

Name: Phan Trần Thanh Huy

ID: ITCSIU22056

## Lab 9

### Exercise 1:

#### Code:

```
 1 .data
 2 strNameId: .asciiz "Phan Tran Thanh Huy\nITCSIU22056\n"
 3 printArray: .asciiz "The initial array: "
 4 printLength: .asciiz "The length of the array is: "
 5 strSpace: .asciiz " "
 6 strLine: .asciiz "\n"
 7 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
 8 a: .float 0.0
 9
10 .text
11 .globl main
12
13 main:
14     la $a0, strNameId
15     li $v0, 4
16     syscall
17
18     subu $sp, $sp, 4      # Allocate space on the stack
19     sw $ra, 0($sp)        # Save $ra on the stack
20
21     la $t1, arr           # t1 = address of arr
22     li $s0, 0              # Counter = 0
23
24     l.s $f0, a            # get f0 = a = 0.0
25
26     jal LEN                # Compute the length of the array
27
28     la $a0, printArray
29     li $v0, 4
30     syscall
31
32     la $a1, arr
33     jal DISPLAY
34
35     la $a0, strLine
36     li $v0, 4
37     syscall
38
39     la $a0, printLength
40     li $v0, 4
41     syscall
42
43     move $a0, $s0          # Print the length of the array
44     li $v0, 1
45     syscall
46
47     li $v0, 10
48     syscall
49
50 LEN:
51     lwcl $f2, 0($t1)       # Load the first float from the array
52
53     c.eq.s $f2, $f0          # Check if the float is 0.0 (end of array)
54     bclt END_FUNC            # If true, go to END_FUNC
55
56     addi $t1, $t1, 4          # Move to the next element
57
58     subu $sp, $sp, 4          # Allocate space on the stack
59     sw $ra, 0($sp)          # Save $ra on the stack
60
61     jal LEN                # Call the recursive
62
63     lw $ra, 0($sp)          # Load $ra on the stack
64     addu $sp, $sp, 4          # Move to the next element
65
66     addi $s0, $s0, 1          # Increase counter
67
68     j $ra                  # Return the $ra
69
70 DISPLAY:
71     lwcl $f2, 0($a1)
72
73     c.eq.s $f2, $f0          # check if the float is 0.0 (end of array)
74     bclt END_FUNC            # if true, go to END_FUNC
75
76     mov.s $f12, $f2
77     li $v0, 2
78     syscall
79
80     la $a0, strSpace
81     li $v0, 4
82     syscall
83
84     addi $a1, $a1, 4
85     j DISPLAY
86
87 END_FUNC:
88     j $ra
89
90
```

## Test case:

Console

```

Phan Tran Thanh Huy
ITCSIU22056
The initial array: 43.00999832 34.06000137 22056.08984375 78.90000153 64.98000336 2126.10009766
643.20001221 451.39999390 423.39999390 45.50000000 566.00000000
The length of the array is: 11

```

## Single Step:

Int Regs [16]

R2	[v0] = 4
R3	[v1] = 0
R4	[a0] = 10010000
R5	[a1] = 7ffff1a0
R6	[a2] = 7ffff1a8
R7	[a3] = 0
R8	[t0] = 0
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
R22	[s6] = 0
R23	[s7] = 0
R24	[t8] = 0
R25	[t9] = 0
R26	[k0] = 0
R27	[k1] = 0
R28	[gp] = 10008000
R29	[sp] = 7ffff108
R30	[s8] = 0
R31	[ra] = 400018

Text

```

00400000 8fa40000 lw $4, 0($29) ; 183: lw $a0 0($sp) # argc
00400004 27a50004 addiu $5, $29, 4 ; 184: addiu $a1 $sp 4 # argv
00400008 24a60004 addiu $6, $5, 4 ; 185: addiu $a2 $a1 4 # envp
0040000c 00041080 sll $2, $4, 2 ; 186: sll $v0 $a2 2
00400010 00c23021 addu $6, $6, $2 ; 187: addu $a2 $a2 $v0
00400014 0c100009 jal 0x00400024 [main] ; 188: jal main
00400018 00000000 nop ; 189: nop
0040001c 3402000a ori $2, $0, 10 ; 190: li $v0 10
00400020 0000000c syscall ; 192: syscall # syscall 10 (exit)
00400024 3c041001 lui $4, 4097 [strNameId] ; 14: la $a0, strNameId
00400028 3c011001 lui $1, 4097 [arr] ; 21: la $t1, arr # t1 = address of arr
00400032 34290058 ori $9, $1, 88 [arr] ; 22: li $s0, 0 # Counter = 0
00400036 34100000 ori $16, $0, 0 ; 23: 1.s $f0, a # get f0 = a = 0.0
00400040 34100000 ori $16, $0, 0 ; 24: 1.s $f0, a # get f0 = a = 0.0
00400044 3c011001 lui $1, 4097 ; 25: la $a1, arr
00400048 c4200088 lwcl $f0, 136($1) ; 26: jal LEN # Compute the length of the array
0040004c 0c100028 jal 0x004000a0 [LEN] ; 26: jal LEN # Compute the length of the array
00400050 3c011001 lui $1, 4097 [printArray] ; 28: la $a0, printArray
00400054 34240021 ori $4, $1, 33 [printArray] ; 29: li $v0, 4
00400058 34020004 ori $2, $0, 4 ; 30: syscall
0040005c 00000000 syscall ; 30: syscall
00400060 3c011001 lui $1, 4097 [arr] ; 32: la $a1, arr
00400064 34250058 ori $5, $1, 88 [arr] ; 33: jal DISPLAY
00400068 0c100033 jal 0x004000cc [DISPLAY] ; 33: jal DISPLAY
0040006c 3c011001 lui $1, 4097 [strLine] ; 35: la $a0, strLine
00400070 34240054 ori $4, $1, 84 [strLine] ; 36: li $v0, 4
00400074 34020004 ori $2, $0, 4 ; 36: li $v0, 4

```

User Text Segment [00400000]..[00440000]

Int Regs [16]

PC	= 40003c
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] = 10010000
R5	[a1] = 7ffff1a0
R6	[a2] = 7ffff1a8
R7	[a3] = 0
R8	[t0] = 0
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

```

00400000 8fa40000 lw $4, 0($29) ; 183: lw $a0 0($sp) # argc
00400004 27a50004 addiu $5, $29, 4 ; 184: addiu $a1 $sp 4 # argv
00400008 24a60004 addiu $6, $5, 4 ; 185: addiu $a2 $a1 4 # envp
0040000c 00041080 sll $2, $4, 2 ; 186: sll $v0 $a2 2
00400010 00c23021 addu $6, $6, $2 ; 187: addu $a2 $a2 $v0
00400014 0c100009 jal 0x00400024 [main] ; 188: jal main
00400018 00000000 nop ; 189: nop
0040001c 3402000a ori $2, $0, 10 ; 190: li $v0 10
00400020 0000000c syscall ; 192: syscall # syscall 10 (exit)
00400024 3c041001 lui $4, 4097 [strNameId] ; 14: la $a0, strNameId
00400028 34240021 ori $4, $1, 33 [printArray] ; 21: la $t1, arr # t1 = address of arr
00400032 34290058 ori $9, $1, 88 [arr] ; 22: li $s0, 0 # Counter = 0
00400036 34100000 ori $16, $0, 0 ; 23: 1.s $f0, a # get f0 = a = 0.0
00400040 34100000 ori $16, $0, 0 ; 24: 1.s $f0, a # get f0 = a = 0.0
00400044 3c011001 lui $1, 4097 [arr] ; 25: la $a1, arr
00400048 c4200088 lwcl $f0, 136($1) ; 26: jal LEN # Compute the length of the array
0040004c 0c100028 jal 0x004000a0 [LEN] ; 26: jal LEN # Compute the length of the array
00400050 3c011001 lui $1, 4097 [printArray] ; 28: la $a0, printArray
00400054 34240021 ori $4, $1, 33 [printArray] ; 29: li $v0, 4
00400058 34020004 ori $2, $0, 4 ; 30: syscall
0040005c 00000000 syscall ; 30: syscall
00400060 3c011001 lui $1, 4097 [arr] ; 32: la $a1, arr
00400064 34250058 ori $5, $1, 88 [arr] ; 33: jal DISPLAY
00400068 0c100033 jal 0x004000cc [DISPLAY] ; 33: jal DISPLAY
0040006c 3c011001 lui $1, 4097 [strLine] ; 35: la $a0, strLine
00400070 34240054 ori $4, $1, 84 [strLine] ; 36: li $v0, 4
00400074 34020004 ori $2, $0, 4 ; 36: li $v0, 4

