, $4, $10 ; 61: add $t2, $a0, $t2 # $t2 = arr + High * 4
[004000cc] 8d540000 lw $20, 0($10) ; 62: lw $s4, 0($t2) # $s4 = arr[$t2]
[004000d0] ad340000 sw $20, 0($9) ; 64: sw $s4, 0($t1) # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
[004000d4] ad530000 sw $19, 0($10) ; 65: sw $s3, 0($t2) # arr[$t2] = $s3
[004000d8] 8faa000c lw $10, 12($29) ; 67: lw $t2, 12($sp) # Load $t2 on the stack
[004000dc] 23bd0010 addi $29, $29, 16 ; 68: addi $sp, $sp, 16
[004000e0] 03e00008 jr $31 ; 69: jr $ra
[004000e4] 23bdfff0 addi $29, $29, -16 ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000 sw $4, 0($29) ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004 sw $5, 4($29) ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008 sw $6, 8($29) ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c sw $31, 12($29) ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821 addu $17, $0, $5 ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021 addu $18, $0, $6 ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880 sll $9, $18, 2 ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820 add $9, $4, $9 ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000 lw $10, 0($9) ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]

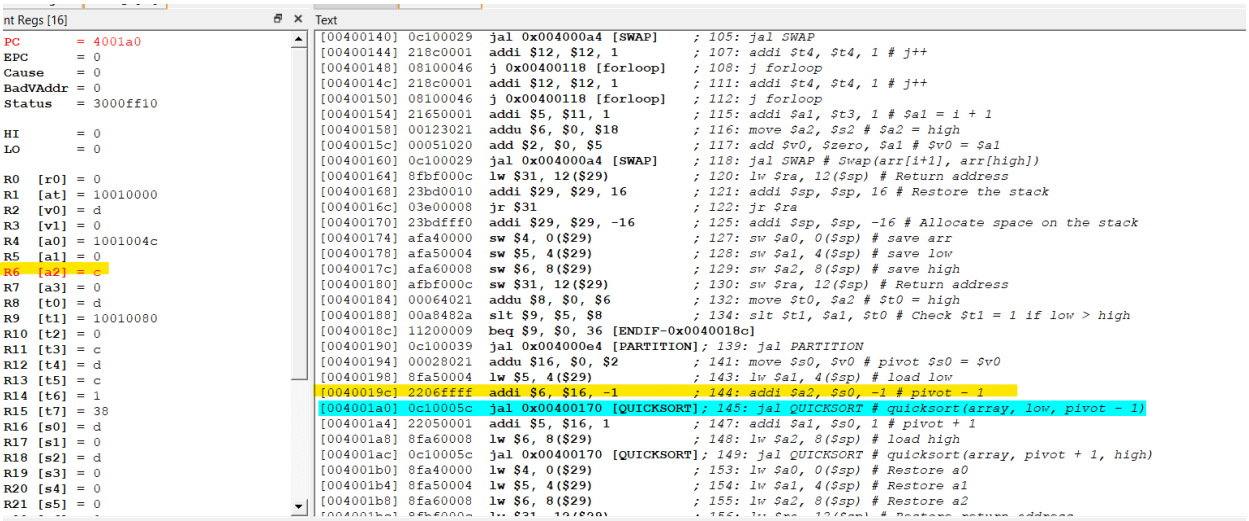

Int Regs [16]
R2 [v0] = d
R3 [v1] = 0
R4 [a0] = 1001004c
R5 [a1] = d
R6 [a2] = d
R7 [a3] = 0
R8 [t0] = d
R9 [t1] = 10010080
R10 [t2] = 0
R11 [t3] = c
R12 [t4] = d
R13 [t5] = c
R14 [t6] = 1
R15 [t7] = 38
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff17c
R30 [s8] = 0
R31 [ra] = 400164
Text
[00400090] 34240048 ori $4, $1, 72 [strLine]
[00400094] 34020004 ori $2, $0, 4 ; 42: li $v0, 4
[00400098] 0000000c syscall ; 43: syscall
[0040009c] 3402000a ori $2, $0, 10 ; 45: li $v0, 10
[004000a0] 0000000c syscall ; 46: syscall
[004000a4] 23bdfff0 addi $29, $29, -16 ; 49: addi $sp, $sp, -16 # Allocate space on the stack
[004000a8] afa40000 sw $4, 0($29) ; 51: sw $a0, 0($sp) # Store array
[004000ac] afa50004 sw $5, 4($29) ; 52: sw $a1, 4($sp) # Store Low
[004000b0] afa60008 sw $6, 8($29) ; 53: sw $a2, 8($sp) # Store High
[004000b4] afaa000c sw $10, 12($29) ; 54: sw $t2, 12($sp) # Store t2 for high
[004000b8] 00054880 sll $9, $5, 2 ; 56: sll $t1, $a1, 2 # $t1 = Low * 4
[004000bc] 00894820 add $9, $4, $9 ; 57: add $t1, $a0, $t1 # $t1 = arr + Low * 4
[004000c0] 8d330000 lw $19, 0($9) ; 58: lw $s3, 0($t1) # $s3 = arr[$t1]
[004000c4] 00065080 sll $10, $6, 2 ; 60: sll $t2, $a2, 2 # $t2 = High * 4
[004000c8] 008a5020 add $10, $4, $10 ; 61: add $t2, $a0, $t2 # $t2 = arr + High * 4
[004000cc] 8d540000 lw $20, 0($10) ; 62: lw $s4, 0($t2) # $s4 = arr[$t2]
[004000d0] ad340000 sw $20, 0($9) ; 64: sw $s4, 0($t1) # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
[004000d4] ad530000 sw $19, 0($10) ; 65: sw $s3, 0($t2) # arr[$t2] = $s3
[004000d8] 8faa000c lw $10, 12($29) ; 67: lw $t2, 12($sp) # Load $t2 on the stack
[004000dc] 23bd0010 addi $29, $29, 16 ; 68: addi $sp, $sp, 16
[004000e0] 03e00008 jr $31 ; 69: jr $ra
[004000e4] 23bdfff0 addi $29, $29, -16 ; 71: addi $sp, $sp, -16 # Allocate space on the stack
[004000e8] afa40000 sw $4, 0($29) ; 73: sw $a0, 0($sp) # Save array on the stack
[004000ec] afa50004 sw $5, 4($29) ; 74: sw $a1, 4($sp) # Save low on the stack
[004000f0] afa60008 sw $6, 8($29) ; 75: sw $a2, 8($sp) # Save high on the stack
[004000f4] afbf000c sw $31, 12($29) ; 76: sw $ra, 12($sp) # Save $ra on the stack
[004000f8] 00058821 addu $17, $0, $5 ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021 addu $18, $0, $6 ; 79: move $s2, $a2 # $s2 = $a2
[00400100] 00124880 sll $9, $18, 2 ; 81: sll $t1, $s2, 2 # t1 = high * 4
[00400104] 00894820 add $9, $4, $9 ; 82: add $t1, $a0, $t1 # t1 = arr + high * 4
[00400108] 8d2a0000 lw $10, 0($9) ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]


Int Regs [16]
R2 [v0] = d
R3 [v1] = 0
R4 [a0] = 1001004c
R5 [a1] = d
R6 [a2] = d
R7 [a3] = 0
R8 [t0] = d
R9 [t1] = 10010080
R10 [t2] = 0
R11 [t3] = c
R12 [t4] = d
R13 [t5] = c
R14 [t6] = 1
R15 [t7] = 38
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff17c
R30 [s8] = 0
R31 [ra] = 400194
Text
[00400110] 00116021 addu $12, $0, $17 ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff addi $13, $18, -1 ; 87: addi $t5, $s2, -1 # t5 = high - 1
[00400118] 01ac702a slt $14, $13, $12 ; 90: slt $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
[0040011c] 15c0000e bne $14, $0, 56 [endfor-0x0040011c]
[00400120] 000c4880 sll $9, $12, 2 ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400124] 01244820 add $9, $9, $4 ; 95: add $t1, $t1, $a0 # $t1 = arr + low * 4
[00400128] 8d2f0000 lw $15, 0($9) ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
[0040012c] 01eac02a slt $24, $15, $10 ; 98: slt $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400130] 17000007 bne $24, $0, 28 [endfif-0x00400130]
[00400134] 216b0001 addi $11, $11, 1 ; 102: addi $t3, $t3, 1 # Increase i++
[00400138] 000b2821 addu $5, $0, $11 ; 103: move $a1, $t3
[0040013c] 000c3021 addu $6, $0, $12 ; 104: move $a2, $t4
[00400140] 0c100029 jal 0x004000a4 [SWAP] ; 105: jal SWAP
[00400144] 218c0001 addi $12, $12, 1 ; 107: addi $t4, $t4, 1 # j++
[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
[0040014c] 218c0001 addi $12, $12, 1 ; 111: addi $t4, $t4, 1 # j++
[00400150] 08100046 j 0x00400118 [forloop] ; 112: j forloop
[00400154] 21650001 addi $5, $11, 1 ; 115: addi $a1, $t3, 1 # $a1 = i + 1
[00400158] 00123021 addu $6, $0, $18 ; 116: move $a2, $s2 # $a2 = high
[0040015c] 00051020 add $2, $0, $5 ; 117: add $v0, $zero, $a1 # $v0 = $a1
[00400160] 0c100029 jal 0x004000a4 [SWAP] ; 118: jal SWAP # Swap(arr[i+1], arr[high])
[00400164] 8fbf000c lw $31, 12($29) ; 120: lw $ra, 12($sp) # Return address
[00400168] 23bd0010 addi $29, $29, 16 ; 121: addi $sp, $sp, 16 # Restore the stack
[0040016c] 03e00008 jr $31 ; 122: jr $ra
[00400170] 23bdfff0 addi $29, $29, -16 ; 125: addi $sp, $sp, -16 # Allocate space on the stack
[00400174] afa40000 sw $4, 0($29) ; 127: sw $a0, 0($sp) # save arr
[00400178] afa50004 sw $5, 4($29) ; 128: sw $a1, 4($sp) # save low
[0040017c] afa60008 sw $6, 8($29) ; 129: sw $a2, 8($sp) # save high
[00400180] afbf000c sw $31, 12($29) ; 130: sw $ra, 12($sp) # Return address
[00400184] 00064021 addu $8, $0, $6 ; 132: move $t0, $a2 # $t0 = high
[00400188] 00a8482a slt $9, $5, $8 ; 134: slt $t1, $a1, $t0 # Check $t1 = 1 if low > high


Int Regs [16]
R2 [v0] = d
R3 [v1] = 0
R4 [a0] = 1001004c
R5 [a1] = d
R6 [a2] = d
R7 [a3] = 0
R8 [t0] = d
R9 [t1] = 10010080
R10 [t2] = 0
R11 [t3] = c
R12 [t4] = d
R13 [t5] = c
R14 [t6] = 1
R15 [t7] = 38
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff18c
R30 [s8] = 0
R31 [ra] = 400194
Text
[00400110] 00116021 addu $12, $0, $17 ; 86: move $t4, $s1 # t4 = j = low
[00400114] 224dffff addi $13, $18, -1 ; 87: addi $t5, $s2, -1 # t5 = high - 1
[00400118] 01ac702a slt $14, $13, $12 ; 90: slt $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
[0040011c] 15c0000e bne $14, $0, 56 [endfor-0x0040011c]
[00400120] 000c4880 sll $9, $12, 2 ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400124] 01244820 add $9, $9, $4 ; 95: add $t1, $t1, $a0 # $t1 = arr + low * 4
[00400128] 8d2f0000 lw $15, 0($9) ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
[0040012c] 01eac02a slt $24, $15, $10 ; 98: slt $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400130] 17000007 bne $24, $0, 28 [endfif-0x00400130]
[00400134] 216b0001 addi $11, $11, 1 ; 102: addi $t3, $t3, 1 # Increase i++
[00400138] 000b2821 addu $5, $0, $11 ; 103: move $a1, $t3
[0040013c] 000c3021 addu $6, $0, $12 ; 104: move $a2, $t4
[00400140] 0c100029 jal 0x004000a4 [SWAP] ; 105: jal SWAP
[00400144] 218c0001 addi $12, $12, 1 ; 107: addi $t4, $t4, 1 # j++
[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
[0040014c] 218c0001 addi $12, $12, 1 ; 111: addi $t4, $t4, 1 # j++
[00400150] 08100046 j 0x00400118 [forloop] ; 112: j forloop
[00400154] 21650001 addi $5, $11, 1 ; 115: addi $a1, $t3, 1 # $a1 = i + 1
[00400158] 00123021 addu $6, $0, $18 ; 116: move $a2, $s2 # $a2 = high
[0040015c] 00051020 add $2, $0, $5 ; 117: add $v0, $zero, $a1 # $v0 = $a1
[00400160] 0c100029 jal 0x004000a4 [SWAP] ; 118: jal SWAP # Swap(arr[i+1], arr[high])
[00400164] 8fbf000c lw $31, 12($29) ; 120: lw $ra, 12($sp) # Return address
[00400168] 23bd0010 addi $29, $29, 16 ; 121: addi $sp, $sp, 16 # Restore the stack
[0040016c] 03e00008 jr $31 ; 122: jr $ra
[00400170] 23bdfff0 addi $29, $29, -16 ; 125: addi $sp, $sp, -16 # Allocate space on the stack
[00400174] afa40000 sw $4, 0($29) ; 127: sw $a0, 0($sp) # save arr
[00400178] afa50004 sw $5, 4($29) ; 128: sw $a1, 4($sp) # save low
[0040017c] afa60008 sw $6, 8($29) ; 129: sw $a2, 8($sp) # save high
[00400180] afbf000c sw $31, 12($29) ; 130: sw $ra, 12($sp) # Return address
[00400184] 00064021 addu $8, $0, $6 ; 132: move $t0, $a2 # $t0 = high
[00400188] 00a8482a slt $9, $5, $8 ; 134: slt $t1, $a1, $t0 # Check $t1 = 1 if low > high