```

User Text Segment [00400000]..[00440000]

Int Regs [16]

PC	= 400040
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] = 10010000
R5	[a1] = 7fffff1a0
R6	[a2] = 7fffff1a8
R7	[a3] = 0
R8	[t0] = 0
R9	[t1] = 10010050
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] = 7fffff198
R30	[s8] = 0
R31	[ra] = 400050

Text

```

[00400000] 8f4a0000 lw $4, 0($29) ; 183: lw $a0, 0($sp) # argc
[00400004] 27a50004 addiu $5, $29, 4 ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004 addiu $6, $5, 4 ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080 sll $2, $4, 2 ; 186: sll $v0 $a0 2
[00400010] 00c23021 addu $6, $6, $2 ; 187: addu $a2 $a2 $v0
[00400014] 0c100000 jal 0x00400024 [main] ; 188: jal main
[00400018] 00000000 nop ; 189: nop
[0040001c] 24020000 ori $2, $0, 10 ; 191: li $v0 10
[00400020] 0000000c syscall ; 192: syscall # syscall 10 (exit)
[00400024] 3c041001 lui $4, 4097 [strNameId] ; 14: la $a0, strNameId
[00400028] 34020004 ori $2, $0, 4 ; 15: li $v0, 4
[0040002c] 0000000c syscall ; 16: syscall
[00400030] 27bdffff addiu $29, $29, -4 ; 18: subu $sp, $sp, 4 # Allocate space on the stack
[00400034] afbf0000 sw $31, 0($29) ; 19: sw $ra, 0($sp) # Save $ra on the stack
[00400038] 3c011001 lui $1, 4097 [arr] ; 21: la $t1, arr # t1 = address of arr
[0040003c] 34290058 ori $9, $1, 88 [arr] ; 22: li $s0, 0 # Counter = 0
[00400040] 34100000 ori $16, $0, 0 ; 24: 1.s $f0, a # get f0 = a = 0.0
[00400044] 3c011001 lui $1, 4097 [arr] ; 24: 1.s $f0, a # get f0 = a = 0.0
[00400048] c4200088 lwcl $f0, 136($1) ; 26: jal LEN # Compute the length of the array
[00400050] 3c011001 lui $1, 4097 [printArray] ; 28: la $a0, printArray
[00400054] 34240021 ori $4, $1, 33 [printArray] ; 29: li $v0, 4
[00400058] 34020004 ori $2, $0, 4 ; 30: syscall
[0040005c] 0000000c syscall ; 30: syscall
[00400060] 3c011001 lui $1, 4097 [arr] ; 32: la $a1, arr
[00400064] 34250058 ori $5, $1, 88 [arr] ; 32: la $a1, arr
[00400068] 0c100033 jal 0x0040000c [DISPLAY] ; 33: jal DISPLAY
[0040006c] 3c011001 lui $1, 4097 [strLine] ; 35: la $a0, strLine
[00400070] 34240054 ori $4, $1, 84 [strLine] ; 36: li $v0, 4
[00400074] 34020004 ori $2, $0, 4 ; 37: syscall
[00400078] 0000000c syscall ; 37: syscall
[0040007c] 3c011001 lui $1, 4097 [printLength] ; 39: la $a0, printLength
[00400080] 34240035 ori $4, $1, 53 [printLength] ; 40: li $v0, 4
[00400084] 34020004 ori $2, $0, 4 ; 41: syscall
[00400088] 0000000c syscall ; 41: syscall
[0040008c] 0c102021 addu $4, $0, $16 ; 43: move $a0, $s0 # Print the length of the array
[00400090] 34020001 ori $2, $0, 1 ; 44: li $v0, 1
[00400094] 0000000c syscall ; 45: syscall
[00400098] 34020000 ori $2, $0, 10 ; 47: li $v0, 10
[0040009c] 0000000c syscall ; 48: syscall
[004000a0] c5220000 lwcl $f2, 0($t1) # Load the first float from the array ; 51: lwcl $f2, 0($t1) # Load the first float from the array

```

## Save the first element to the stack:

Int Regs [16]

R2	[v0] = 4
R3	[v1] = 0
R4	[a0] = 10010000
R5	[a1] = 7fffff1a0
R6	[a2] = 7fffff1a8
R7	[a3] = 0
R8	[t0] = 0
R9	[t1] = 10010050
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] = 7fffff198
R30	[s8] = 0
R31	[ra] = 400050

Text

```

[00400028] 34020004 ori $2, $0, 4 ; 15: li $v0, 4
[0040002c] 0000000c syscall ; 16: syscall
[00400030] 27bdffff addiu $29, $29, -4 ; 18: subu $sp, $sp, 4 # Allocate space on the stack
[00400034] afbf0000 sw $31, 0($29) ; 19: sw $ra, 0($sp) # Save $ra on the stack
[00400038] 3c011001 lui $1, 4097 [arr] ; 21: la $t1, arr # t1 = address of arr
[00400040] 34100000 ori $16, $0, 0 ; 22: li $s0, 0 # Counter = 0
[00400044] 3c011001 lui $1, 4097 [arr] ; 24: 1.s $f0, a # get f0 = a = 0.0
[00400048] c4200088 lwcl $f0, 136($1) ; 26: jal LEN # Compute the length of the array
[00400050] 3c011001 lui $1, 4097 [printArray] ; 28: la $a0, printArray
[00400054] 34240021 ori $4, $1, 33 [printArray] ; 29: li $v0, 4
[00400058] 34020004 ori $2, $0, 4 ; 30: syscall
[0040005c] 0000000c syscall ; 30: syscall
[00400060] 3c011001 lui $1, 4097 [arr] ; 32: la $a1, arr
[00400064] 34250058 ori $5, $1, 88 [arr] ; 32: la $a1, arr
[00400068] 0c100033 jal 0x0040000c [DISPLAY] ; 33: jal DISPLAY
[0040006c] 3c011001 lui $1, 4097 [strLine] ; 35: la $a0, strLine
[00400070] 34240054 ori $4, $1, 84 [strLine] ; 36: li $v0, 4
[00400074] 34020004 ori $2, $0, 4 ; 37: syscall
[00400078] 0000000c syscall ; 37: syscall
[0040007c] 3c011001 lui $1, 4097 [printLength] ; 39: la $a0, printLength
[00400080] 34240035 ori $4, $1, 53 [printLength] ; 40: li $v0, 4
[00400084] 34020004 ori $2, $0, 4 ; 41: syscall
[00400088] 0000000c syscall ; 41: syscall
[0040008c] 0c102021 addu $4, $0, $16 ; 43: move $a0, $s0 # Print the length of the array
[00400090] 34020001 ori $2, $0, 1 ; 44: li $v0, 1
[00400094] 0000000c syscall ; 45: syscall
[00400098] 34020000 ori $2, $0, 10 ; 47: li $v0, 10
[0040009c] 0000000c syscall ; 48: syscall
[004000a0] c5220000 lwcl $f2, 0($t1) # Load the first float from the array ; 51: lwcl $f2, 0($t1) # Load the first float from the array