Int Regs [16]
R2 [v0] = d
R3 [v1] = 0
R4 [a0] = 1001004c
R5 [a1] = d
R6 [a2] = d
R7 [a3] = 0
R8 [t0] = d
R9 [t1] = 10010080
R10 [t2] = 0
R11 [t3] = c
R12 [t4] = d
R13 [t5] = c
R14 [t6] = 1
R15 [t7] = 38
R16 [s0] = d
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff18c
R30 [s8] = 0
R31 [ra] = 400194
Text
[00400140] 0c100029 jal 0x004000a4 [SWAP] ; 105: jal SWAP
[00400144] 218c0001 addi $12, $12, 1 ; 107: addi $t4, $t4, 1 # j++
[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
[0040014c] 218c0001 addi $12, $12, 1 ; 111: addi $t4, $t4, 1 # j++
[00400150] 08100046 j 0x00400118 [forloop] ; 112: j forloop
[00400154] 21650001 addi $5, $11, 1 ; 115: addi $a1, $t3, 1 # $a1 = i + 1
[00400158] 00123021 addu $6, $0, $18 ; 116: move $a2, $s2 # $a2 = high
[0040015c] 00051020 add $2, $0, $5 ; 117: add $v0, $zero, $a1 # $v0 = $a1
[00400160] 0c100029 jal 0x004000a4 [SWAP] ; 118: jal SWAP # Swap(arr[i+1], arr[high])
[00400164] 8fbf000c lw $31, 12($29) ; 120: lw $ra, 12($sp) # Return address
[00400168] 23bd0010 addi $29, $29, 16 ; 121: addi $sp, $sp, 16 # Restore the stack
[0040016c] 03e00008 jr $31 ; 122: jr $ra
[00400170] 23bdfff0 addi $29, $29, -16 ; 125: addi $sp, $sp, -16 # Allocate space on the stack
[00400174] afa40000 sw $4, 0($29) ; 127: sw $a0, 0($sp) # save arr
[00400178] afa50004 sw $5, 4($29) ; 128: sw $a1, 4($sp) # save low
[0040017c] afa60008 sw $6, 8($29) ; 129: sw $a2, 8($sp) # save high
[00400180] afbf000c sw $31, 12($29) ; 130: sw $ra, 12($sp) # Return address
[00400184] 00064021 addu $8, $0, $6 ; 132: move $t0, $a2 # $t0 = high
[00400188] 00a8482a slt $9, $5, $8 ; 134: slt $t1, $a1, $t0 # Check $t1 = 1 if low > high
[0040018c] 11200009 beq $9, $0, 36 [ENDIF-0x0040018c]
[00400190] 0c100039 jal 0x004000e4 [PARTITION]; 139: jal PARTITION
[00400194] 00028021 addu $16, $0, $2 ; 141: move $s0, $v0 # pivot $s0 = $v0
[00400198] 8fa50004 lw $5, 4($29) ; 143: lw $a1, 4($sp) # load low
[0040019c] 2206ffff addi $6, $16, -1 ; 144: addi $a2, $s0, -1 # pivot - 1
[004001a0] 0c10005c jal 0x00400170 [QUICKSORT]; 145: jal QUICKSORT # quicksort(array, low, pivot - 1)
[004001a4] 22050001 addi $5, $16, 1 ; 147: addi $a1, $s0, 1 # pivot + 1
[004001a8] 8fa60008 lw $6, 8($29) ; 148: lw $a2, 8($sp) # load high
[004001ac] 0c10005c jal 0x00400170 [QUICKSORT]; 149: jal QUICKSORT # quicksort(array, pivot + 1, high)
[004001b0] 8fa40000 lw $4, 0($29) ; 153: lw $a0, 0($sp) # Restore a0
[004001b4] 8fa50004 lw $5, 4($29) ; 154: lw $a1, 4($sp) # Restore a1
[004001b8] 8fa60008 lw $6, 8($29) ; 155: lw $a2, 8($sp) # Restore a2


Int Regs [16]
PC = 40019c
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 0
R0 [r0] = 0
R1 [at] = 10010000
R2 [v0] = d
R3 [v1] = 0
R4 [a0] = 1001004c
R5 [a1] = 0
R6 [a2] = d
R7 [a3] = 0
R8 [t0] = d
R9 [t1] = 10010080
R10 [t2] = 0
R11 [t3] = c
R12 [t4] = d
R13 [t5] = c
R14 [t6] = 1
R15 [t7] = 38
R16 [s0] = d
R17 [s1] = 0
R18 [s2] = d
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
Text
[00400140] 0c100029 jal 0x004000a4 [SWAP] ; 105: jal SWAP
[00400144] 218c0001 addi $12, $12, 1 ; 107: addi $t4, $t4, 1 # j++
[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
[0040014c] 218c0001 addi $12, $12, 1 ; 111: addi $t4, $t4, 1 # j++
[00400150] 08100046 j 0x00400118 [forloop] ; 112: j forloop
[00400154] 21650001 addi $5, $11, 1 ; 115: addi $a1, $t3, 1 # $a1 = i + 1
[00400158] 00123021 addu $6, $0, $18 ; 116: move $a2, $s2 # $a2 = high
[0040015c] 00051020 add $2, $0, $5 ; 117: add $v0, $zero, $a1 # $v0 = $a1
[00400160] 0c100029 jal 0x004000a4 [SWAP] ; 118: jal SWAP # Swap(arr[i+1], arr[high])
[00400164] 8fbf000c lw $31, 12($29) ; 120: lw $ra, 12($sp) # Return address
[00400168] 23bd0010 addi $29, $29, 16 ; 121: addi $sp, $sp, 16 # Restore the stack
[0040016c] 03e00008 jr $31 ; 122: jr $ra
[00400170] 23bdfff0 addi $29, $29, -16 ; 125: addi $sp, $sp, -16 # Allocate space on the stack
[00400174] afa40000 sw $4, 0($29) ; 127: sw $a0, 0($sp) # save arr
[00400178] afa50004 sw $5, 4($29) ; 128: sw $a1, 4($sp) # save low
[0040017c] afa60008 sw $6, 8($29) ; 129: sw $a2, 8($sp) # save high
[00400180] afbf000c sw $31, 12($29) ; 130: sw $ra, 12($sp) # Return address
[00400184] 00064021 addu $8, $0, $6 ; 132: move $t0, $a2 # $t0 = high
[00400188] 00a8482a slt $9, $5, $8 ; 134: slt $t1, $a1, $t0 # Check $t1 = 1 if low > high
[0040018c] 11200009 beq $9, $0, 36 [ENDIF-0x0040018c]
[00400190] 0c100039 jal 0x004000e4 [PARTITION]; 139: jal PARTITION
[00400194] 00028021 addu $16, $0, $2 ; 141: move $s0, $v0 # pivot $s0 = $v0
[00400198] 8fa50004 lw $5, 4($29) ; 143: lw $a1, 4($sp) # load low
[0040019c] 2206ffff addi $6, $16, -1 ; 144: addi $a2, $s0, -1 # pivot - 1
[004001a0] 0c10005c jal 0x00400170 [QUICKSORT]; 145: jal QUICKSORT # quicksort(array, low, pivot - 1)
[004001a4] 22050001 addi $5, $16, 1 ; 147: addi $a1, $s0, 1 # pivot + 1
[004001a8] 8fa60008 lw $6, 8($29) ; 148: lw $a2, 8($sp) # load high
[004001ac] 0c10005c jal 0x00400170 [QUICKSORT]; 149: jal QUICKSORT # quicksort(array, pivot + 1, high)
[004001b0] 8fa40000 lw $4, 0($29) ; 153: lw $a0, 0($sp) # Restore a0
[004001b4] 8fa50004 lw $5, 4($29) ; 154: lw $a1, 4($sp) # Restore a1
[004001b8] 8fa60008 lw $6, 8($29) ; 155: lw $a2, 8($sp) # Restore a2

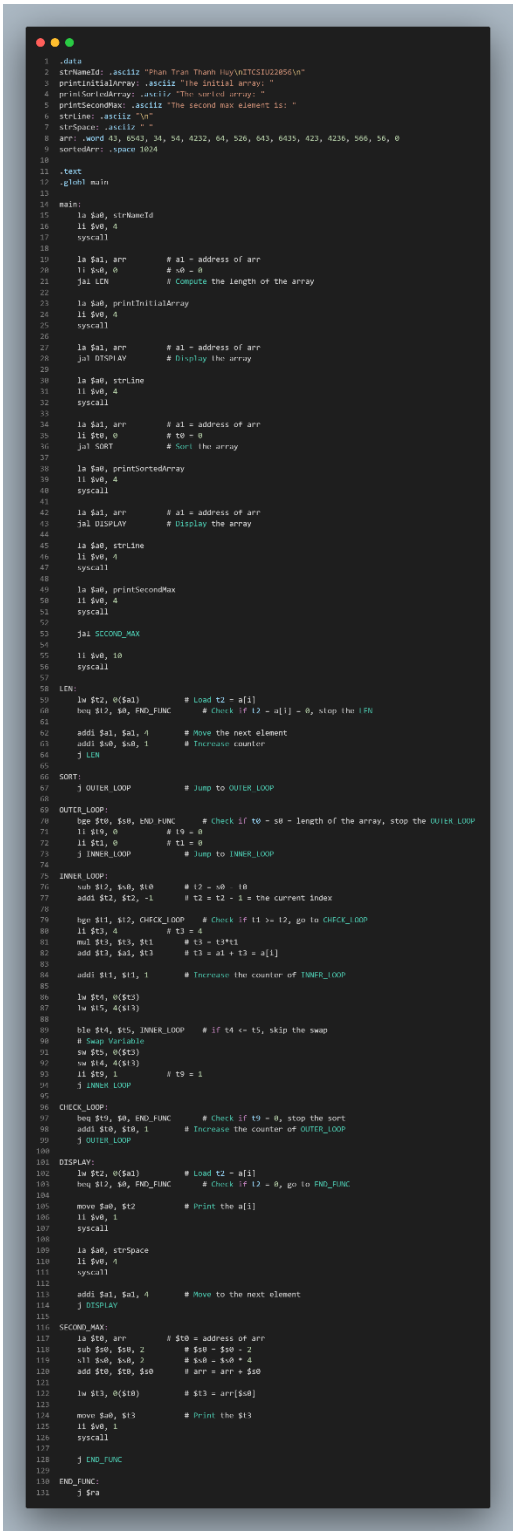


It is the same with QuickSort (Array, Low, Pivot -1)

And QuickSort (Array, Pivot + 1, High)

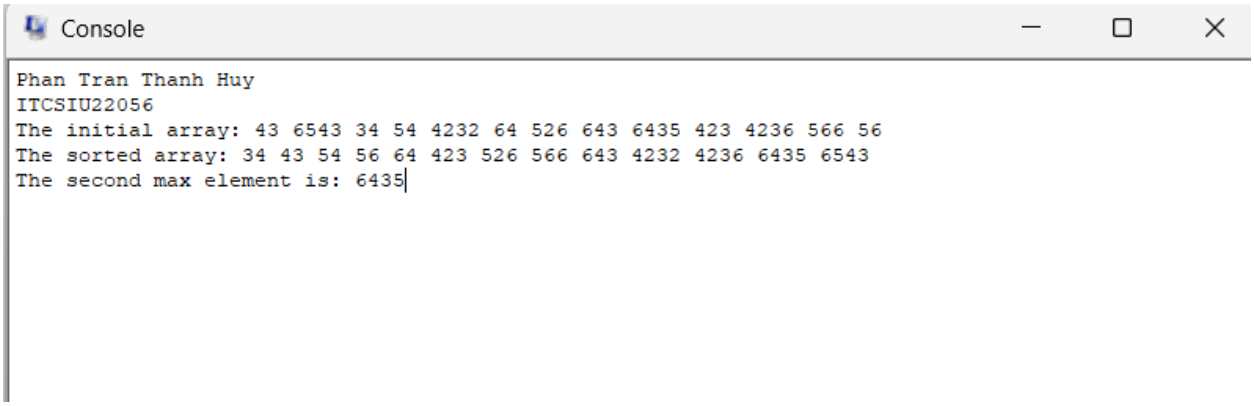
Exercise 3:

Code:



```
.data
strNameId: .asciiz "Phan Tran Thanh Huy\nITCSIU22056\n"
printInitialArray: .asciiz "The initial array: "
printSortedArray: .asciiz "The sorted array: "
printSecondMax: .asciiz "The second max element is: "
strLine: .asciiz "\n"
strSpace: .asciiz " "
arr: .word 43, 6543, 34, 54, 4232, 64, 526, 643, 6435, 423, 4236, 566, 56, 0
sortedArr: .space 1024

.text
.globl main

main:
    la $a0, strNameId
    li $v0, 4
    syscall

    la $a1, arr         # a1 = address of arr
    li $a0, 0           # a0 = 0
    jal LEN             # Compute the length of the array

    la $a0, printInitialArray
    li $v0, 4
    syscall

    la $a1, arr         # a1 = address of arr
    jal DISPLAY         # Display the array

    la $a0, strLine
    li $v0, 4
    syscall

    la $a1, arr         # a1 = address of arr
    li $t0, 0           # t0 = 0
    jal SORT            # Sort the array

    la $a0, printSortedArray
    li $v0, 4
    syscall

    la $a1, arr         # a1 = address of arr
    jal DISPLAY         # Display the array

    la $a0, strLine
    li $v0, 4
    syscall

    la $a0, printSecondMax
    li $v0, 4
    syscall

    jal SECOND_MAX

    li $v0, 10
    syscall

LEN:
    lw $t2, 0($a1)          # Load t2 = a[i]
    beq $t2, $0, END_FUNC       # Check if t2 = a[i] = 0, stop the LEN

    addi $a1, $a1, 4        # Move the next element
    addi $a0, $a0, 1        # Increase counter
    j LEN

SORT:
    j OUTER_LOOP            # Jump to OUTER_LOOP

OUTER_LOOP:
    bge $t0, $s0, END_FUNC      # Check if t0 = s0 = length of the array, stop the OUTER_LOOP
    li $t9, 0           # t9 = 0
    li $t1, 0           # t1 = 0
    j INNER_LOOP            # Jump to INNER_LOOP

INNER_LOOP:
    sub $t2, $s0, $t0       # t2 = s0 - t0
    addi $t2, $t2, -1       # t2 = t2 - 1 = the current index

    bge $t1, $t2, CHECK_LOOP    # Check if t1 >= t2, go to CHECK_LOOP
    li $t3, 4           # t3 = 4
    mul $t3, $t3, $t1       # t3 = t3*t1
    add $t3, $a1, $t3       # t3 = a1 + t3 = a[i]

    addi $t1, $t1, 1        # Increase the counter of INNER_LOOP

    lw $t4, 0($t3)
    lw $t5, 4($t3)

    ble $t4, $t5, INNER_LOOP    # if t4 <= t5, skip the swap
    # Swap Variable
    sw $t5, 0($t3)
    sw $t4, 4($t3)
    li $t9, 1           # t9 = 1
    j INNER_LOOP

CHECK_LOOP:
    beq $t9, $0, END_FUNC       # Check if t9 = 0, stop the sort
    addi $t0, $t0, 1        # Increase the counter of OUTER_LOOP
    j OUTER_LOOP

DISPLAY:
    lw $t2, 0($a1)          # Load t2 = a[i]
    beq $t2, $0, END_FUNC       # Check if t2 = 0, go to END_FUNC

    move $a0, $t2           # Print the a[i]
    li $v0, 1
    syscall

    la $a0, strSpace
    li $v0, 4
    syscall

    addi $a1, $a1, 4        # Move to the next element
    j DISPLAY

SECOND_MAX:
    la $t0, arr         # $t0 = address of arr
    sub $s0, $s0, 2         # $s0 = $s0 - 2
    sll $s0, $s0, 2         # $s0 = $s0 * 4
    add $t0, $t0, $s0       # arr = arr + $s0

    lw $t3, 0($t0)          # $t3 = arr[$s0]

    move $a0, $t3           # Print the $t3
    li $v0, 1
    syscall

    j END_FUNC

END_FUNC:
    j $ra
```

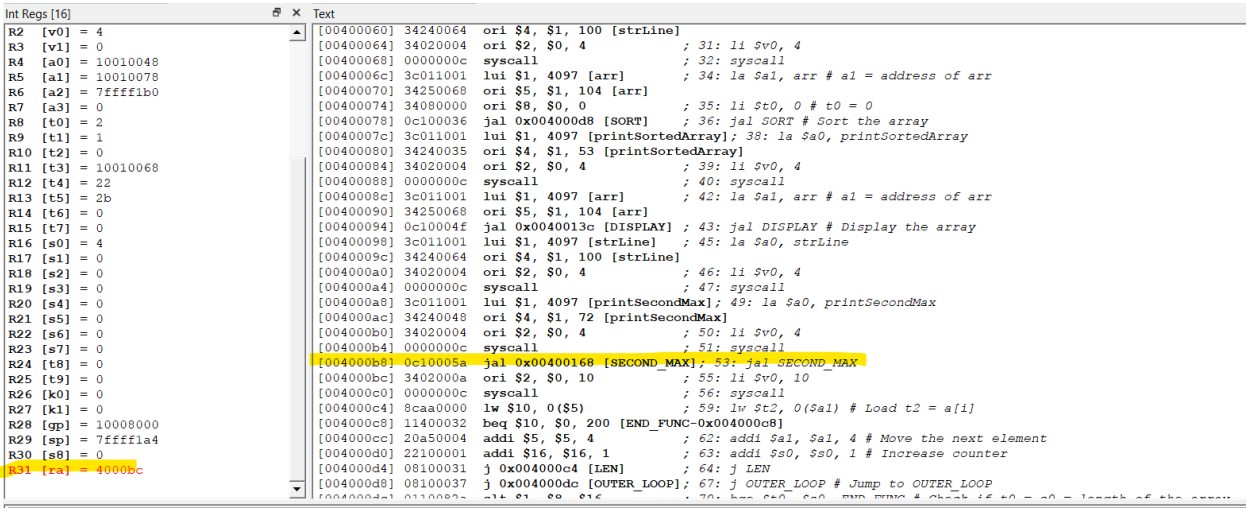
Test case:

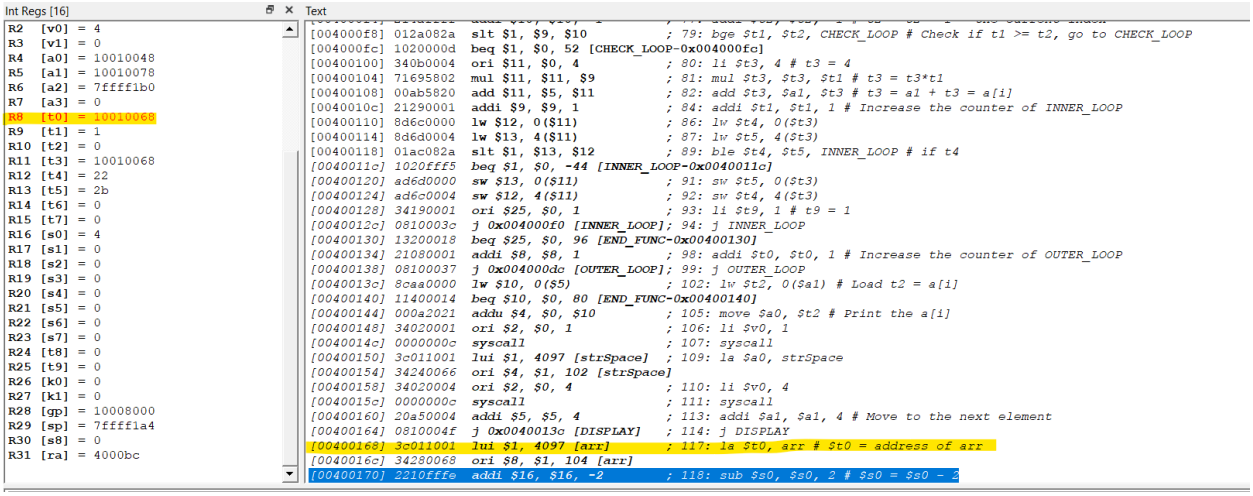


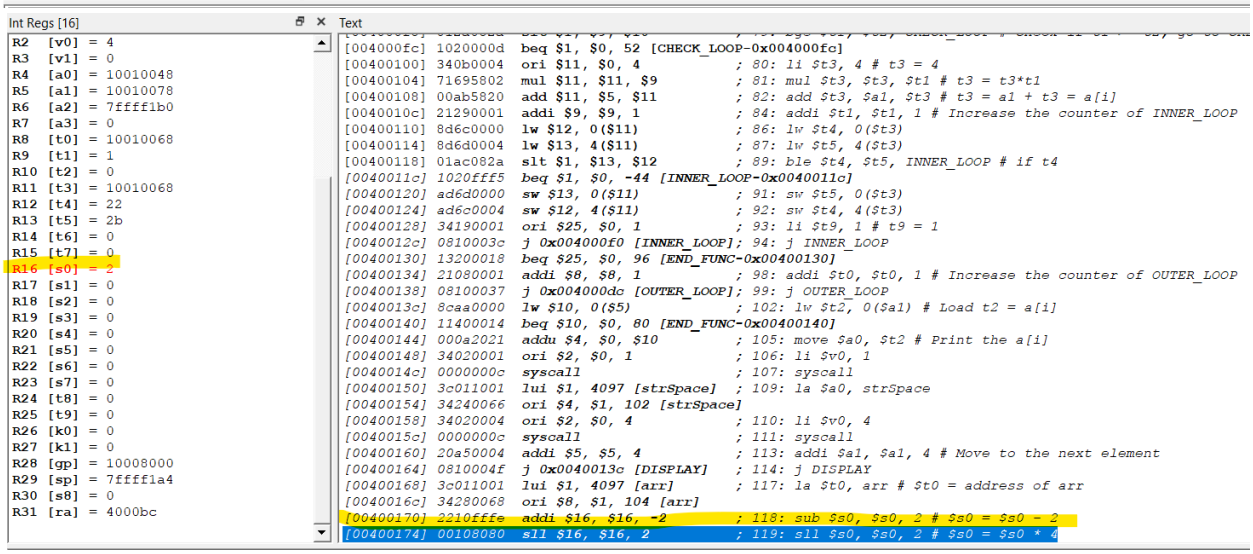
Single Step:

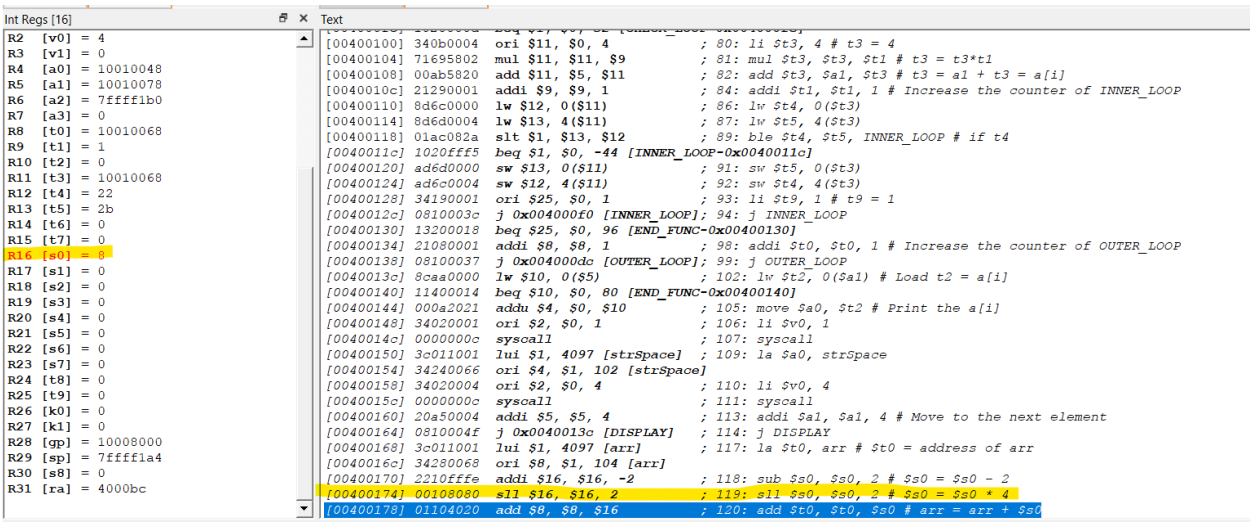
It is the same steps with Bubble_Sort

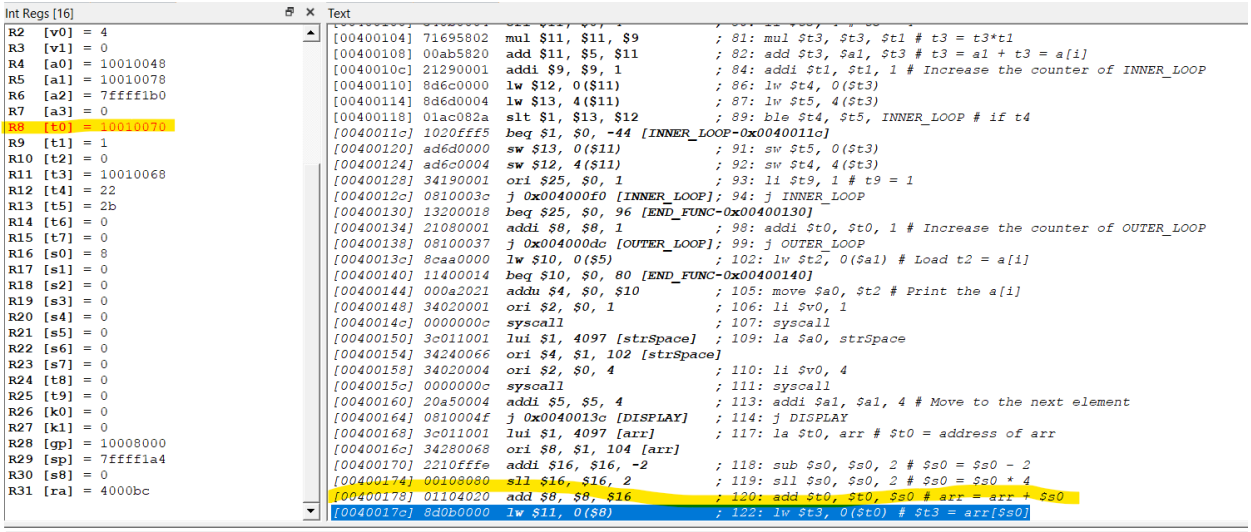
SECOND_MAX steps:

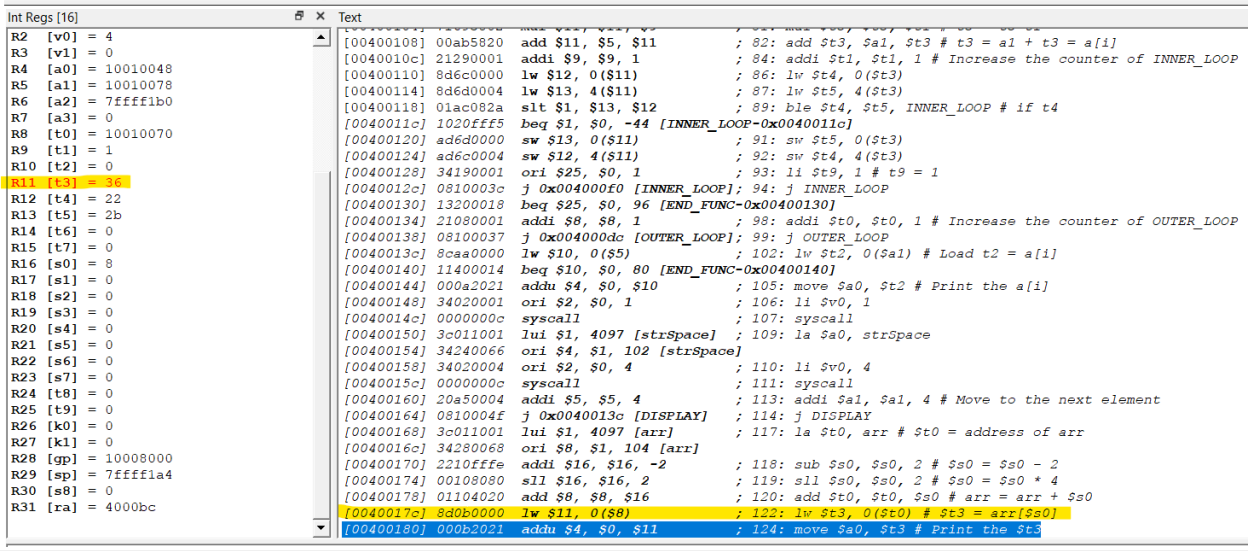


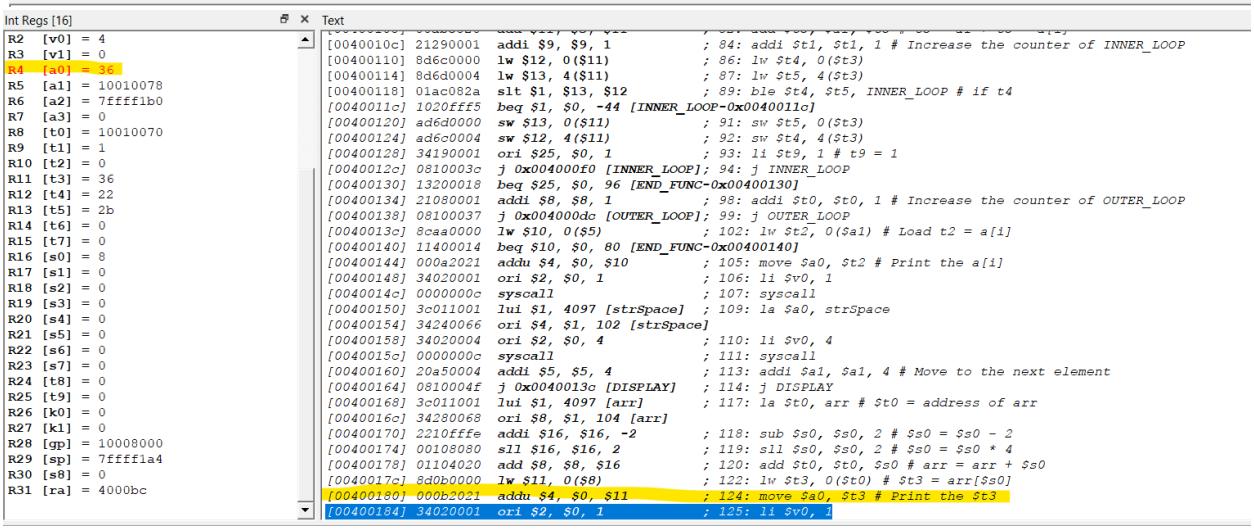

Int Regs [16]
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 10010048
R5 [a1] = 10010078
R6 [a2] = 7ffff1b0
R7 [a3] = 0
R8 [t0] = 10010068
R9 [t1] = 1
R10 [t2] = 0
R11 [t3] = 10010068
R12 [t4] = 22
R13 [t5] = 2b
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 4
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff1a4
R30 [s8] = 0
R31 [ra] = 4000bc
Text
[004000f8] 012a082a slt $1, $9, $10 ; 79: bge $t1, $t2, CHECK_LOOP # Check if t1 >= t2, go to CHECK_LOOP
[004000fc] 1020000d beq $1, $0, 52 [CHECK_LOOP-0x004000fc]
[00400100] 340b0004 ori $11, $0, 4 ; 80: li $t3, 4 # t3 = 4
[00400104] 71695802 mul $11, $11, $9 ; 81: mul $t3, $t3, $t1 # t3 = t3*t1
[00400108] 00ab5820 add $11, $5, $11 ; 82: add $t3, $a1, $t3 # t3 = a1 + t3 = a[i]
[0040010c] 21290001 addi $9, $9, 1 ; 84: addi $t1, $t1, 1 # Increase the counter of INNER_LOOP
[00400110] 8d6c0000 lw $12, 0($11) ; 86: lw $t4, 0($t3)
[00400114] 8d6d0004 lw $13, 4($11) ; 87: lw $t5, 4($t3)
[00400118] 01ac082a slt $1, $13, $12 ; 89: ble $t4, $t5, INNER_LOOP # if t4
[0040011c] 1020fff5 beq $1, $0, -44 [INNER_LOOP-0x0040011c]
[00400120] ad6d0000 sw $13, 0($11) ; 91: sw $t5, 0($t3)
[00400124] ad6c0004 sw $12, 4($11) ; 92: sw $t4, 4($t3)
[00400128] 34190001 ori $25, $0, 1 ; 93: li $t9, 1 # t9 = 1
[0040012c] 0810003c j 0x004000f0 [INNER_LOOP]; 94: j INNER_LOOP
[00400130] 13200018 beq $25, $0, 96 [END_FUNC-0x00400130]
[00400134] 21080001 addi $8, $8, 1 ; 98: addi $t0, $t0, 1 # Increase the counter of OUTER_LOOP
[00400138] 08100037 j 0x004000dc [OUTER_LOOP]; 99: j OUTER_LOOP
[0040013c] 8caa0000 lw $10, 0($5) ; 102: lw $t2, 0($a1) # Load t2 = a[i]
[00400140] 11400014 beq $10, $0, 80 [END_FUNC-0x00400140]
[00400144] 000a2021 addu $4, $0, $10 ; 105: move $a0, $t2 # Print the a[i]
[00400148] 34020001 ori $2, $0, 1 ; 106: li $v0, 1
[0040014c] 0000000c syscall ; 107: syscall
[00400150] 3c011001 lui $1, 4097 [strSpace] ; 109: la $a0, strSpace
[00400154] 34240066 ori $4, $1, 102 [strSpace]
[00400158] 34020004 ori $2, $0, 4 ; 110: li $v0, 4
[0040015c] 0000000c syscall ; 111: syscall
[00400160] 20a50004 addi $5, $5, 4 ; 113: addi $a1, $a1, 4 # Move to the next element
[00400164] 0810004f j 0x0040013c [DISPLAY] ; 114: j DISPLAY
[00400168] 3c011001 lui $1, 4097 [arr] ; 117: la $t0, arr # $t0 = address of arr
[0040016c] 34280068 ori $8, $1, 104 [arr]
[00400170] 2210fffe addi $16, $16, -2 ; 118: sub $s0, $s0, 2 # $s0 = $s0 - 2