```

FP Regs

FIR	= 9800
FCSR	= 0
Single Precision	
F00	= 0
FG1	= 0
FG2	= 422c0a3d
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

```

[00400034] afbf0000 sw $31, 0($29) ; 19: sw $ra, 0($sp) # Save $ra on the stack
[00400038] 3c011001 lui $1, 4097 [arr] ; 21: la $t1, arr # t1 = address of arr
[0040003c] 34290058 ori $9, $1, 88 [arr] ; 22: li $s0, 0 # Counter = 0
[00400040] 34100000 ori $16, $0, 0 ; 24: 1.s $f0, a # get f0 = a = 0.0
[00400044] 3c011001 lui $1, 4097 [arr] ; 26: jal LEN # Compute the length of the array
[00400048] 0c100028 jal 0x004000a0 [LEN] ; 28: la $a0, printArray
[00400050] 3c011001 lui $1, 4097 [printArray] ; 28: la $a0, printArray
[00400054] 34240021 ori $4, $1, 33 [printArray] ; 29: li $v0, 4
[00400058] 34020004 ori $2, $0, 4 ; 30: syscall
[0040005c] 0000000c syscall ; 30: syscall
[00400060] 3c011001 lui $1, 4097 [arr] ; 32: la $a1, arr
[00400064] 34250058 ori $5, $1, 88 [arr] ; 32: la $a1, arr
[00400068] 0c100033 jal 0x0040000c [DISPLAY] ; 33: jal DISPLAY
[0040006c] 3c011001 lui $1, 4097 [strLine] ; 35: la $a0, strLine
[00400070] 34240054 ori $4, $1, 84 [strLine] ; 36: li $v0, 4
[00400074] 34020004 ori $2, $0, 4 ; 37: syscall
[00400078] 0000000c syscall ; 37: syscall
[0040007c] 3c011001 lui $1, 4097 [printLength] ; 39: la $a0, printLength
[00400080] 34240035 ori $4, $1, 53 [printLength] ; 40: li $v0, 4
[00400084] 34020004 ori $2, $0, 4 ; 41: syscall
[00400088] 0000000c syscall ; 41: syscall
[0040008c] 0c102021 addu $4, $0, $16 ; 43: move $a0, $s0 # Print the length of the array
[00400090] 34020001 ori $2, $0, 1 ; 44: li $v0, 1
[00400094] 0000000c syscall ; 45: syscall
[00400098] 34020000 ori $2, $0, 10 ; 47: li $v0, 10
[0040009c] 0000000c syscall ; 48: syscall
[004000a0] c5220000 lwcl $f2, 0($t1) # Load the first float from the array ; 51: lwcl $f2, 0($t1) # Load the first float from the array
[004000a4] 46001032 c.eq.s $f2, $f0 ; 53: c.eq.s $f2, $f0 # Check if the float is 0.0 (end of array)
[004000a8] 45010015 bclt $4 [END_FUNC-0x004000a8]; 54: bclt END_FUNC # If true, go to END_FUNC
[004000ac] 21290004 addi $9, $9, 4 ; 56: addi $t1, $t1, 4 # Move to the next element

```

Int Regs [16]

Text

```

PC = 4000b0
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] = 10010000
R5 [a1] = 7fffff1a0
R6 [a2] = 7fffff1a8
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = 1001005c
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

[00400038] 3c011001 lui $1, 4097 [arr] ; 21: la $t1, arr # t1 = address of arr
[0040003c] 34290058 ori $9, $1, 88 [arr]
[00400040] 34100000 ori $16, $0, 0 ; 22: li $s0, 0 # Counter = 0
[00400044] 3c011001 lui $1, 4097 ; 24: 1.s $f0, a # get f0 = a = 0.0
[00400048] c4200088 lwc1 $f0, 136($1)
[0040004c] 0c100028 jal 0x004000a0 [LEN] ; 26: jal LEN # Compute the length of the array
[00400050] 3c011001 lui $1, 4097 [printArray]; 28: la $a0, printArray
[00400054] 34240021 lui $4, $1, 33 [printArray]
[00400058] 34020004 ori $2, $0, 4 ; 29: li $v0, 4
[00400060] 3c011001 lui $1, 4097 [arr] ; 30: syscall
[00400064] 34250058 ori $5, $1, 88 [arr]
[00400068] 0c100033 jal 0x004000cc [DISPLAY] ; 33: jal DISPLAY
[0040006c] 3c011001 lui $1, 4097 [strLine] ; 35: la $a0, strLine
[00400070] 34240054 ori $4, $1, 84 [strLine]
[00400074] 34020004 ori $2, $0, 4 ; 36: li $v0, 4
[00400078] 0000000c syscall ; 37: syscall
[00400080] 34240035 lwc1 $f2, 0($9) ; 51: lwc1 $f2, 0($t1) # Load the first float from the array
[00400084] 46001032 c.eq.s $f2, $f0 ; 53: c.eq.s $f2, $f0 # Check if the float is 0.0 (end of array)
[00400088] 45010015 bclt 84 [END_FUNC-0x004000a8]; 54: bclt END_FUNC # If true, go to END_FUNC
[004000ac] 21290004 addi $9, $9, 4 ; 56: addi $t1, $t1, 4 # Move to the next element
[004000b0] 27bdfffc addiu $29, $29, -4 ; 58: subu $sp, $sp, 4 # Allocate space on the stack
[004000b4] afbf0000 sw $31, 0($29) ; 59: sw $ra, 0($sp) # Save $ra on the stack
[004000b8] 0c100028 jal 0x004000a0 [LEN] ; 61: jal LEN # Call the recursive

```

Int Regs [16]

Text

```

R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 10010000
R5 [a1] = 7fffff1a0
R6 [a2] = 7fffff1a8
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = 1001005c
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] = 7fffff194
R30 [ra] = 400050
R31 [ra] = 40005c

[0040003c] 34290058 ori $9, $1, 88 [arr]
[00400040] 34100000 ori $16, $0, 0 ; 22: li $s0, 0 # Counter = 0
[00400044] 3c011001 lui $1, 4097 ; 24: 1.s $f0, a # get f0 = a = 0.0
[00400048] c4200088 lwc1 $f0, 136($1)
[00400050] 0c100028 jal 0x004000a0 [LEN] ; 26: jal LEN # Compute the length of the array
[00400054] 3c011001 lui $1, 4097 [printArray]; 28: la $a0, printArray
[00400058] 34240021 lui $4, $1, 33 [printArray]
[00400060] 34020004 ori $2, $0, 4 ; 29: li $v0, 4
[00400064] 3c011001 lui $1, 4097 [arr] ; 30: syscall
[00400068] 34250058 ori $5, $1, 88 [arr]
[0040006c] 0c100033 jal 0x004000cc [DISPLAY] ; 33: jal DISPLAY
[00400070] 3c011001 lui $1, 4097 [strLine] ; 35: la $a0, strLine
[00400074] 34240054 ori $4, $1, 84 [strLine]
[00400078] 34020004 ori $2, $0, 4 ; 36: li $v0, 4
[00400080] 0000000c syscall ; 37: syscall
[00400084] 46001032 c.eq.s $f2, $f0 ; 53: c.eq.s $f2, $f0 # Check if the float is 0.0 (end of array)
[00400088] 45010015 bclt 84 [END_FUNC-0x004000a8]; 54: bclt END_FUNC # If true, go to END_FUNC
[004000ac] 21290004 addi $9, $9, 4 ; 56: addi $t1, $t1, 4 # Move to the next element
[004000b0] 27bdfffc addiu $29, $29, -4 ; 58: subu $sp, $sp, 4 # Allocate space on the stack
[004000b4] afbf0000 sw $31, 0($29) ; 59: sw $ra, 0($sp) # Save $ra on the stack
[004000b8] 0c100028 jal 0x004000a0 [LEN] ; 61: jal LEN # Call the recursive

```

Save the second element to the stack:

Int Regs [16]

Text

```

R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 10010000
R5 [a1] = 7fffff1a0
R6 [a2] = 7fffff1a8
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = 1001005c
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] = 7fffff194
R30 [ra] = 400050
R31 [ra] = 40005c

[00400040] 34100000 ori $16, $0, 0 ; 22: li $s0, 0 # Counter = 0
[00400044] 3c011001 lui $1, 4097 ; 24: 1.s $f0, a # get f0 = a = 0.0
[00400048] c4200088 lwc1 $f0, 136($1)
[00400050] 0c100028 jal 0x004000a0 [LEN] ; 26: jal LEN # Compute the length of the array
[00400054] 3c011001 lui $1, 4097 [printArray]; 28: la $a0, printArray
[00400058] 34240021 lui $4, $1, 33 [printArray]
[00400060] 34020004 ori $2, $0, 4 ; 29: li $v0, 4
[00400064] 3c011001 lui $1, 4097 [arr] ; 30: syscall
[00400068] 34250058 ori $5, $1, 88 [arr]
[0040006c] 0c100033 jal 0x004000cc [DISPLAY] ; 33: jal DISPLAY
[00400070] 3c011001 lui $1, 4097 [strLine] ; 35: la $a0, strLine
[00400074] 34240054 ori $4, $1, 84 [strLine]
[00400078] 34020004 ori $2, $0, 4 ; 36: li $v0, 4
[00400080] 0000000c syscall ; 37: syscall
[00400084] 46001032 c.eq.s $f2, $f0 ; 53: c.eq.s $f2, $f0 # Check if the float is 0.0 (end of array)
[00400088] 45010015 bclt 84 [END_FUNC-0x004000a8]; 54: bclt END_FUNC # If true, go to END_FUNC
[004000ac] 21290004 addi $9, $9, 4 ; 56: addi $t1, $t1, 4 # Move to the next element
[004000b0] 27bdfffc addiu $29, $29, -4 ; 58: subu $sp, $sp, 4 # Allocate space on the stack
[004000b4] afbf0000 sw $31, 0($29) ; 59: sw $ra, 0($sp) # Save $ra on the stack
[004000b8] 0c100028 jal 0x004000a0 [LEN] ; 61: jal LEN # Call the recursive