Int Regs [16]
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 10010048
R5 [a1] = 10010078
R6 [a2] = 7ffff1b0
R7 [a3] = 0
R8 [t0] = 10010068
R9 [t1] = 1
R10 [t2] = 0
R11 [t3] = 10010068
R12 [t4] = 22
R13 [t5] = 2b
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 2
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff1a4
R30 [s8] = 0
R31 [ra] = 4000bc
Text
[004000fc] 1020000d beq $1, $0, 52 [CHECK_LOOP-0x004000fc]
[00400100] 340b0004 ori $11, $0, 4 ; 80: li $t3, 4 # t3 = 4
[00400104] 71695802 mul $11, $11, $9 ; 81: mul $t3, $t3, $t1 # t3 = t3*t1
[00400108] 00ab5820 add $11, $5, $11 ; 82: add $t3, $a1, $t3 # t3 = a1 + t3 = a[i]
[0040010c] 21290001 addi $9, $9, 1 ; 84: addi $t1, $t1, 1 # Increase the counter of INNER_LOOP
[00400110] 8d6c0000 lw $12, 0($11) ; 86: lw $t4, 0($t3)
[00400114] 8d6d0004 lw $13, 4($11) ; 87: lw $t5, 4($t3)
[00400118] 01ac082a slt $1, $13, $12 ; 89: ble $t4, $t5, INNER_LOOP # if t4
[0040011c] 1020fff5 beq $1, $0, -44 [INNER_LOOP-0x0040011c]
[00400120] ad6d0000 sw $13, 0($11) ; 91: sw $t5, 0($t3)
[00400124] ad6c0004 sw $12, 4($11) ; 92: sw $t4, 4($t3)
[00400128] 34190001 ori $25, $0, 1 ; 93: li $t9, 1 # t9 = 1
[0040012c] 0810003c j 0x004000f0 [INNER_LOOP]; 94: j INNER_LOOP
[00400130] 13200018 beq $25, $0, 96 [END_FUNC-0x00400130]
[00400134] 21080001 addi $8, $8, 1 ; 98: addi $t0, $t0, 1 # Increase the counter of OUTER_LOOP
[00400138] 08100037 j 0x004000dc [OUTER_LOOP]; 99: j OUTER_LOOP
[0040013c] 8caa0000 lw $10, 0($5) ; 102: lw $t2, 0($a1) # Load t2 = a[i]
[00400140] 11400014 beq $10, $0, 80 [END_FUNC-0x00400140]
[00400144] 000a2021 addu $4, $0, $10 ; 105: move $a0, $t2 # Print the a[i]
[00400148] 34020001 ori $2, $0, 1 ; 106: li $v0, 1
[0040014c] 0000000c syscall ; 107: syscall
[00400150] 3c011001 lui $1, 4097 [strSpace] ; 109: la $a0, strSpace
[00400154] 34240066 ori $4, $1, 102 [strSpace]
[00400158] 34020004 ori $2, $0, 4 ; 110: li $v0, 4
[0040015c] 0000000c syscall ; 111: syscall
[00400160] 20a50004 addi $5, $5, 4 ; 113: addi $a1, $a1, 4 # Move to the next element
[00400164] 0810004f j 0x0040013c [DISPLAY] ; 114: j DISPLAY
[00400168] 3c011001 lui $1, 4097 [arr] ; 117: la $t0, arr # $t0 = address of arr
[0040016c] 34280068 ori $8, $1, 104 [arr]
[00400170] 2210fffe addi $16, $16, -2 ; 118: sub $s0, $s0, 2 # $s0 = $s0 - 2
[00400174] 00108080 sll $16, $16, 2 ; 119: sll $s0, $s0, 2 # $s0 = $s0 * 4

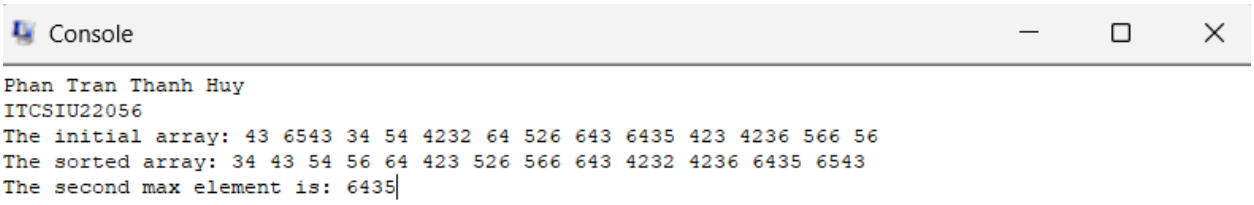

Int Regs [16]
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 10010048
R5 [a1] = 10010078
R6 [a2] = 7ffff1b0
R7 [a3] = 0
R8 [t0] = 10010068
R9 [t1] = 1
R10 [t2] = 0
R11 [t3] = 10010068
R12 [t4] = 22
R13 [t5] = 2b
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff1a4
R30 [s8] = 0
R31 [ra] = 4000bc
Text
[00400100] 340b0004 ori $11, $0, 4 ; 80: li $t3, 4 # t3 = 4
[00400104] 71695802 mul $11, $11, $9 ; 81: mul $t3, $t3, $t1 # t3 = t3*t1
[00400108] 00ab5820 add $11, $5, $11 ; 82: add $t3, $a1, $t3 # t3 = a1 + t3 = a[i]
[0040010c] 21290001 addi $9, $9, 1 ; 84: addi $t1, $t1, 1 # Increase the counter of INNER_LOOP
[00400110] 8d6c0000 lw $12, 0($11) ; 86: lw $t4, 0($t3)
[00400114] 8d6d0004 lw $13, 4($11) ; 87: lw $t5, 4($t3)
[00400118] 01ac082a slt $1, $13, $12 ; 89: ble $t4, $t5, INNER_LOOP # if t4
[0040011c] 1020fff5 beq $1, $0, -44 [INNER_LOOP-0x0040011c]
[00400120] ad6d0000 sw $13, 0($11) ; 91: sw $t5, 0($t3)
[00400124] ad6c0004 sw $12, 4($11) ; 92: sw $t4, 4($t3)
[00400128] 34190001 ori $25, $0, 1 ; 93: li $t9, 1 # t9 = 1
[0040012c] 0810003c j 0x004000f0 [INNER_LOOP]; 94: j INNER_LOOP
[00400130] 13200018 beq $25, $0, 96 [END_FUNC-0x00400130]
[00400134] 21080001 addi $8, $8, 1 ; 98: addi $t0, $t0, 1 # Increase the counter of OUTER_LOOP
[00400138] 08100037 j 0x004000dc [OUTER_LOOP]; 99: j OUTER_LOOP
[0040013c] 8caa0000 lw $10, 0($5) ; 102: lw $t2, 0($a1) # Load t2 = a[i]
[00400140] 11400014 beq $10, $0, 80 [END_FUNC-0x00400140]
[00400144] 000a2021 addu $4, $0, $10 ; 105: move $a0, $t2 # Print the a[i]
[00400148] 34020001 ori $2, $0, 1 ; 106: li $v0, 1
[0040014c] 0000000c syscall ; 107: syscall
[00400150] 3c011001 lui $1, 4097 [strSpace] ; 109: la $a0, strSpace
[00400154] 34240066 ori $4, $1, 102 [strSpace]
[00400158] 34020004 ori $2, $0, 4 ; 110: li $v0, 4
[0040015c] 0000000c syscall ; 111: syscall
[00400160] 20a50004 addi $5, $5, 4 ; 113: addi $a1, $a1, 4 # Move to the next element
[00400164] 0810004f j 0x0040013c [DISPLAY] ; 114: j DISPLAY
[00400168] 3c011001 lui $1, 4097 [arr] ; 117: la $t0, arr # $t0 = address of arr
[0040016c] 34280068 ori $8, $1, 104 [arr]
[00400170] 2210fffe addi $16, $16, -2 ; 118: sub $s0, $s0, 2 # $s0 = $s0 - 2
[00400174] 00108080 sll $16, $16, 2 ; 119: sll $s0, $s0, 2 # $s0 = $s0 * 4
[00400178] 01104020 add $8, $8, $16 ; 120: add $t0, $t0, $s0 # arr = arr + $s0


Int Regs [16]
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 10010048
R5 [a1] = 10010078
R6 [a2] = 7ffff1b0
R7 [a3] = 0
R8 [t0] = 10010070
R9 [t1] = 1
R10 [t2] = 0
R11 [t3] = 10010068
R12 [t4] = 22
R13 [t5] = 2b
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 8
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff1a4
R30 [s8] = 0
R31 [ra] = 4000bc
Text
[00400104] 71695802 mul $11, $11, $9 ; 81: mul $t3, $t3, $t1 # t3 = t3*t1
[00400108] 00ab5820 add $11, $5, $11 ; 82: add $t3, $a1, $t3 # t3 = a1 + t3 = a[i]
[0040010c] 21290001 addi $9, $9, 1 ; 84: addi $t1, $t1, 1 # Increase the counter of INNER_LOOP
[00400110] 8d6c0000 lw $12, 0($11) ; 86: lw $t4, 0($t3)
[00400114] 8d6d0004 lw $13, 4($11) ; 87: lw $t5, 4($t3)
[00400118] 01ac082a slt $1, $13, $12 ; 89: ble $t4, $t5, INNER_LOOP # if t4
[0040011c] 1020fff5 beq $1, $0, -44 [INNER_LOOP-0x0040011c]
[00400120] ad6d0000 sw $13, 0($11) ; 91: sw $t5, 0($t3)
[00400124] ad6c0004 sw $12, 4($11) ; 92: sw $t4, 4($t3)
[00400128] 34190001 ori $25, $0, 1 ; 93: li $t9, 1 # t9 = 1
[0040012c] 0810003c j 0x004000f0 [INNER_LOOP]; 94: j INNER_LOOP
[00400130] 13200018 beq $25, $0, 96 [END_FUNC-0x00400130]
[00400134] 21080001 addi $8, $8, 1 ; 98: addi $t0, $t0, 1 # Increase the counter of OUTER_LOOP
[00400138] 08100037 j 0x004000dc [OUTER_LOOP]; 99: j OUTER_LOOP
[0040013c] 8caa0000 lw $10, 0($5) ; 102: lw $t2, 0($a1) # Load t2 = a[i]
[00400140] 11400014 beq $10, $0, 80 [END_FUNC-0x00400140]
[00400144] 000a2021 addu $4, $0, $10 ; 105: move $a0, $t2 # Print the a[i]
[00400148] 34020001 ori $2, $0, 1 ; 106: li $v0, 1
[0040014c] 0000000c syscall ; 107: syscall
[00400150] 3c011001 lui $1, 4097 [strSpace] ; 109: la $a0, strSpace
[00400154] 34240066 ori $4, $1, 102 [strSpace]
[00400158] 34020004 ori $2, $0, 4 ; 110: li $v0, 4
[0040015c] 0000000c syscall ; 111: syscall
[00400160] 20a50004 addi $5, $5, 4 ; 113: addi $a1, $a1, 4 # Move to the next element
[00400164] 0810004f j 0x0040013c [DISPLAY] ; 114: j DISPLAY
[00400168] 3c011001 lui $1, 4097 [arr] ; 117: la $t0, arr # $t0 = address of arr
[0040016c] 34280068 ori $8, $1, 104 [arr]
[00400170] 2210fffe addi $16, $16, -2 ; 118: sub $s0, $s0, 2 # $s0 = $s0 - 2
[00400174] 00108080 sll $16, $16, 2 ; 119: sll $s0, $s0, 2 # $s0 = $s0 * 4
[00400178] 01104020 add $8, $8, $16 ; 120: add $t0, $t0, $s0 # arr = arr + $s0
[0040017c] 8d0b0000 lw $11, 0($8) ; 122: lw $t3, 0($t0) # $t3 = arr[$s0]


Int Regs [16]
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 10010048
R5 [a1] = 10010078
R6 [a2] = 7ffff1b0
R7 [a3] = 0
R8 [t0] = 10010070
R9 [t1] = 1
R10 [t2] = 0
R11 [t3] = 36
R12 [t4] = 22
R13 [t5] = 2b
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 8
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff1a4
R30 [s8] = 0
R31 [ra] = 4000bc
Text
[00400108] 00ab5820 add $11, $5, $11 ; 82: add $t3, $a1, $t3 # t3 = a1 + t3 = a[i]
[0040010c] 21290001 addi $9, $9, 1 ; 84: addi $t1, $t1, 1 # Increase the counter of INNER_LOOP
[00400110] 8d6c0000 lw $12, 0($11) ; 86: lw $t4, 0($t3)
[00400114] 8d6d0004 lw $13, 4($11) ; 87: lw $t5, 4($t3)
[00400118] 01ac082a slt $1, $13, $12 ; 89: ble $t4, $t5, INNER_LOOP # if t4
[0040011c] 1020fff5 beq $1, $0, -44 [INNER_LOOP-0x0040011c]
[00400120] ad6d0000 sw $13, 0($11) ; 91: sw $t5, 0($t3)
[00400124] ad6c0004 sw $12, 4($11) ; 92: sw $t4, 4($t3)
[00400128] 34190001 ori $25, $0, 1 ; 93: li $t9, 1 # t9 = 1
[0040012c] 0810003c j 0x004000f0 [INNER_LOOP]; 94: j INNER_LOOP
[00400130] 13200018 beq $25, $0, 96 [END_FUNC-0x00400130]
[00400134] 21080001 addi $8, $8, 1 ; 98: addi $t0, $t0, 1 # Increase the counter of OUTER_LOOP
[00400138] 08100037 j 0x004000dc [OUTER_LOOP]; 99: j OUTER_LOOP
[0040013c] 8caa0000 lw $10, 0($5) ; 102: lw $t2, 0($a1) # Load t2 = a[i]
[00400140] 11400014 beq $10, $0, 80 [END_FUNC-0x00400140]
[00400144] 000a2021 addu $4, $0, $10 ; 105: move $a0, $t2 # Print the a[i]
[00400148] 34020001 ori $2, $0, 1 ; 106: li $v0, 1
[0040014c] 0000000c syscall ; 107: syscall
[00400150] 3c011001 lui $1, 4097 [strSpace] ; 109: la $a0, strSpace
[00400154] 34240066 ori $4, $1, 102 [strSpace]
[00400158] 34020004 ori $2, $0, 4 ; 110: li $v0, 4
[0040015c] 0000000c syscall ; 111: syscall
[00400160] 20a50004 addi $5, $5, 4 ; 113: addi $a1, $a1, 4 # Move to the next element
[00400164] 0810004f j 0x0040013c [DISPLAY] ; 114: j DISPLAY
[00400168] 3c011001 lui $1, 4097 [arr] ; 117: la $t0, arr # $t0 = address of arr
[0040016c] 34280068 ori $8, $1, 104 [arr]
[00400170] 2210fffe addi $16, $16, -2 ; 118: sub $s0, $s0, 2 # $s0 = $s0 - 2
[00400174] 00108080 sll $16, $16, 2 ; 119: sll $s0, $s0, 2 # $s0 = $s0 * 4
[00400178] 01104020 add $8, $8, $16 ; 120: add $t0, $t0, $s0 # arr = arr + $s0
[0040017c] 8d0b0000 lw $11, 0($8) ; 122: lw $t3, 0($t0) # $t3 = arr[$s0]
[00400180] 000b2021 addu $4, $0, $11 ; 124: move $a0, $t3 # Print the $t3