```

FP Regs	Int Regs [16]	Data	Text
FIR	= 9800	[00400070] 34240054 ori \$4, \$1, 84 [strLine]	
FCSR	= 0	[00400074] 34020004 ori \$2, \$0, 4 ; 36: li \$v0, 4	
Single Precision		[00400078] 00000000 syscall ; 37: syscall	
FPG0	= 0	[0040007c] 3c011001 lui \$1, 4097 [printLength]; 39: la \$a0, printLength	
FG1	= 0	[00400080] 34240035 ori \$4, \$1, 53 [printLength]	
FG2	= 42083071	[00400084] 34020004 ori \$2, \$0, 4 ; 40: li \$v0, 4	
FG3	= 0	[00400088] 00000000 syscall ; 41: syscall	
FG4	= 0	[00400090] 34020001 addu \$4, \$0, \$16 ; 43: move \$a0, \$s0 # Print the length of the array	
FG5	= 0	[00400094] 00000000 ori \$2, \$0, 1 ; 44: li \$v0, 1	
FG6	= 0	[00400098] 34020004 ori \$2, \$0, 10 ; 45: syscall	
FG7	= 0	[0040009c] 00000000 syscall ; 46: syscall	
FG8	= 0	[004000a0] c52200000 lwc1 \$f2, 0(\$9) ; 51: lwc1 \$f2, 0(\$t1) # Load the first float from the array	
FG9	= 0	[004000a4] 46001032 c.eqs \$f2, \$f0 ; 53: c.eqs \$f2, \$f0 # Check if the float is 0.0 (end of array)	
FG10	= 0	[004000a8] 45010015 bc1to 84 [END_FUNC-0x004000a8] ; 54: bc1 END_FUNC # If true, go to END_FUNC	
FG11	= 0	[004000ac] 21290000 addi \$9, \$9, 4 ; 56: addi \$t1, \$t1, 4 # Move to the next element	
FG12	= 0	[004000b0] 27bdffff addiu \$29, \$29, -4 ; 58: subu \$sp, \$sp, 4 # Allocate space on the stack	
FG13	= 0	[004000b4] afbf0000 sw \$31, 0(\$29) ; 59: subu \$ra, 0(\$sp) # Save \$ra on the stack	
FG14	= 0	[004000b8] 0c100028 jal 0x004000a0 [LEN] ; 61: jal LEN # Call the recursive	
FG15	= 0	[004000bc] 8fb00000 lw \$31, 0(\$29) ; 63: lw \$ra, 0(\$sp) # Load \$ra on the stack	
FG16	= 0	[004000cc] 27bd0004 addi \$29, \$29, 4 ; 64: addu \$sp, \$sp, 4 # Move to the next element	
FG17	= 0	[004000d0] 22100001 addi \$16, \$16, 1 ; 66: addi \$s0, \$s0, 1 # Increase counter	
FG18	= 0	[004000d4] 03e00008 jr \$31 ; 68: j \$ra # Return the \$ra	
FG19	= 0	[004000d8] c4a20000 lwc1 \$f2, 0(\$5) ; 71: lwc1 \$f2, 0(\$a1)	
FG20	= 0	[004000e0] 46001032 c.eqs \$f2, \$f0 ; 73: c.eqs \$f2, \$f0 # Check if the float is 0.0 (end of array)	
FG21	= 0	[004000e4] 45010004 bc1to 40 [END_FUNC-0x004000d4] ; 74: bc1 END_FUNC # If true, go to END_FUNC	
FG22	= 0	[004000e8] 46001036 mov.s \$f12, \$f2 ; 76: mov.s \$f12, \$f2	
FG23	= 0	[004000f0] 34020002 ori \$2, \$0, 2 ; 77: li \$v0, 2	
FG24	= 0	[004000f4] 3c011001 lui \$1, 4097 [strSpace]; 80: la \$a0, strSpace	
FG25	= 0	[004000f8] 34240052 ori \$4, \$1, 82 [strSpace]	
...	...	[004000fc] 34020004 ori \$2, \$0, 4	
PC	= 4000b0		
EPIC	= 0	[00400070] 34240054 ori \$4, \$1, 84 [strLine]	
Cause	= 0	[00400074] 34020004 ori \$2, \$0, 4 ; 36: li \$v0, 4	
BadAddr	= 0	[00400078] 00000000 syscall ; 37: syscall	
Status	= 3000ff10	[0040007c] 3c011001 lui \$1, 4097 [printLength]; 39: la \$a0, printLength	
HI	= 0	[00400080] 34240035 ori \$4, \$1, 53 [printLength]	
LO	= 0	[00400084] 34020004 ori \$2, \$0, 4 ; 40: li \$v0, 4	
R0 [r0]	= 0	[00400088] 00000000 syscall ; 41: syscall	
R1 [at]	= 10010000	[00400090] 34020001 addu \$4, \$0, \$16 ; 43: move \$a0, \$s0 # Print the length of the array	
R2 [v0]	= 4	[00400094] 3c011001 ori \$2, \$0, 1 ; 44: li \$v0, 1	
R3 [v1]	= 0	[00400098] c52200000 lwc1 \$f2, 0(\$29) ; 51: lwc1 \$f2, 0(\$t1) # Load the first float from the array	
R4 [a0]	= 10010000	[0040009c] 046001032 c.eqs \$f2, \$f0 ; 53: c.eqs \$f2, \$f0 # Check if the float is 0.0 (end of array)	
R5 [at]	= 7fffffa0	[004000a0] 45010004 bc1to 84 [END_FUNC-0x004000a8] ; 54: bc1 END_FUNC # If true, go to END_FUNC	
R6 [a2]	= 7fffffa8	[004000a4] 27bdffff addi \$9, \$9, -4 ; 56: subu \$sp, \$sp, 4 # Move to the next element	
R7 [a3]	= 0	[004000a8] afbf0000 sw \$31, 0(\$29) ; 59: subu \$ra, 0(\$sp) # Save \$ra on the stack	
R8 [t0]	= 0	[004000b0] 27bdffff addiu \$29, \$29, -4 ; 61: jal LEN # Call the recursive	
R9 [t1]	= 0	[004000b4] 0c100028 jal 0x004000a0 [LEN] ; 63: lw \$ra, 0(\$sp) # Load \$ra on the stack	
R10 [t2]	= 0	[004000b8] 27bd0004 addiu \$29, \$29, 4 ; 64: addu \$sp, \$sp, 4 # Move to the next element	
R11 [t3]	= 0	[004000cc] 22100001 addi \$16, \$16, 1 ; 66: addi \$s0, \$s0, 1 # Increase counter	
R12 [t4]	= 0	[004000d0] 03e00008 jr \$31 ; 68: j \$ra # Return the \$ra	
R13 [t5]	= 0	[004000d4] c4a20000 lwc1 \$f2, 0(\$5) ; 71: lwc1 \$f2, 0(\$a1)	
R14 [t6]	= 0	[004000d8] 046001032 c.eqs \$f2, \$f0 ; 73: c.eqs \$f2, \$f0 # Check if the float is 0.0 (end of array)	
R15 [t7]	= 0	[004000e0] 45010004 bc1to 40 [END_FUNC-0x004000d4] ; 74: bc1 END_FUNC # If true, go to END_FUNC	
R16 [s0]	= 0	[004000e4] 46001306 mov.s \$f12, \$f2 ; 76: mov.s \$f12, \$f2	
R17 [s1]	= 0	[004000e8] 34020002 ori \$2, \$0, 2 ; 77: li \$v0, 2	
R18 [s2]	= 0	[004000f0] 00000000 syscall ; 78: syscall	
R19 [s3]	= 0	[004000f4] 3c011001 lui \$1, 4097 [strSpace]; 80: la \$a0, strSpace	
R20 [s4]	= 0	[004000f8] 34240052 ori \$4, \$1, 82 [strSpace]	
R21 [s5]	= 0	[004000fc] 34020004 ori \$2, \$0, 4	
Int Regs [16]			
R2 [v0]	= 4	[00400064] 34250058 ori \$5, \$1, 88 [arr]	
R3 [v1]	= 0	[00400068] 0c100033 jal 0x00400000 [DISPLAY]; 33: jal DISPLAY	
R4 [a0]	= 10010000	[0040006c] 3c011001 lui \$1, 4097 [strLine]	
R5 [at]	= 7fffffa0	[00400070] 34240054 ori \$4, \$1, 84 [strLine]	
R6 [a2]	= 7fffffa8	[00400074] 046001032 c.eqs \$f2, \$f0 ; 35: c.eqs \$f2, \$f0 # Check if the float is 0.0 (end of array)	
R7 [a3]	= 0	[00400078] 00000000 syscall ; 37: syscall	
R8 [t1]	= 10010060	[0040007c] 3c011001 lui \$1, 4097 [printLength]; 39: la \$a0, printLength	
R10 [t2]	= 0	[00400080] 34240035 ori \$4, \$1, 53 [printLength]	
R11 [t3]	= 0	[00400084] 34020004 ori \$2, \$0, 4 ; 40: li \$v0, 4	
R12 [t4]	= 0	[00400088] 00000000 syscall ; 41: syscall	
R13 [t5]	= 0	[00400090] 34020001 addu \$4, \$0, \$16 ; 43: move \$a0, \$s0 # Print the length of the array	
R14 [t6]	= 0	[00400094] 3c011001 ori \$2, \$0, 1 ; 44: li \$v0, 1	
R15 [t7]	= 0	[00400098] 34240000 ori \$2, \$0, 10 ; 45: syscall	
R16 [s0]	= 0	[0040009c] 00000000 syscall ; 46: syscall	
R17 [s1]	= 0	[004000a0] c52200000 lwc1 \$f2, 0(\$29) ; 51: lwc1 \$f2, 0(\$t1) # Load the first float from the array	
R18 [s2]	= 0	[004000a4] 046001032 c.eqs \$f2, \$f0 ; 53: c.eqs \$f2, \$f0 # Check if the float is 0.0 (end of array)	
R19 [s3]	= 0	[004000a8] 45010004 bc1to 84 [END_FUNC-0x004000a8]; 54: bc1 END_FUNC # If true, go to END_FUNC	
R20 [s4]	= 0	[004000b0] 27bdffff addi \$9, \$9, -4 ; 56: subu \$sp, \$sp, 4 # Move to the next element	
R21 [s5]	= 0	[004000b4] afbf0000 sw \$31, 0(\$29) ; 59: subu \$ra, 0(\$sp) # Save \$ra on the stack	
R22 [s6]	= 0	[004000b8] 0c100028 jal 0x004000a0 [LEN] ; 61: jal LEN # Call the recursive	
R23 [s7]	= 0	[004000bc] 8f600000 lw \$31, 0(\$29) ; 63: lw \$ra, 0(\$sp) # Load \$ra on the stack	
R24 [t8]	= 0	[004000cc] 27bd0004 addiu \$29, \$29, 4 ; 64: addu \$sp, \$sp, 4 # Move to the next element	
R25 [t9]	= 0	[004000cc] 22100001 addi \$16, \$16, 1 ; 66: addi \$s0, \$s0, 1 # Increase counter	
R26 [k0]	= 0	[004000cc] 03e00008 jr \$31 ; 68: j \$ra # Return the \$ra	
R27 [k1]	= 0	[004000cc] c4a20000 lwc1 \$f2, 0(\$5) ; 71: lwc1 \$f2, 0(\$a1)	
R28 [gm]	= 100008000	[004000cc] 046001032 c.eqs \$f2, \$f0 ; 73: c.eqs \$f2, \$f0 # Check if the float is 0.0 (end of array)	
R29 [sp]	= 7fffffa0	[004000cc] 45010004 bc1to 40 [END_FUNC-0x004000d4]; 74: bc1 END_FUNC # If true, go to END_FUNC	
R30 [s8]	= 0	[004000cc] 27bdffff addi \$9, \$9, -4 ; 76: subu \$sp, \$sp, 4 # Move to the next element	
R31 [ra]	= 4000bc	[004000cc] 0c000002 ori \$2, \$0, 2 ; 77: li \$v0, 2	
R32 [s9]	= 0	[004000cc] 00000000 syscall ; 79: syscall	

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

Int Regs [16]	Text
R2 [v0] = 4	[00400090] 34020001 ori \$2, \$0, 1 ; 44: li \$v0, 1
R3 [v1] = 0	[00400094] 0000000c syscall ; 45: syscall
R4 [a0] = 10010000	[00400098] 34020004 ori \$2, \$0, 10 ; 47: li \$v0, 10
R5 [a1] = 7ffff1a0	[0040009c] 0000000c syscall ; 48: syscall
R6 [a2] = 7ffff1a8	[004000a0] c5220000 lwc1 \$f2, 0(\$9) ; 51: lwc1 \$f2, 0(\$t1) # Load the first float from the array
R7 [a3] = 0	[004000a4] 46001032 c.eq.s \$f2, \$f0 ; 53: c.eq.s \$f2, \$f0 # Check if the float is 0.0 (end of array)
R8 [t0] = 0	[004000a8] 45010015 bclt 84 [END_FUNC-0x004000a8]; 54: bclt END_FUNC # If true, go to END_FUNC
R9 [t1] = 10010084	[004000ac] 21290004 addi \$9, \$9, 4 ; 56: addi \$t1, \$t1, 4 # Move to the next element
R10 [t2] = 0	[004000b0] 27bdffff addiu \$29, \$29, -4 ; 58: subu \$sp, \$sp, 4 # Allocate space on the stack
R11 [t3] = 0	[004000b4] afbf0000 sw \$31, 0(\$29) ; 59: sw \$ra, 0(\$sp) # Save \$ra on the stack
R12 [t4] = 0	[004000b8] 0c100028 jal 0x004000a0 [LEN] ; 61: jal LEN # Call the recursive
R13 [t5] = 0	[004000bc] 8fbff000 lw \$31, 0(\$29) ; 63: lw \$ra, 0(\$sp) # Load \$ra on the stack
R14 [t6] = 0	[004000c0] 27bd4004 addiu \$29, \$29, 4 ; 64: addu \$sp, \$sp, 4 # Move to the next element
R15 [t7] = 0	[004000c4] 22100001 addi \$16, \$16, 1 ; 66: addi \$s0, \$s0, 1 # Increase counter
R16 [s0] = 0	[004000c8] 03e00008 jr \$31 ; 68: j \$ra # Return the \$ra
R17 [s1] = 0	[004000cc] c4a20000 lwc1 \$f2, 0(\$5) ; 71: lwc1 \$f2, 0(\$a1)
R18 [s2] = 0	[004000d0] 46001032 c.eq.s \$f2, \$f0 ; 73: c.eq.s \$f2, \$f0 # Check if the float is 0.0 (end of array)
R19 [s3] = 0	[004000d4] 45010004 bclt 40 [END_FUNC-0x004000d4]; 74: bclt END_FUNC # If true, go to END_FUNC
R20 [s4] = 0	[004000d8] 46001306 mov.s \$12, \$f2 ; 76: mov.s \$f12, \$f2
R21 [s5] = 0	[004000dc] 34020002 ori \$2, \$0, 2 ; 77: li \$v0, 2
R22 [s6] = 0	[004000e0] 0000000c syscall ; 82: syscall
R23 [s7] = 0	[004000e4] 3c011001 lui \$1, 4097 [strSpace] ; 80: la \$a0, strSpace
R24 [t8] = 0	[004000e8] 34240052 ori \$4, \$1, 82 [strSpace] ; 81: li \$v0, 4
R25 [t9] = 0	[004000f0] 0000000c syscall ; 82: syscall
R26 [k0] = 0	[004000f4] 20a50004 addi \$5, \$5, 4 ; 84: addi \$a1, \$a1, 4
R27 [k1] = 0	[004000f8] 08100033 j 0x004000cc [DISPLAY] ; 85: j DISPLAY
R28 [gp] = 10008000	[004000fc] 03e00008 jr \$31 ; 88: j \$ra
R29 [sp] = 7ffff170	
R30 [s8] = 0	
R31 [ra] = 4000bc	
	Kernel Text Segment [80000000]..[80010000]
	[80000180] 0001d821 addu \$27, \$0, \$1 ; 90: move \$k1 \$at # Save \$at

Int Regs [16]	Text
R2 [v0] = 4	[00400090] 34020001 ori \$2, \$0, 1 ; 44: li \$v0, 1
R3 [v1] = 0	[00400094] 0000000c syscall ; 45: syscall
R4 [a0] = 10010000	[00400098] 34020004 ori \$2, \$0, 10 ; 47: li \$v0, 10
R5 [a1] = 7ffff1a0	[0040009c] 0000000c syscall ; 48: syscall
R6 [a2] = 7ffff1a8	[004000a0] c5220000 lwc1 \$f2, 0(\$9) ; 51: lwc1 \$f2, 0(\$t1) # Load the first float from the array
R7 [a3] = 0	[004000a4] 46001032 c.eq.s \$f2, \$f0 ; 53: c.eq.s \$f2, \$f0 # Check if the float is 0.0 (end of array)
R8 [t0] = 0	[004000a8] 45010015 bclt 84 [END_FUNC-0x004000a8]; 54: bclt END_FUNC # If true, go to END_FUNC
R9 [t1] = 10010084	[004000ac] 21290004 addi \$9, \$9, 4 ; 56: addi \$t1, \$t1, 4 # Move to the next element
R10 [t2] = 0	[004000b0] 27bdffff addiu \$29, \$29, -4 ; 58: subu \$sp, \$sp, 4 # Allocate space on the stack
R11 [t3] = 0	[004000b4] afbf0000 sw \$31, 0(\$29) ; 59: sw \$ra, 0(\$sp) # Save \$ra on the stack
R12 [t4] = 0	[004000b8] 0c100028 jal 0x004000a0 [LEN] ; 61: jal LEN # Call the recursive
R13 [t5] = 0	[004000bc] 8fbff000 lw \$31, 0(\$29) ; 63: lw \$ra, 0(\$sp) # Load \$ra on the stack
R14 [t6] = 0	[004000c0] 27bd4004 addiu \$29, \$29, 4 ; 64: addu \$sp, \$sp, 4 # Move to the next element
R15 [t7] = 0	[004000c4] 22100001 addi \$16, \$16, 1 ; 66: addi \$s0, \$s0, 1 # Increase counter
R16 [s0] = 1	[004000c8] 03e00008 jr \$31 ; 68: j \$ra # Return the \$ra
R17 [s1] = 0	[004000cc] c4a20000 lwc1 \$f2, 0(\$5) ; 71: lwc1 \$f2, 0(\$a1)
R18 [s2] = 0	[004000d0] 46001032 c.eq.s \$f2, \$f0 ; 73: c.eq.s \$f2, \$f0 # Check if the float is 0.0 (end of array)
R19 [s3] = 0	[004000d4] 45010004 bclt 40 [END_FUNC-0x004000d4]; 74: bclt END_FUNC # If true, go to END_FUNC
R20 [s4] = 0	[004000d8] 46001306 mov.s \$12, \$f2 ; 76: mov.s \$f12, \$f2
R21 [s5] = 0	[004000dc] 34020002 ori \$2, \$0, 2 ; 77: li \$v0, 2
R22 [s6] = 0	[004000e0] 0000000c syscall ; 82: syscall
R23 [s7] = 0	[004000e4] 3c011001 lui \$1, 4097 [strSpace] ; 80: la \$a0, strSpace
R24 [t8] = 0	[004000e8] 34240052 ori \$4, \$1, 82 [strSpace] ; 81: li \$v0, 4
R25 [t9] = 0	[004000f0] 0000000c syscall ; 82: syscall
R26 [k0] = 0	[004000f4] 20a50004 addi \$5, \$5, 4 ; 84: addi \$a1, \$a1, 4
R27 [k1] = 0	[004000f8] 08100033 j 0x004000cc [DISPLAY] ; 85: j DISPLAY
R28 [gp] = 10008000	[004000fc] 03e00008 jr \$31 ; 88: j \$ra
R29 [sp] = 7ffff170	
R30 [s8] = 0	
R31 [ra] = 4000bc	
	Kernel Text Segment [80000000]..[80010000]
	[80000180] 0001d821 addu \$27, \$0, \$1 ; 90: move \$k1 \$at # Save \$at