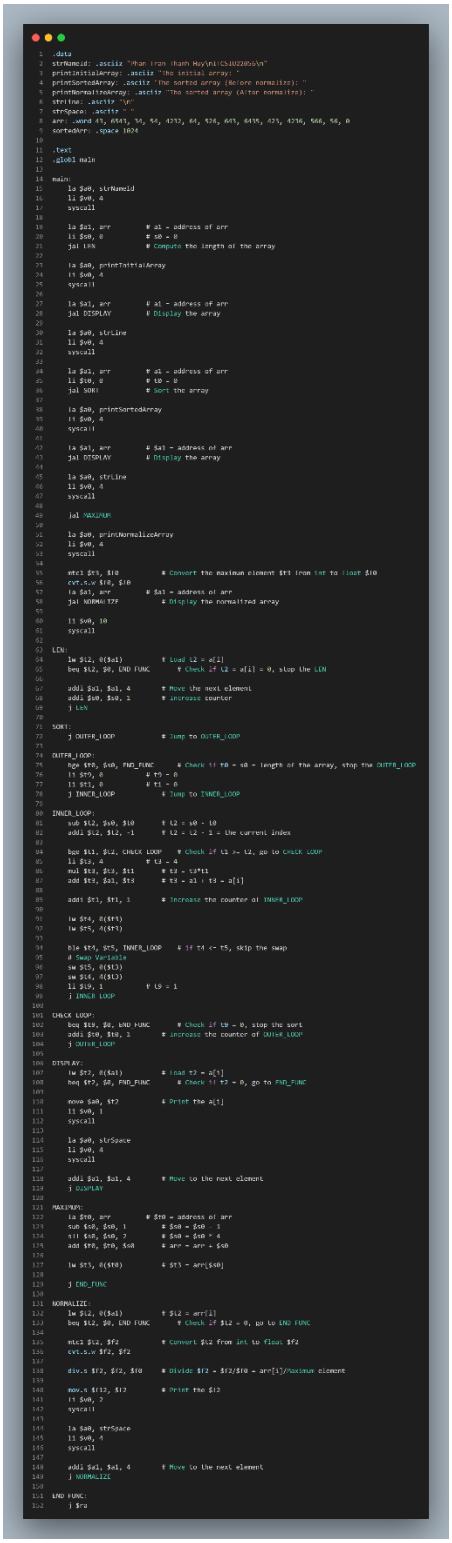

Int Regs [16]
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 36
R5 [a1] = 10010078
R6 [a2] = 7ffff1b0
R7 [a3] = 0
R8 [t0] = 10010070
R9 [t1] = 1
R10 [t2] = 0
R11 [t3] = 36
R12 [t4] = 22
R13 [t5] = 2b
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 8
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff1a4
R30 [s8] = 0
R31 [ra] = 4000bc
Text
[0040010c] 21290001 addi $9, $9, 1 ; 84: addi $t1, $t1, 1 # Increase the counter of INNER_LOOP
[00400110] 8d6c0000 lw $12, 0($11) ; 86: lw $t4, 0($t3)
[00400114] 8d6d0004 lw $13, 4($11) ; 87: lw $t5, 4($t3)
[00400118] 01ac082a slt $1, $13, $12 ; 89: ble $t4, $t5, INNER_LOOP # if t4
[0040011c] 1020fff5 beq $1, $0, -44 [INNER_LOOP-0x0040011c]
[00400120] ad6d0000 sw $13, 0($11) ; 91: sw $t5, 0($t3)
[00400124] ad6c0004 sw $12, 4($11) ; 92: sw $t4, 4($t3)
[00400128] 34190001 ori $25, $0, 1 ; 93: li $t9, 1 # t9 = 1
[0040012c] 0810003c j 0x004000f0 [INNER_LOOP]; 94: j INNER_LOOP
[00400130] 13200018 beq $25, $0, 96 [END_FUNC-0x00400130]
[00400134] 21080001 addi $8, $8, 1 ; 98: addi $t0, $t0, 1 # Increase the counter of OUTER_LOOP
[00400138] 08100037 j 0x004000dc [OUTER_LOOP]; 99: j OUTER_LOOP
[0040013c] 8caa0000 lw $10, 0($5) ; 102: lw $t2, 0($a1) # Load t2 = a[i]
[00400140] 11400014 beq $10, $0, 80 [END_FUNC-0x00400140]
[00400144] 000a2021 addu $4, $0, $10 ; 105: move $a0, $t2 # Print the a[i]
[00400148] 34020001 ori $2, $0, 1 ; 106: li $v0, 1
[0040014c] 0000000c syscall ; 107: syscall
[00400150] 3c011001 lui $1, 4097 [strSpace] ; 109: la $a0, strSpace
[00400154] 34240066 ori $4, $1, 102 [strSpace]
[00400158] 34020004 ori $2, $0, 4 ; 110: li $v0, 4
[0040015c] 0000000c syscall ; 111: syscall
[00400160] 20a50004 addi $5, $5, 4 ; 113: addi $a1, $a1, 4 # Move to the next element
[00400164] 0810004f j 0x0040013c [DISPLAY] ; 114: j DISPLAY
[00400168] 3c011001 lui $1, 4097 [arr] ; 117: la $t0, arr # $t0 = address of arr
[0040016c] 34280068 ori $8, $1, 104 [arr]
[00400170] 2210fffe addi $16, $16, -2 ; 118: sub $s0, $s0, 2 # $s0 = $s0 - 2
[00400174] 00108080 sll $16, $16, 2 ; 119: sll $s0, $s0, 2 # $s0 = $s0 * 4
[00400178] 01104020 add $8, $8, $16 ; 120: add $t0, $t0, $s0 # arr = arr + $s0
[0040017c] 8d0b0000 lw $11, 0($8) ; 122: lw $t3, 0($t0) # $t3 = arr[$s0]
[00400180] 000b2021 addu $4, $0, $11 ; 124: move $a0, $t3 # Print the $t3
[00400184] 34020001 ori $2, $0, 1 ; 125: li $v0, 1

Print the result:



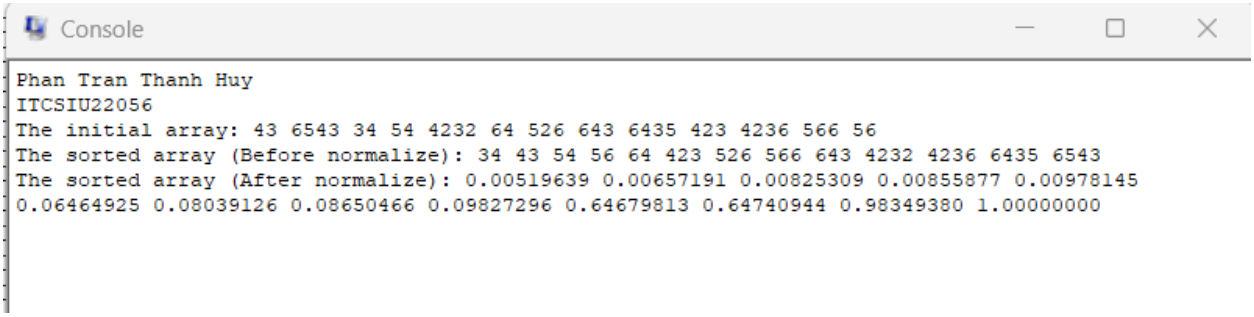
Exercise 4:

Code:



```
.data
strNameId: .asciiz "Phan Tran Thanh Huy\nMSSV:B1602285\n"
printInitialArray: .asciiz "The initial array: "
printSortedArray: .asciiz "The sorted array (Before normalize): "
printNormalizeArray: .asciiz "The sorted array (After normalize): "
strLine: .asciiz "\n"
strSpace: .asciiz " "
arr: .word 43, 6543, 76, 54, 4232, 64, 926, 643, 6436, 423, 4256, 968, 56, 0
sortedArr: .space 1024

.text
.globl main

main:
    la $a0, strNameId
    li $v0, 4
    syscall

    la $a1, arr         # a1 = address of arr
    li $s0, 0           # s0 = 0
    jal LEN             # Compute the length of the array

    la $a0, printInitialArray
    li $v0, 4
    syscall

    la $a1, arr         # a1 = address of arr
    jal DISPLAY         # Display the array

    la $a0, strLine
    li $v0, 4
    syscall

    la $a1, arr         # a1 = address of arr
    li $t0, 0           # t0 = 0
    jal SORT            # Sort the array

    la $a0, printSortedArray
    li $v0, 4
    syscall

    la $a1, arr         # $a1 = address of arr
    jal DISPLAY         # Display the array

    la $a0, strLine
    li $v0, 4
    syscall

    jal MAXIMUM

    la $a0, printNormalizeArray
    li $v0, 4
    syscall

    mtc1 $t1, $f0           # Convert the maximum element $t1 from int to float $f0
    cvt.s.w $f0, $f0
    la $a1, arr         # $a1 = address of arr
    jal NORMALIZE           # Display the normalized array

    li $v0, 10
    syscall

LEN:
    lw $t2, 0($a1)          # Load t2 = a[i]
    beq $t2, $0, END_FUNC       # Check if t2 = a[i] = 0, stop the LEN

    addi $a1, $a1, 4        # Move the next element
    addi $s0, $s0, 1        # Increase counter
    j LEN

SORT:
    j OUTER_LOOP            # Jump to OUTER_LOOP

OUTER_LOOP:
    bge $t0, $s0, END_FUNC      # Check if t0 = s0 = length of the array, stop the OUTER_LOOP
    li $t7, 0           # t7 = 0
    li $t1, 0           # t1 = 0
    j INNER_LOOP            # Jump to INNER_LOOP

INNER_LOOP:
    sub $t2, $s0, $t0       # t2 = s0 - t0
    addi $t2, $t2, -1       # t2 = t2 - 1 = the current index

    bge $t1, $t2, CHECK_LOOP    # Check if t1 >= t2, go to CHECK_LOOP
    li $t3, 4           # t3 = 4
    mul $t8, $t3, $t1       # t8 = t3*t1
    add $t9, $a1, $t8       # t9 = a1 + t8 = a[i]

    addi $t1, $t1, 1        # Increase the counter of INNER_LOOP

    lw $t4, 0($t9)
    lw $t5, 4($t9)

    ble $t4, $t5, INNER_LOOP    # if t4 <= t5, skip the swap
    # Swap Variable
    sw $t5, 0($t9)
    sw $t4, 4($t9)
    li $t7, 1           # t7 = 1
    j INNER_LOOP

CHECK_LOOP:
    beq $t7, $0, END_FUNC       # Check if t7 = 0, stop the sort
    addi $t0, $t0, 1        # Increase the counter of OUTER_LOOP
    j OUTER_LOOP

DISPLAY:
    lw $t2, 0($a1)          # Load t2 = a[i]
    beq $t2, $0, END_FUNC       # Check if t2 = 0, go to END_FUNC

    move $a0, $t2           # Print the a[i]
    li $v0, 1
    syscall

    la $a0, strSpace
    li $v0, 4
    syscall

    addi $a1, $a1, 4        # Move to the next element
    j DISPLAY

MAXIMUM:
    la $t0, arr         # $t0 = address of arr
    sub $s0, $s0, 1         # $s0 = $s0 - 1
    sll $s0, $s0, 2         # $s0 = $s0 * 4
    add $t0, $t0, $s0       # arr = arr + $s0

    lw $t1, 0($t0)          # $t1 = arr[$s0]

    j END_FUNC

NORMALIZE:
    lw $t2, 0($a1)          # $t2 = arr[i]
    beq $t2, $0, END_FUNC       # Check if $t2 = 0, go to END_FUNC

    mtc1 $t2, $f2           # Convert $t2 from int to float $f2
    cvt.s.w $f2, $f2

    div.s $f2, $f2, $f0     # Divide $f2 = $f2/$f0 = arr[i]/maximum element

    mov.s $f12, $f2         # Print the $f2
    li $v0, 2
    syscall

    la $a0, strSpace
    li $v0, 4
    syscall

    addi $a1, $a1, 4        # Move to the next element
    j NORMALIZE

END_FUNC:
    jr $ra
```

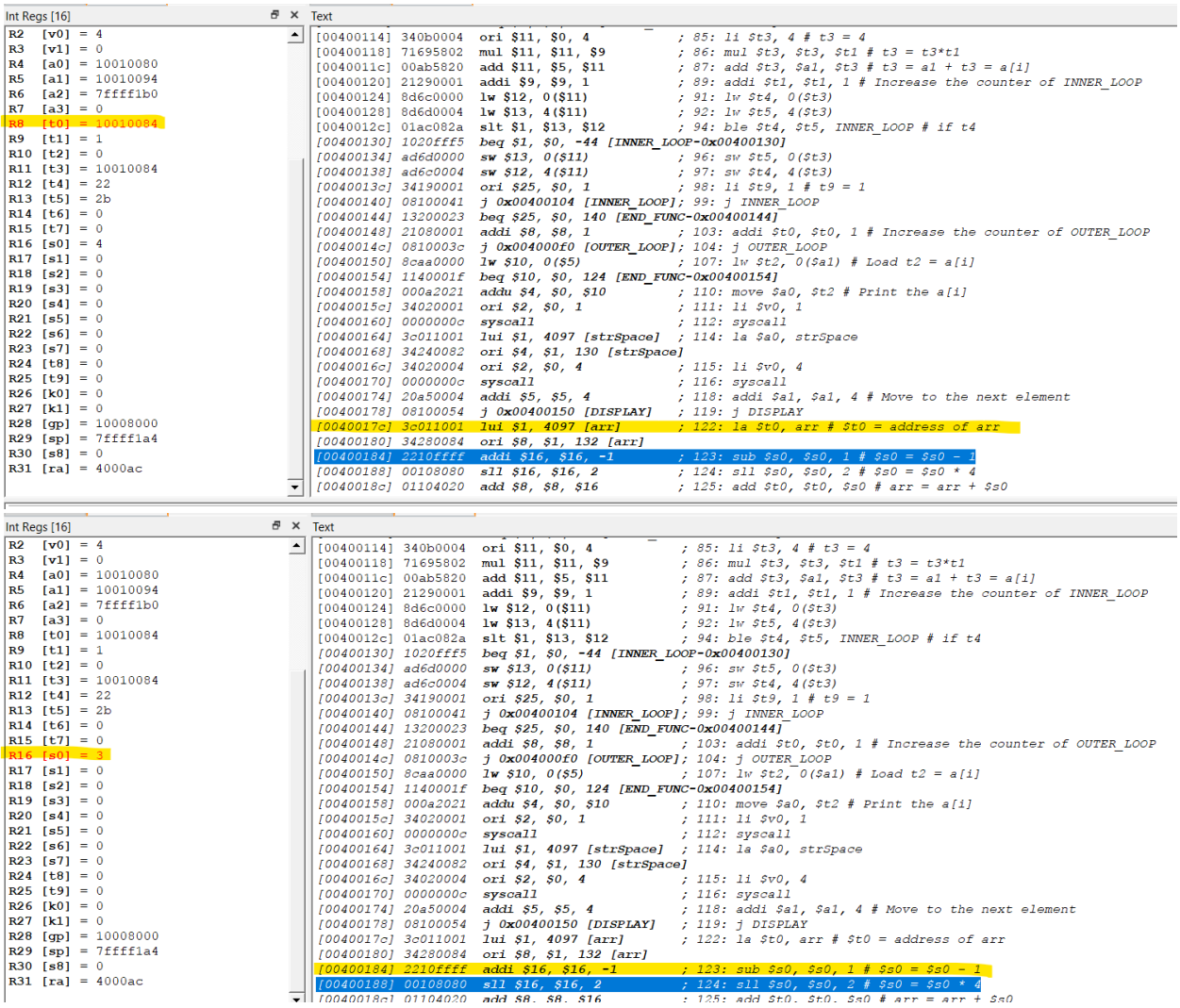
Test case:

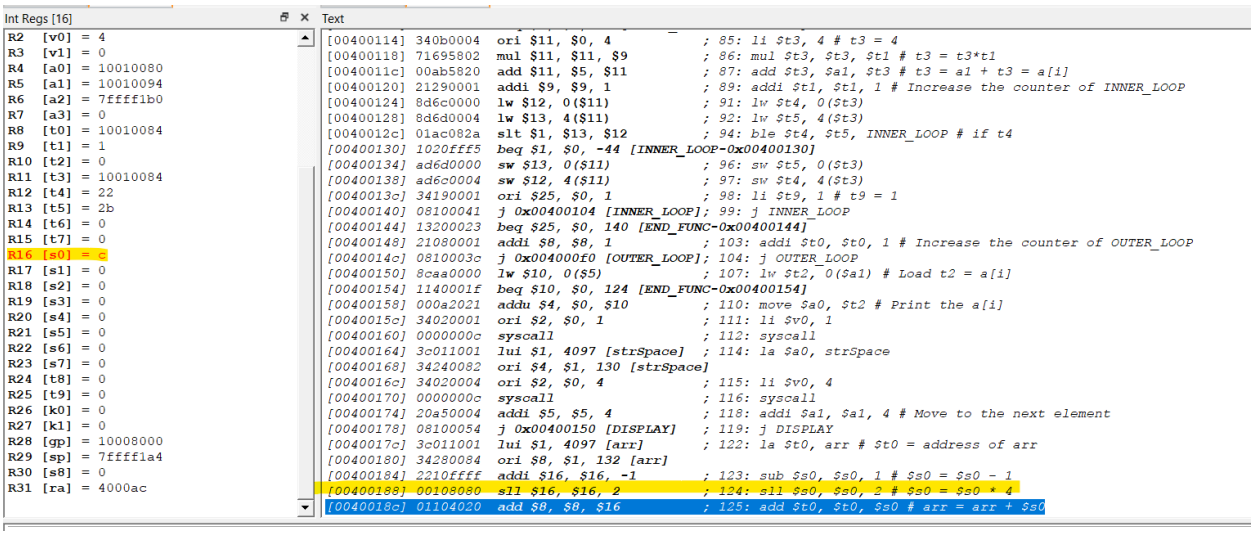

Console
Phan Tran Thanh Huy
ITCSIU22056
The initial array: 43 6543 34 54 4232 64 526 643 6435 423 4236 566 56
The sorted array (Before normalize): 34 43 54 56 64 423 526 566 643 4232 4236 6435 6543
The sorted array (After normalize): 0.00519639 0.00657191 0.00825309 0.00855877 0.00978145
0.06464925 0.08039126 0.08650466 0.09827296 0.64679813 0.64740944 0.98349380 1.00000000

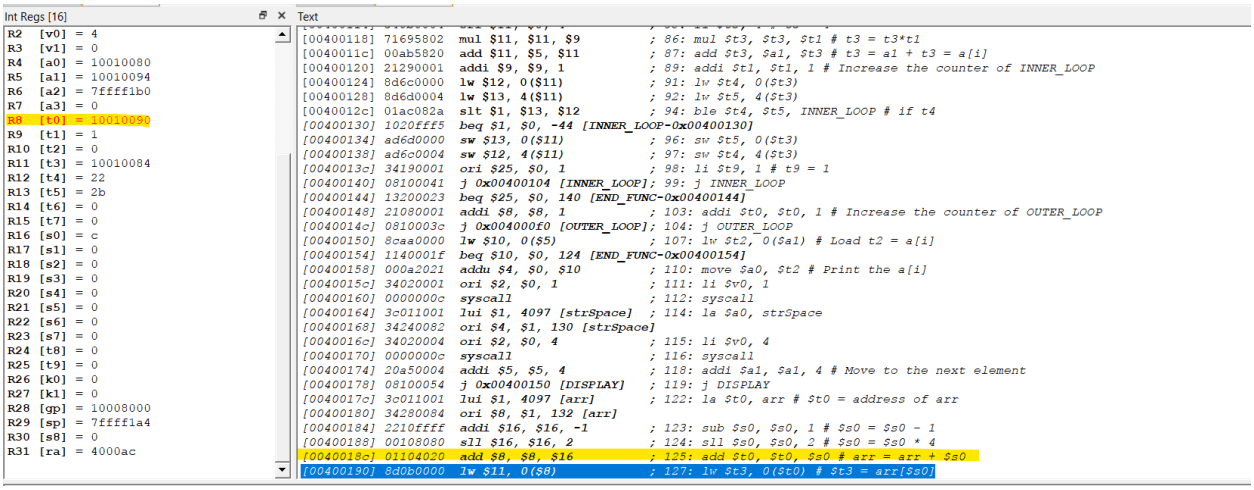
Single Step:

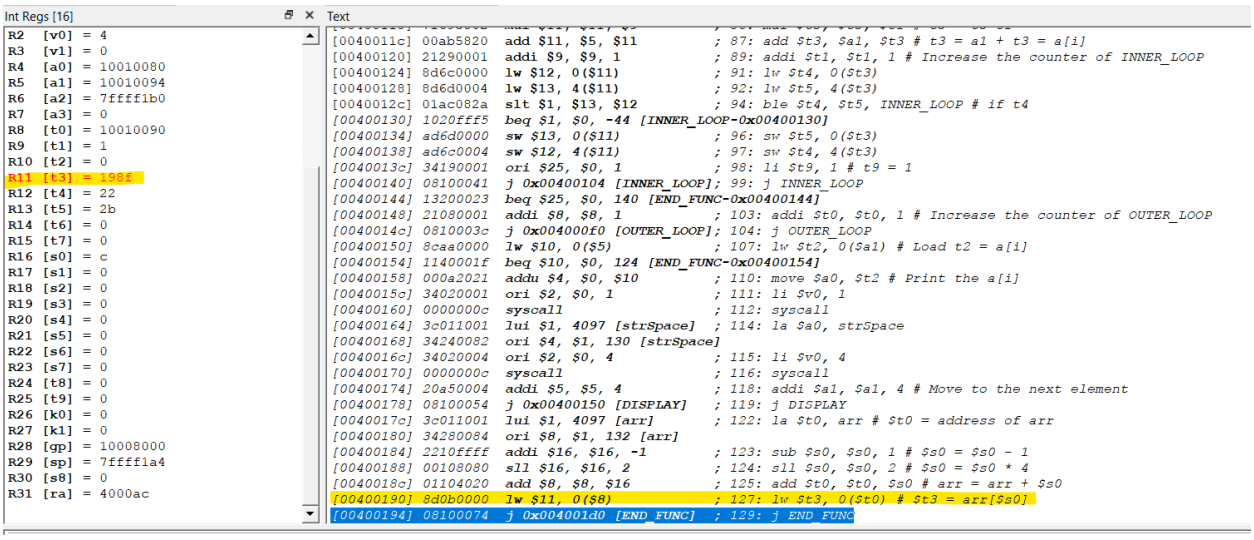
It is the same steps with Bubble_Sort

MAXIMUM element steps:

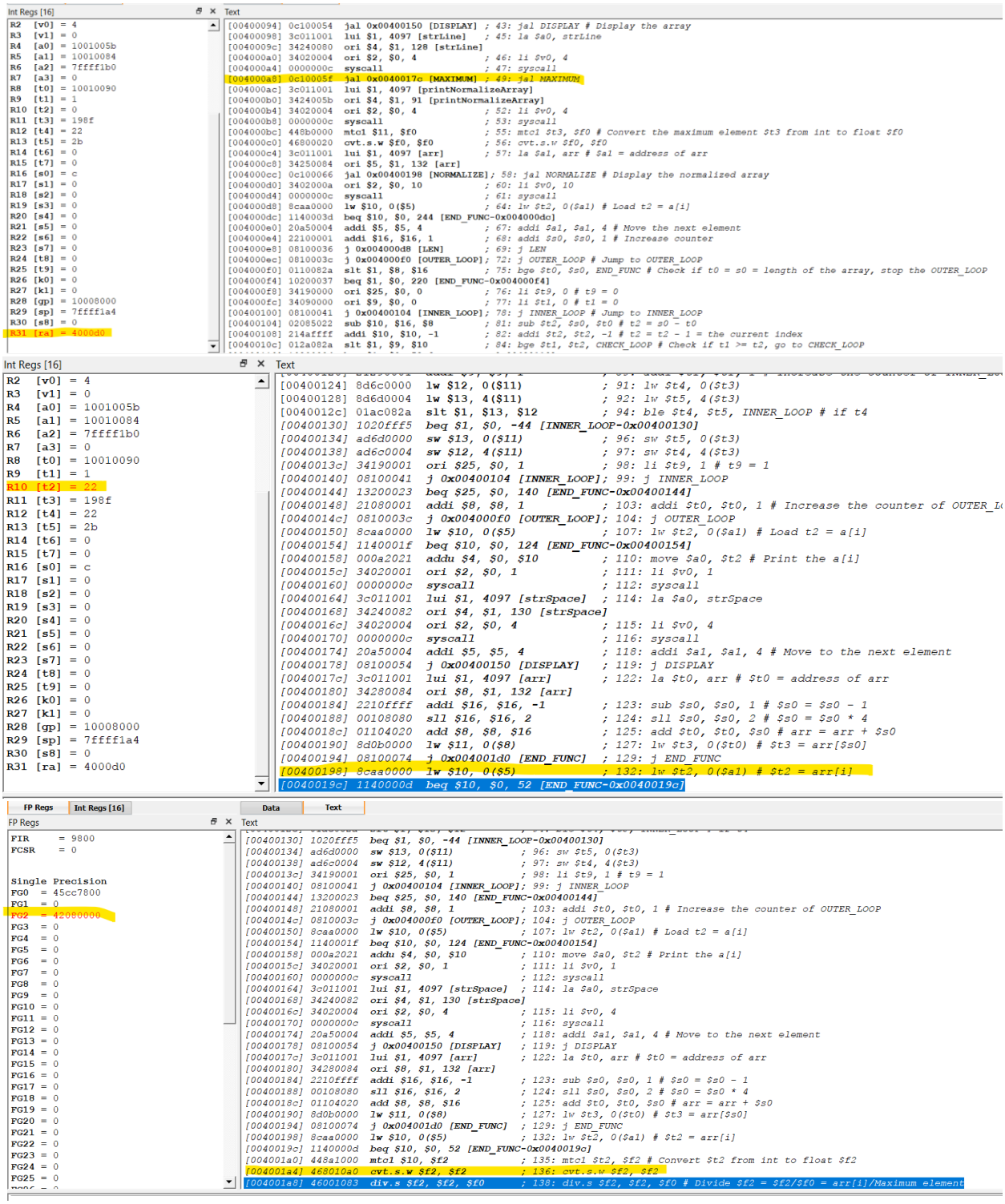


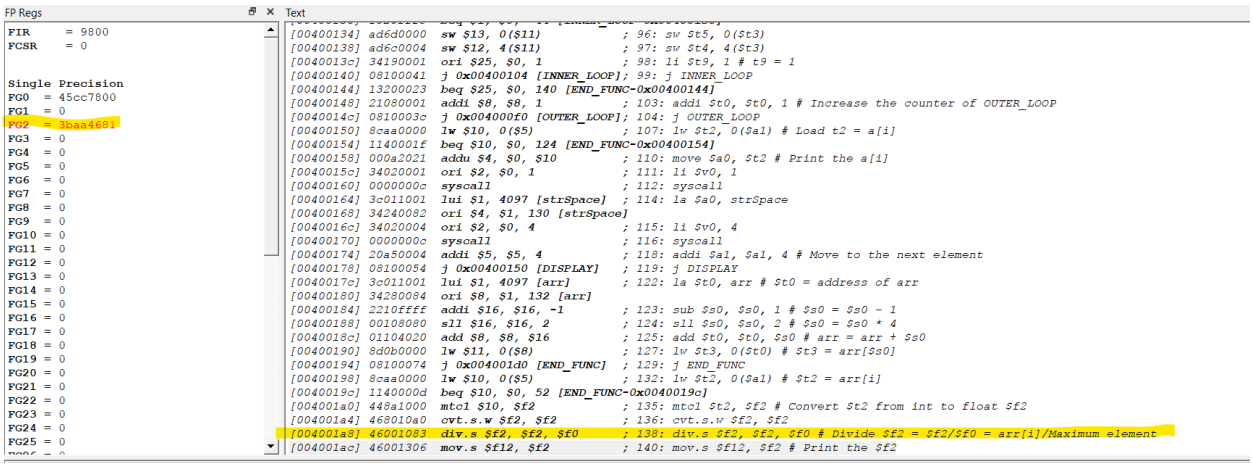

Int Regs [16]
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 10010080
R5 [a1] = 10010094
R6 [a2] = 7ffff1b0
R7 [a3] = 0
R8 [t0] = 10010084
R9 [t1] = 1
R10 [t2] = 0
R11 [t3] = 10010084
R12 [t4] = 22
R13 [t5] = 2b
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = c
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff1a4
R30 [s8] = 0
R31 [ra] = 4000ac
Text
[00400114] 340b0004 ori $11, $0, 4 ; 85: li $t3, 4 # t3 = 4
[00400118] 71695802 mul $11, $11, $9 ; 86: mul $t3, $t3, $t1 # t3 = t3*t1
[0040011c] 00ab5820 add $11, $5, $11 ; 87: add $t3, $a1, $t3 # t3 = a1 + t3 = a[i]
[00400120] 21290001 addi $9, $9, 1 ; 89: addi $t1, $t1, 1 # Increase the counter of INNER_LOOP
[00400124] 8d6c0000 lw $12, 0($11) ; 91: lw $t4, 0($t3)
[00400128] 8d6d0004 lw $13, 4($11) ; 92: lw $t5, 4($t3)
[0040012c] 01ac082a slt $1, $13, $12 ; 94: ble $t4, $t5, INNER_LOOP # if t4
[00400130] 1020fff5 beq $1, $0, -44 [INNER_LOOP-0x00400130]
[00400134] ad6d0000 sw $13, 0($11) ; 96: sw $t5, 0($t3)
[00400138] ad6c0004 sw $12, 4($11) ; 97: sw $t4, 4($t3)
[0040013c] 34190001 ori $25, $0, 1 ; 98: li $t9, 1 # t9 = 1
[00400140] 08100041 j 0x00400104 [INNER_LOOP]; 99: j INNER_LOOP
[00400144] 13200023 beq $25, $0, 140 [END_FUNC-0x00400144]
[00400148] 21080001 addi $8, $8, 1 ; 103: addi $t0, $t0, 1 # Increase the counter of OUTER_LOOP
[0040014c] 0810003c j 0x004000f0 [OUTER_LOOP]; 104: j OUTER_LOOP
[00400150] 8caa0000 lw $10, 0($5) ; 107: lw $t2, 0($a1) # Load t2 = a[i]
[00400154] 1140001f beq $10, $0, 124 [END_FUNC-0x00400154]
[00400158] 000a2021 addu $4, $0, $10 ; 110: move $a0, $t2 # Print the a[i]
[0040015c] 34020001 ori $2, $0, 1 ; 111: li $v0, 1
[00400160] 0000000c syscall ; 112: syscall
[00400164] 3c011001 lui $1, 4097 [strSpace] ; 114: la $a0, strSpace
[00400168] 34240082 ori $4, $1, 130 [strSpace]
[0040016c] 34020004 ori $2, $0, 4 ; 115: li $v0, 4
[00400170] 0000000c syscall ; 116: syscall
[00400174] 20a50004 addi $5, $5, 4 ; 118: addi $a1, $a1, 4 # Move to the next element
[00400178] 08100054 j 0x00400150 [DISPLAY] ; 119: j DISPLAY
[0040017c] 3c011001 lui $1, 4097 [arr] ; 122: la $t0, arr # $t0 = address of arr
[00400180] 34280084 ori $8, $1, 132 [arr]
[00400184] 2210ffff addi $16, $16, -1 ; 123: sub $s0, $s0, 1 # $s0 = $s0 - 1
[00400188] 00108080 sll $16, $16, 2 ; 124: sll $s0, $s0, 2 # $s0 = $s0 * 4
[0040018c] 01104020 add $8, $8, $16 ; 125: add $t0, $t0, $s0 # arr = arr + $s0

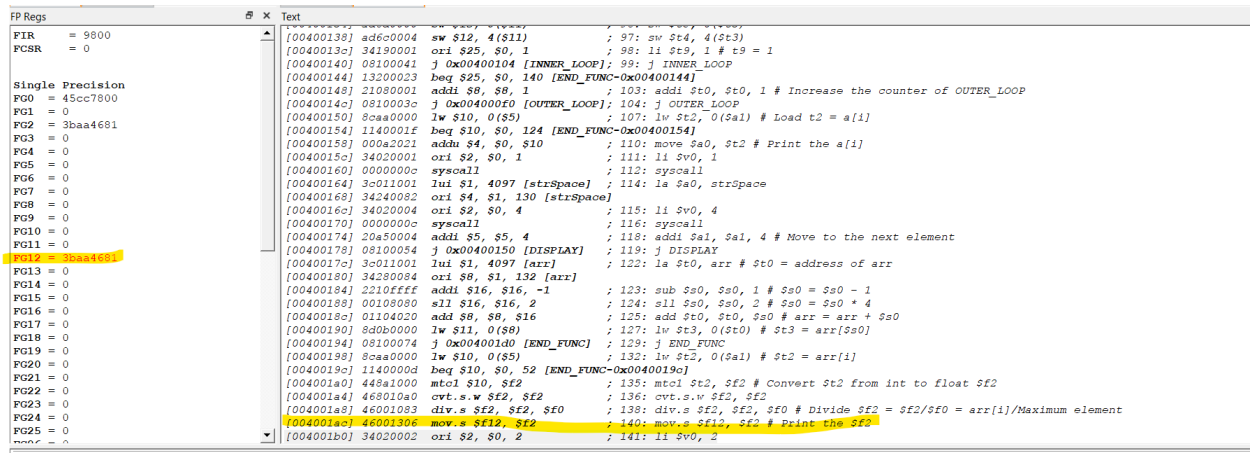

Int Regs [16]
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 10010080
R5 [a1] = 10010094
R6 [a2] = 7ffff1b0
R7 [a3] = 0
R8 [t0] = 10010090
R9 [t1] = 1
R10 [t2] = 0
R11 [t3] = 10010084
R12 [t4] = 22
R13 [t5] = 2b
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = c
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff1a4
R30 [s8] = 0
R31 [ra] = 4000ac
Text
[00400118] 71695802 mul $11, $11, $9 ; 86: mul $t3, $t3, $t1 # t3 = t3*t1
[0040011c] 00ab5820 add $11, $5, $11 ; 87: add $t3, $a1, $t3 # t3 = a1 + t3 = a[i]
[00400120] 21290001 addi $9, $9, 1 ; 89: addi $t1, $t1, 1 # Increase the counter of INNER_LOOP
[00400124] 8d6c0000 lw $12, 0($11) ; 91: lw $t4, 0($t3)
[00400128] 8d6d0004 lw $13, 4($11) ; 92: lw $t5, 4($t3)
[0040012c] 01ac082a slt $1, $13, $12 ; 94: ble $t4, $t5, INNER_LOOP # if t4
[00400130] 1020fff5 beq $1, $0, -44 [INNER_LOOP-0x00400130]
[00400134] ad6d0000 sw $13, 0($11) ; 96: sw $t5, 0($t3)
[00400138] ad6c0004 sw $12, 4($11) ; 97: sw $t4, 4($t3)
[0040013c] 34190001 ori $25, $0, 1 ; 98: li $t9, 1 # t9 = 1
[00400140] 08100041 j 0x00400104 [INNER_LOOP]; 99: j INNER_LOOP
[00400144] 13200023 beq $25, $0, 140 [END_FUNC-0x00400144]
[00400148] 21080001 addi $8, $8, 1 ; 103: addi $t0, $t0, 1 # Increase the counter of OUTER_LOOP
[0040014c] 0810003c j 0x004000f0 [OUTER_LOOP]; 104: j OUTER_LOOP
[00400150] 8caa0000 lw $10, 0($5) ; 107: lw $t2, 0($a1) # Load t2 = a[i]
[00400154] 1140001f beq $10, $0, 124 [END_FUNC-0x00400154]
[00400158] 000a2021 addu $4, $0, $10 ; 110: move $a0, $t2 # Print the a[i]
[0040015c] 34020001 ori $2, $0, 1 ; 111: li $v0, 1
[00400160] 0000000c syscall ; 112: syscall
[00400164] 3c011001 lui $1, 4097 [strSpace] ; 114: la $a0, strSpace
[00400168] 34240082 ori $4, $1, 130 [strSpace]
[0040016c] 34020004 ori $2, $0, 4 ; 115: li $v0, 4
[00400170] 0000000c syscall ; 116: syscall
[00400174] 20a50004 addi $5, $5, 4 ; 118: addi $a1, $a1, 4 # Move to the next element
[00400178] 08100054 j 0x00400150 [DISPLAY] ; 119: j DISPLAY
[0040017c] 3c011001 lui $1, 4097 [arr] ; 122: la $t0, arr # $t0 = address of arr
[00400180] 34280084 ori $8, $1, 132 [arr]
[00400184] 2210ffff addi $16, $16, -1 ; 123: sub $s0, $s0, 1 # $s0 = $s0 - 1
[00400188] 00108080 sll $16, $16, 2 ; 124: sll $s0, $s0, 2 # $s0 = $s0 * 4
[0040018c] 01104020 add $8, $8, $16 ; 125: add $t0, $t0, $s0 # arr = arr + $s0
[00400190] 8d0b0000 lw $11, 0($8) ; 127: lw $t3, 0($t0) # $t3 = arr[$s0]

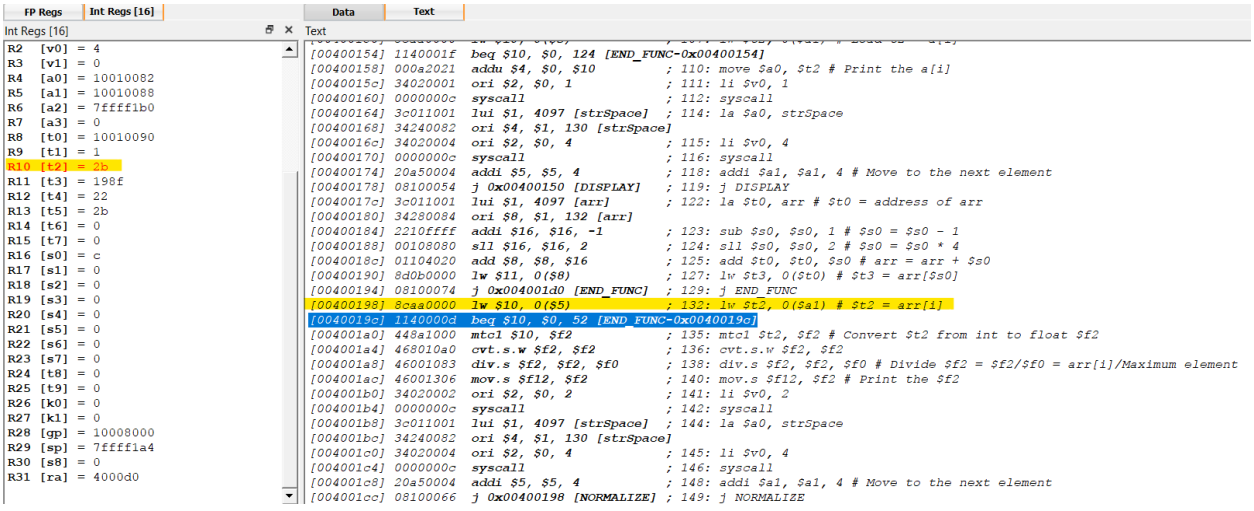

Int Regs [16]
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 10010080
R5 [a1] = 10010094
R6 [a2] = 7ffff1b0
R7 [a3] = 0
R8 [t0] = 10010090
R9 [t1] = 1
R10 [t2] = 0
R11 [t3] = 198f
R12 [t4] = 22
R13 [t5] = 2b
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = c
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff1a4
R30 [s8] = 0
R31 [ra] = 4000ac
Text
[0040011c] 00ab5820 add $11, $5, $11 ; 87: add $t3, $a1, $t3 # t3 = a1 + t3 = a[i]
[00400120] 21290001 addi $9, $9, 1 ; 89: addi $t1, $t1, 1 # Increase the counter of INNER_LOOP
[00400124] 8d6c0000 lw $12, 0($11) ; 91: lw $t4, 0($t3)
[00400128] 8d6d0004 lw $13, 4($11) ; 92: lw $t5, 4($t3)
[0040012c] 01ac082a slt $1, $13, $12 ; 94: ble $t4, $t5, INNER_LOOP # if t4
[00400130] 1020fff5 beq $1, $0, -44 [INNER_LOOP-0x00400130]
[00400134] ad6d0000 sw $13, 0($11) ; 96: sw $t5, 0($t3)
[00400138] ad6c0004 sw $12, 4($11) ; 97: sw $t4, 4($t3)
[0040013c] 34190001 ori $25, $0, 1 ; 98: li $t9, 1 # t9 = 1
[00400140] 08100041 j 0x00400104 [INNER_LOOP]; 99: j INNER_LOOP
[00400144] 13200023 beq $25, $0, 140 [END_FUNC-0x00400144]
[00400148] 21080001 addi $8, $8, 1 ; 103: addi $t0, $t0, 1 # Increase the counter of OUTER_LOOP
[0040014c] 0810003c j 0x004000f0 [OUTER_LOOP]; 104: j OUTER_LOOP
[00400150] 8caa0000 lw $10, 0($5) ; 107: lw $t2, 0($a1) # Load t2 = a[i]
[00400154] 1140001f beq $10, $0, 124 [END_FUNC-0x00400154]
[00400158] 000a2021 addu $4, $0, $10 ; 110: move $a0, $t2 # Print the a[i]
[0040015c] 34020001 ori $2, $0, 1 ; 111: li $v0, 1
[00400160] 0000000c syscall ; 112: syscall
[00400164] 3c011001 lui $1, 4097 [strSpace] ; 114: la $a0, strSpace
[00400168] 34240082 ori $4, $1, 130 [strSpace]
[0040016c] 34020004 ori $2, $0, 4 ; 115: li $v0, 4
[00400170] 0000000c syscall ; 116: syscall
[00400174] 20a50004 addi $5, $5, 4 ; 118: addi $a1, $a1, 4 # Move to the next element
[00400178] 08100054 j 0x00400150 [DISPLAY] ; 119: j DISPLAY
[0040017c] 3c011001 lui $1, 4097 [arr] ; 122: la $t0, arr # $t0 = address of arr
[00400180] 34280084 ori $8, $1, 132 [arr]
[00400184] 2210ffff addi $16, $16, -1 ; 123: sub $s0, $s0, 1 # $s0 = $s0 - 1
[00400188] 00108080 sll $16, $16, 2 ; 124: sll $s0, $s0, 2 # $s0 = $s0 * 4
[0040018c] 01104020 add $8, $8, $16 ; 125: add $t0, $t0, $s0 # arr = arr + $s0
[00400190] 8d0b0000 lw $11, 0($8) ; 127: lw $t3, 0($t0) # $t3 = arr[$s0]
[00400194] 08100074 j 0x004001d0 [END_FUNC] ; 129: j END_FUNC

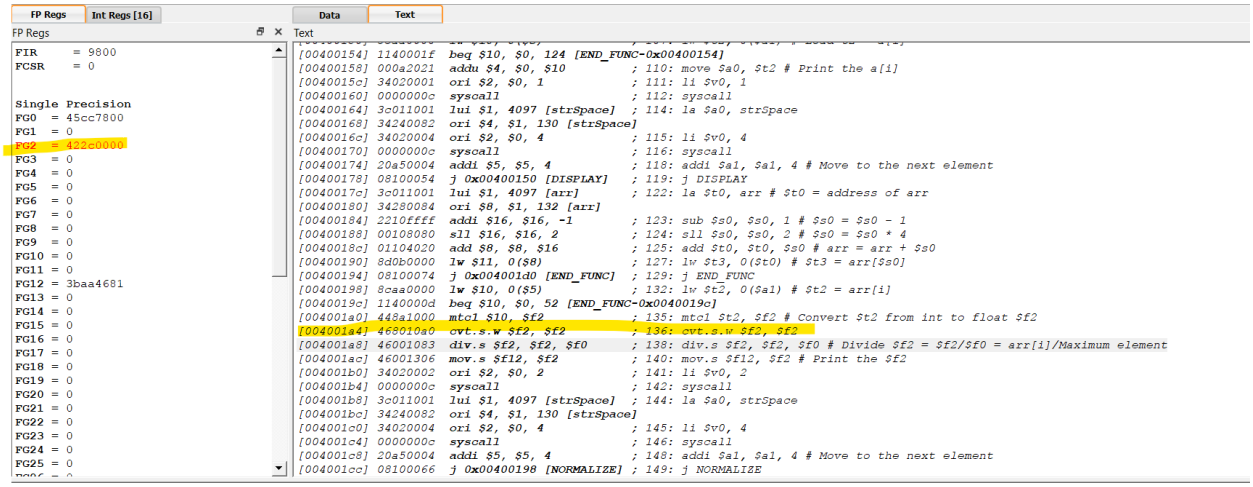
Normalize display steps:

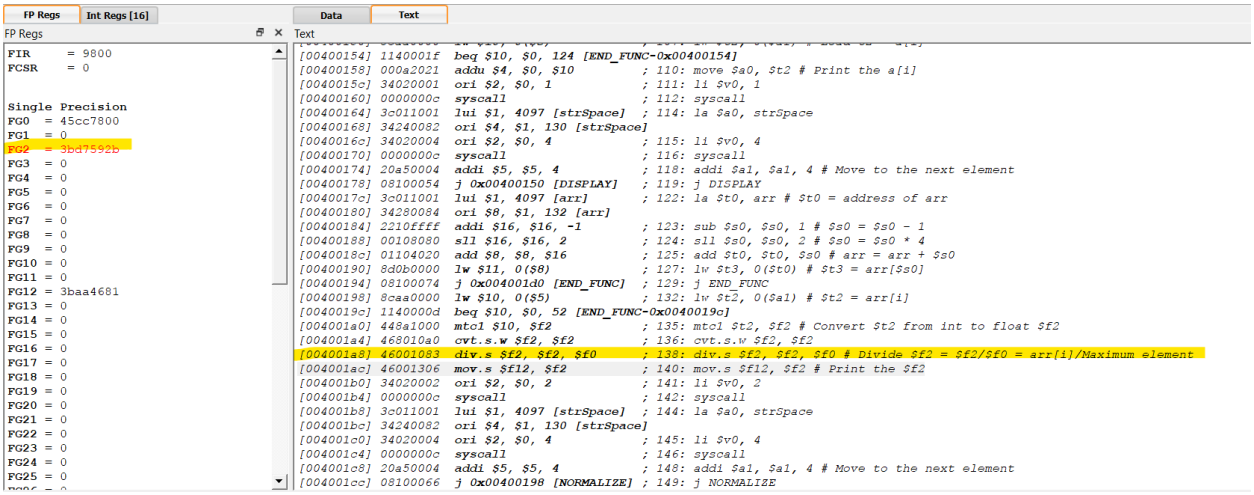


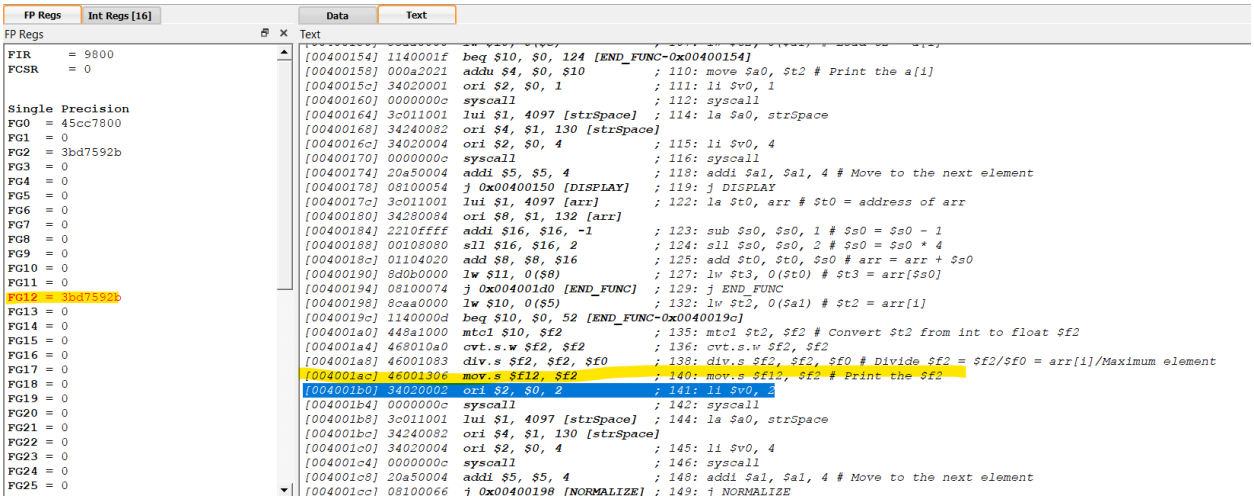

FP Regs
FIR = 9800
FCSR = 0
Single Precision
FG0 = 45cc7800
FG1 = 0
FG2 = 3baa4681
FG3 = 0
FG4 = 0
FG5 = 0
FG6 = 0
FG7 = 0
FG8 = 0
FG9 = 0
FG10 = 0
FG11 = 0
FG12 = 0
FG13 = 0
FG14 = 0
FG15 = 0
FG16 = 0
FG17 = 0
FG18 = 0
FG19 = 0
FG20 = 0
FG21 = 0
FG22 = 0
FG23 = 0
FG24 = 0
FG25 = 0
Text
[00400134] ad6d0000 sw $13, 0($11) ; 96: sw $t5, 0($t3)
[00400138] ad6c0004 sw $12, 4($11) ; 97: sw $t4, 4($t3)
[0040013c] 34190001 ori $25, $0, 1 ; 98: li $t9, 1 # t9 = 1
[00400140] 08100041 j 0x00400104 [INNER_LOOP]; 99: j INNER_LOOP
[00400144] 13200023 beq $25, $0, 140 [END_FUNC-0x00400144]
[00400148] 21080001 addi $8, $8, 1 ; 103: addi $t0, $t0, 1 # Increase the counter of OUTER_LOOP
[0040014c] 0810003c j 0x0040000f0 [OUTER_LOOP]; 104: j OUTER_LOOP
[00400150] 8caa0000 lw $10, 0($5) ; 107: lw $t2, 0($a1) # Load t2 = a[i]
[00400154] 1140001f beq $10, $0, 124 [END_FUNC-0x00400154]
[00400158] 000a2021 addu $4, $0, $10 ; 110: move $a0, $t2 # Print the a[i]
[0040015c] 34020001 ori $2, $0, 1 ; 111: li $v0, 1
[00400160] 0000000c syscall ; 112: syscall
[00400164] 3c011001 lui $1, 4097 [strSpace] ; 114: la $a0, strSpace
[00400168] 34240082 ori $4, $1, 130 [strSpace]
[0040016c] 34020004 ori $2, $0, 4 ; 115: li $v0, 4
[00400170] 0000000c syscall ; 116: syscall
[00400174] 20a50004 addi $5, $5, 4 ; 118: addi $a1, $a1, 4 # Move to the next element
[00400178] 08100054 j 0x00400150 [DISPLAY] ; 119: j DISPLAY
[0040017c] 3c011001 lui $1, 4097 [arr] ; 122: la $t0, arr # $t0 = address of arr
[00400180] 34280084 ori $8, $1, 132 [arr]
[00400184] 2210ffff addi $16, $16, -1 ; 123: sub $s0, $s0, 1 # $s0 = $s0 - 1
[00400188] 00108080 sll $16, $16, 2 ; 124: sll $s0, $s0, 2 # $s0 = $s0 * 4
[0040018c] 01104020 add $8, $8, $16 ; 125: add $t0, $t0, $s0 # arr = arr + $s0
[00400190] 8d0b0000 lw $11, 0($8) ; 127: lw $t3, 0($t0) # $t3 = arr[$s0]
[00400194] 08100074 j 0x004001d0 [END_FUNC] ; 129: j END_FUNC
[00400198] 8caa0000 lw $10, 0($5) ; 132: lw $t2, 0($a1) # $t2 = arr[i]
[0040019c] 1140000d beq $10, $0, 52 [END_FUNC-0x0040019c]
[004001a0] 448a1000 mtc1 $10, $f2 ; 135: mtc1 $t2, $f2 # Convert $t2 from int to float $f2
[004001a4] 468010a0 cvt.s.w $f2, $f2 ; 136: cvt.s.w $f2, $f2
[004001a8] 46001083 div.s $f2, $f2, $f0 ; 138: div.s $f2, $f2, $f0 # Divide $f2 = $f2/$f0 = arr[i]/Maximum element
[004001ac] 46001306 mov.s $f12, $f2 ; 140: mov.s $f12, $f2 # Print the $f2


FP Regs
FIR = 9800
FCSR = 0
Single Precision
FG0 = 45cc7800
FG1 = 0
FG2 = 3baa4681
FG3 = 0
FG4 = 0
FG5 = 0
FG6 = 0
FG7 = 0
FG8 = 0
FG9 = 0
FG10 = 0
FG11 = 0
FG12 = 3baa4681
FG13 = 0
FG14 = 0
FG15 = 0
FG16 = 0
FG17 = 0
FG18 = 0
FG19 = 0
FG20 = 0
FG21 = 0
FG22 = 0
FG23 = 0
FG24 = 0
FG25 = 0
Text
[00400138] ad6c0004 sw $12, 4($11) ; 97: sw $t4, 4($t3)
[0040013c] 34190001 ori $25, $0, 1 ; 98: li $t9, 1 # t9 = 1
[00400140] 08100041 j 0x00400104 [INNER_LOOP]; 99: j INNER_LOOP
[00400144] 13200023 beq $25, $0, 140 [END_FUNC-0x00400144]
[00400148] 21080001 addi $8, $8, 1 ; 103: addi $t0, $t0, 1 # Increase the counter of OUTER_LOOP
[0040014c] 0810003c j 0x0040000f0 [OUTER_LOOP]; 104: j OUTER_LOOP
[00400150] 8caa0000 lw $10, 0($5) ; 107: lw $t2, 0($a1) # Load t2 = a[i]
[00400154] 1140001f beq $10, $0, 124 [END_FUNC-0x00400154]
[00400158] 000a2021 addu $4, $0, $10 ; 110: move $a0, $t2 # Print the a[i]
[0040015c] 34020001 ori $2, $0, 1 ; 111: li $v0, 1
[00400160] 0000000c syscall ; 112: syscall
[00400164] 3c011001 lui $1, 4097 [strSpace] ; 114: la $a0, strSpace
[00400168] 34240082 ori $4, $1, 130 [strSpace]
[0040016c] 34020004 ori $2, $0, 4 ; 115: li $v0, 4
[00400170] 0000000c syscall ; 116: syscall
[00400174] 20a50004 addi $5, $5, 4 ; 118: addi $a1, $a1, 4 # Move to the next element
[00400178] 08100054 j 0x00400150 [DISPLAY] ; 119: j DISPLAY
[0040017c] 3c011001 lui $1, 4097 [arr] ; 122: la $t0, arr # $t0 = address of arr
[00400180] 34280084 ori $8, $1, 132 [arr]
[00400184] 2210ffff addi $16, $16, -1 ; 123: sub $s0, $s0, 1 # $s0 = $s0 - 1
[00400188] 00108080 sll $16, $16, 2 ; 124: sll $s0, $s0, 2 # $s0 = $s0 * 4
[0040018c] 01104020 add $8, $8, $16 ; 125: add $t0, $t0, $s0 # arr = arr + $s0
[00400190] 8d0b0000 lw $11, 0($8) ; 127: lw $t3, 0($t0) # $t3 = arr[$s0]
[00400194] 08100074 j 0x004001d0 [END_FUNC] ; 129: j END_FUNC
[00400198] 8caa0000 lw $10, 0($5) ; 132: lw $t2, 0($a1) # $t2 = arr[i]
[0040019c] 1140000d beq $10, $0, 52 [END_FUNC-0x0040019c]
[004001a0] 448a1000 mtc1 $10, $f2 ; 135: mtc1 $t2, $f2 # Convert $t2 from int to float $f2
[004001a4] 468010a0 cvt.s.w $f2, $f2 ; 136: cvt.s.w $f2, $f2
[004001a8] 46001083 div.s $f2, $f2, $f0 ; 138: div.s $f2, $f2, $f0 # Divide $f2 = $f2/$f0 = arr[i]/Maximum element
[004001ac] 46001306 mov.s $f12, $f2 ; 140: mov.s $f12, $f2 # Print the $f2
[004001b0] 34020002 ori $2, $0, 2 ; 141: li $v0, 2


FP Regs
Int Regs [16]
Data
Text
Int Regs [16]
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 10010082
R5 [a1] = 10010088
R6 [a2] = 7ffff1b0
R7 [a3] = 0
R8 [t0] = 10010090
R9 [t1] = 1
R10 [t2] = 2b
R11 [t3] = 198f
R12 [t4] = 22
R13 [t5] = 2b
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = c
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff1a4
R30 [s8] = 0
R31 [ra] = 4000d0
Text
[00400154] 1140001f beq $10, $0, 124 [END_FUNC-0x00400154]
[00400158] 000a2021 addu $4, $0, $10 ; 110: move $a0, $t2 # Print the a[i]
[0040015c] 34020001 ori $2, $0, 1 ; 111: li $v0, 1
[00400160] 0000000c syscall ; 112: syscall
[00400164] 3c011001 lui $1, 4097 [strSpace] ; 114: la $a0, strSpace
[00400168] 34240082 ori $4, $1, 130 [strSpace]
[0040016c] 34020004 ori $2, $0, 4 ; 115: li $v0, 4
[00400170] 0000000c syscall ; 116: syscall
[00400174] 20a50004 addi $5, $5, 4 ; 118: addi $a1, $a1, 4 # Move to the next element
[00400178] 08100054 j 0x00400150 [DISPLAY] ; 119: j DISPLAY
[0040017c] 3c011001 lui $1, 4097 [arr] ; 122: la $t0, arr # $t0 = address of arr
[00400180] 34280084 ori $8, $1, 132 [arr]
[00400184] 2210ffff addi $16, $16, -1 ; 123: sub $s0, $s0, 1 # $s0 = $s0 - 1
[00400188] 00108080 sll $16, $16, 2 ; 124: sll $s0, $s0, 2 # $s0 = $s0 * 4
[0040018c] 01104020 add $8, $8, $16 ; 125: add $t0, $t0, $s0 # arr = arr + $s0
[00400190] 8d0b0000 lw $11, 0($8) ; 127: lw $t3, 0($t0) # $t3 = arr[$s0]
[00400194] 08100074 j 0x004001d0 [END_FUNC] ; 129: j END_FUNC
[00400198] 8caa0000 lw $10, 0($5) ; 132: lw $t2, 0($a1) # $t2 = arr[i]
[0040019c] 1140000d beq $10, $0, 52 [END_FUNC-0x0040019c]
[004001a0] 448a1000 mtc1 $10, $f2 ; 135: mtc1 $t2, $f2 # Convert $t2 from int to float $f2
[004001a4] 468010a0 cvt.s.w $f2, $f2 ; 136: cvt.s.w $f2, $f2
[004001a8] 46001083 div.s $f2, $f2, $f0 ; 138: div.s $f2, $f2, $f0 # Divide $f2 = $f2/$f0 = arr[i]/Maximum element
[004001ac] 46001306 mov.s $f12, $f2 ; 140: mov.s $f12, $f2 # Print the $f2
[004001b0] 34020002 ori $2, $0, 2 ; 141: li $v0, 2
[004001b4] 0000000c syscall ; 142: syscall
[004001b8] 3c011001 lui $1, 4097 [strSpace] ; 144: la $a0, strSpace
[004001bc] 34240082 ori $4, $1, 130 [strSpace]
[004001c0] 34020004 ori $2, $0, 4 ; 145: li $v0, 4
[004001c4] 0000000c syscall ; 146: syscall
[004001c8] 20a50004 addi $5, $5, 4 ; 148: addi $a1, $a1, 4 # Move to the next element
[004001cc] 08100066 j 0x00400198 [NORMALIZE] ; 149: j NORMALIZE







It will load, divide and print the normalize element repeatedly until $t2 = a[i] = 0