Int Regs [16]	Text
R2 [v0] = 4	[00400090] 34020001 ori \$2, \$0, 1 ; 44: li \$v0, 1
R3 [v1] = 0	[00400094] 0000000c syscall ; 45: syscall
R4 [a0] = 10010000	[00400098] 34020004 ori \$2, \$0, 10 ; 47: li \$v0, 10
R5 [a1] = 7ffff1a0	[0040009c] 0000000c syscall ; 48: syscall
R6 [a2] = 7ffff1a8	[004000a0] c5220000 lwc1 \$f2, 0(\$9) ; 51: lwc1 \$f2, 0(\$t1) # Load the first float from the array
R7 [a3] = 0	[004000a4] 46001032 c.eq.s \$f2, \$f0 ; 53: c.eq.s \$f2, \$f0 # Check if the float is 0.0 (end of array)
R8 [t0] = 0	[004000a8] 45010015 bclt 84 [END_FUNC-0x004000a8]; 54: bclt END_FUNC # If true, go to END_FUNC
R9 [t1] = 10010084	[004000ac] 21290004 addi \$9, \$9, 4 ; 56: addi \$t1, \$t1, 4 # Move to the next element
R10 [t2] = 0	[004000b0] 27bdffff addiu \$29, \$29, -4 ; 58: subu \$sp, \$sp, 4 # Allocate space on the stack
R11 [t3] = 0	[004000b4] afbf0000 sw \$31, 0(\$29) ; 59: sw \$ra, 0(\$sp) # Save \$ra on the stack
R12 [t4] = 0	[004000b8] 0c100028 jal 0x004000a0 [LEN] ; 61: jal LEN # Call the recursive
R13 [t5] = 0	[004000bc] 8fbff000 lw \$31, 0(\$29) ; 63: lw \$ra, 0(\$sp) # Load \$ra on the stack
R14 [t6] = 0	[004000c0] 27bd4004 addiu \$29, \$29, 4 ; 64: addu \$sp, \$sp, 4 # Move to the next element
R15 [t7] = 0	[004000c4] 22100001 addi \$16, \$16, 1 ; 66: addi \$s0, \$s0, 1 # Increase counter
R16 [s0] = 1	[004000c8] 03e00008 jr \$31 ; 68: j \$ra # Return the \$ra
R17 [s1] = 0	[004000cc] c4a20000 lwc1 \$f2, 0(\$5) ; 71: lwc1 \$f2, 0(\$a1)
R18 [s2] = 0	[004000d0] 46001032 c.eq.s \$f2, \$f0 ; 73: c.eq.s \$f2, \$f0 # Check if the float is 0.0 (end of array)
R19 [s3] = 0	[004000d4] 45010004 bclt 40 [END_FUNC-0x004000d4]; 74: bclt END_FUNC # If true, go to END_FUNC
R20 [s4] = 0	[004000d8] 46001306 mov.s \$12, \$f2 ; 76: mov.s \$f12, \$f2
R21 [s5] = 0	[004000dc] 34020002 ori \$2, \$0, 2 ; 77: li \$v0, 2
R22 [s6] = 0	[004000e0] 0000000c syscall ; 82: syscall
R23 [s7] = 0	[004000e4] 3c011001 lui \$1, 4097 [strSpace] ; 80: la \$a0, strSpace
R24 [t8] = 0	[004000e8] 34240052 ori \$4, \$1, 82 [strSpace] ; 81: li \$v0, 4
R25 [t9] = 0	[004000f0] 0000000c syscall ; 82: syscall
R26 [k0] = 0	[004000f4] 20a50004 addi \$5, \$5, 4 ; 84: addi \$a1, \$a1, 4
R27 [k1] = 0	[004000f8] 08100033 j 0x004000cc [DISPLAY] ; 85: j DISPLAY
R28 [gp] = 10008000	[004000fc] 03e00008 jr \$31 ; 88: j \$ra
R29 [sp] = 7ffff174	
R30 [s8] = 0	
R31 [ra] = 4000bc	
	Kernel Text Segment [80000000]..[80010000]
	[80000180] 0001d821 addu \$27, \$0, \$1 ; 90: move \$k1 \$at # Save \$at

Int Regs [16]

R2	[v0] = 4
R3	[v1] = 0
R4	[a0] = 10010000
R5	[a1] = 7ffff1a0
R6	[a2] = 7ffff1a8
R7	[a3] = 0
R8	[t0] = 0
R9	[t1] = 10010084
R10	[t2] = 0
R11	[t3] = 0
R12	[t4] = 0
R13	[t5] = 0
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] = 7ffff174
R30	[s8] = 0
R31	[ra] = 4000bc

Text

```

[00400090] 34020001 ori $2, $0, 1 ; 44: li $v0, 1
[00400094] 0000000c syscall ; 45: syscall
[00400098] 34020000 ori $2, $0, 10 ; 47: li $v0, 10
[0040009c] 0000000c syscall ; 48: syscall
[004000a0] c5220000 lwc1 $f2, 0($9) ; 51: lwc1 $f2, 0($t1) # Load the first float from the array
[004000a4] 46001032 c.eq.s $f2, $f0 ; 53: c.eq.s $f2, $f0 # Check if the float is 0.0 (end of array)
[004000a8] 45010015 bclt 84 [END_FUNC-0x004000a8]; 54: bclt END_FUNC # If true, go to END_FUNC
[004000ac] 21290004 addi $9, $9, 4 ; 56: addi $t1, $t1, 4 # Move to the next element
[004000b0] 27bdfffc addiu $29, $29, -4 ; 58: subu $sp, $sp, 4 # Allocate space on the stack
[004000b4] afbf0000 sw $31, 0($29) ; 59: sw $ra, 0($sp) # Save $ra on the stack
[004000b8] 0c100028 jal 0x004000a0 [LEN] ; 61: jal LEN # Call the recursive
[004000bc] 8fbff000 lw $31, 0($29) ; 63: lw $ra, 0($sp) # Load $ra on the stack
[004000c0] 27bd0004 addiu $29, $29, 4 ; 64: addu $sp, $sp, 4 # Move to the next element
[004000c4] 22100001 addi $16, $16, 1 ; 66: addi $s0, $s0, 1 # Increase counter
[004000c8] 03e00008 jr $31 ; 68: j $ra # Return the $ra
[004000cc] c4a20000 lwc1 $f2, 0($9) ; 71: lwc1 $f2, 0($a1)
[004000d0] 46001032 c.eq.s $f2, $f0 ; 73: c.eq.s $f2, $f0 # Check if the float is 0.0 (end of array)
[004000d4] 45010016 bclt 40 [END_FUNC-0x004000d4]; 74: bclt END_FUNC # If true, go to END_FUNC
[004000d8] 46001306 mov.s $f12, $f2 ; 76: mov.s $f12, $f2
[004000dc] 34020002 ori $2, $0, 2 ; 77: li $v0, 2
[004000e0] 0000000c syscall ; 78: syscall
[004000e4] 3c011001 lui $1, 4097 [strSpace] ; 80: la $a0, strSpace
[004000e8] 34240052 ori $4, $1, 82 [strSpace]
[004000ec] 34020004 ori $2, $0, 4 ; 81: li $v0, 4
[004000f0] 0000000c syscall ; 82: syscall
[004000f4] 20a50004 addi $5, $5, 4 ; 84: addi $a1, $a1, 4
[004000f8] 08100033 j 0x004000cc [DISPLAY] ; 85: j DISPLAY
[004000fc] 03e00008 jr $31 ; 88: j $ra

```

Kernel Text Segment [80000000]..[80010000]

Int Regs [16]

R2	[v0] = 4
R3	[v1] = 0
R4	[a0] = 10010000
R5	[a1] = 7ffff1a0
R6	[a2] = 7ffff1a8
R7	[a3] = 0
R8	[t0] = 0
R9	[t1] = 10010084
R10	[t2] = 0
R11	[t3] = 0
R12	[t4] = 0
R13	[t5] = 0
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] = 7ffff178
R30	[s8] = 0
R31	[ra] = 4000bc

Text

```

[00400090] 34020001 ori $2, $0, 1 ; 44: li $v0, 1
[00400094] 0000000c syscall ; 45: syscall
[00400098] 34020000 ori $2, $0, 10 ; 47: li $v0, 10
[0040009c] 0000000c syscall ; 48: syscall
[004000a0] c5220000 lwc1 $f2, 0($9) ; 51: lwc1 $f2, 0($t1) # Load the first float from the array
[004000a4] 46001032 c.eq.s $f2, $f0 ; 53: c.eq.s $f2, $f0 # Check if the float is 0.0 (end of array)
[004000a8] 45010015 bclt 84 [END_FUNC-0x004000a8]; 54: bclt END_FUNC # If true, go to END_FUNC
[004000ac] 21290004 addi $9, $9, 4 ; 56: addi $t1, $t1, 4 # Move to the next element
[004000b0] 27bdfffc addiu $29, $29, -4 ; 58: subu $sp, $sp, 4 # Allocate space on the stack
[004000b4] afbf0000 sw $31, 0($29) ; 59: sw $ra, 0($sp) # Save $ra on the stack
[004000b8] 0c100028 jal 0x004000a0 [LEN] ; 61: jal LEN # Call the recursive
[004000bc] 8fbff000 lw $31, 0($29) ; 63: lw $ra, 0($sp) # Load $ra on the stack
[004000c0] 27bd0004 addiu $29, $29, 4 ; 64: addu $sp, $sp, 4 # Move to the next element
[004000c4] 22100001 addi $16, $16, 1 ; 66: addi $s0, $s0, 1 # Increase counter
[004000c8] 03e00008 jr $31 ; 68: j $ra # Return the $ra
[004000cc] c4a20000 lwc1 $f2, 0($5) ; 71: lwc1 $f2, 0($a1)
[004000d0] 46001032 c.eq.s $f2, $f0 ; 73: c.eq.s $f2, $f0 # Check if the float is 0.0 (end of array)
[004000d4] 45010016 bclt 40 [END_FUNC-0x004000d4]; 74: bclt END_FUNC # If true, go to END_FUNC
[004000d8] 46001306 mov.s $f12, $f2 ; 76: mov.s $f12, $f2
[004000dc] 34020002 ori $2, $0, 2 ; 77: li $v0, 2
[004000e0] 0000000c syscall ; 78: syscall
[004000e4] 3c011001 lui $1, 4097 [strSpace] ; 80: la $a0, strSpace
[004000e8] 34240052 ori $4, $1, 82 [strSpace]
[004000ec] 34020004 ori $2, $0, 4 ; 81: li $v0, 4
[004000f0] 0000000c syscall ; 82: syscall
[004000f4] 20a50004 addi $5, $5, 4 ; 84: addi $a1, $a1, 4
[004000f8] 08100033 j 0x004000cc [DISPLAY] ; 85: j DISPLAY
[004000fc] 03e00008 jr $31 ; 88: j $ra

```

Kernel Text Segment [80000000]..[80010000]

FP Rgs Int Regs [16]

FP Rgs	Int Regs [16]
Int Regs [16]	
Text	
Int Regs [16]	
R2	[v0] = 4
R3	[v1] = 0
R4	[a0] = 10010000
R5	[a1] = 7ffff1a0
R6	[a2] = 7ffff1a8
R7	[a3] = 0
R8	[t0] = 0
R9	[t1] = 10010084
R10	[t2] = 0
R11	[t3] = 0
R12	[t4] = 0
R13	[t5] = 0
R14	[t6] = 0
R15	[t7] = 0
R16	[s0] = 3
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] = 7ffff178
R30	[s8] = 0
R31	[ra] = 4000bc

Data

00400090	34020001	ori \$2, \$0, 1	; 44: li \$v0, 1
00400094	0000000c	syscall	; 45: syscall
00400098	34020000	ori \$2, \$0, 10	; 47: li \$v0, 10
0040009c	0000000c	syscall	; 48: syscall
004000a0	c5220000	lwc1 \$f2, 0(\$9)	; 51: lwc1 \$f2, 0(\$t1) # Load the first float from the array
004000a4	46001032	c.eq.s \$f2, \$f0	; 53: c.eq.s \$f2, \$f0 # Check if the float is 0.0 (end of array)
004000a8	45010015	bclt 84 [END_FUNC-0x004000a8]; 54: bclt END_FUNC # If true, go to END_FUNC	
004000ac	21290004	addi \$9, \$9, 4	; 56: addi \$t1, \$t1, 4 # Move to the next element
004000b0	27bdfffc	addiu \$29, \$29, -4	; 58: subu \$sp, \$sp, 4 # Allocate space on the stack
004000b4	afbf0000	sw \$31, 0(\$29)	; 59: sw \$ra, 0(\$sp) # Save \$ra on the stack
004000b8	0c100028	jal 0x004000a0 [LEN]	; 61: jal LEN # Call the recursive
004000bc	8fbff000	lw \$31, 0(\$29)	; 63: lw \$ra, 0(\$sp) # Load \$ra on the stack
004000c0	27bd0004	addiu \$29, \$29, 4	; 64: addu \$sp, \$sp, 4 # Move to the next element
004000c4	22100001	addi \$16, \$16, 1	; 66: addi \$s0, \$s0, 1 # Increase counter
004000c8	03e00008	jr \$31	; 68: j \$ra # Return the \$ra
004000cc	c4a20000	lwc1 \$f2, 0(\$5)	; 71: lwc1 \$f2, 0(\$a1)
004000d0	46001032	c.eq.s \$f2, \$f0	; 73: c.eq.s \$f2, \$f0 # Check if the float is 0.0 (end of array)
004000d4	45010016	bclt 40 [END_FUNC-0x004000d4]; 74: bclt END_FUNC # If true, go to END_FUNC	
004000d8	46001306	mov.s \$f12, \$f2	; 76: mov.s \$f12, \$f2
004000dc	34020002	ori \$2, \$0, 2	; 77: li \$v0, 2
004000e0	0000000c	syscall	; 78: syscall
004000e4	3c011001	lui \$1, 4097 [strSpace]	; 80: la \$a0, strSpace
004000e8	34240052	ori \$4, \$1, 82 [strSpace]	
004000ec	34020004	ori \$2, \$0, 4	; 81: li \$v0, 4
004000f0	0000000c	syscall	; 82: syscall
004000f4	20a50004	addi \$5, \$5, 4	; 84: addi \$a1, \$a1, 4
004000f8	08100033	j 0x004000cc [DISPLAY]	; 85: j DISPLAY
004000fc	03e00008	jr \$31	; 88: j \$ra

Text

```

[00400090] 34020001 ori $2, $0, 1 ; 44: li $v0, 1
[00400094] 0000000c syscall ; 45: syscall
[00400098] 34020000 ori $2, $0, 10 ; 47: li $v0, 10
[0040009c] 0000000c syscall ; 48: syscall
[004000a0] c5220000 lwc1 $f2, 0($9) ; 51: lwc1 $f2, 0($t1) # Load the first float from the array
[004000a4] 46001032 c.eq.s $f2, $f0 ; 53: c.eq.s $f2, $f0 # Check if the float is 0.0 (end of array)
[004000a8] 45010015 bclt 84 [END_FUNC-0x004000a8]; 54: bclt END_FUNC # If true, go to END_FUNC
[004000ac] 21290004 addi $9, $9, 4 ; 56: addi $t1, $t1, 4 # Move to the next element
[004000b0] 27bdfffc addiu $29, $29, -4 ; 58: subu $sp, $sp, 4 # Allocate space on the stack
[004000b4] afbf0000 sw $31, 0($29) ; 59: sw $ra, 0($sp) # Save $ra on the stack
[004000b8] 0c100028 jal 0x004000a0 [LEN] ; 61: jal LEN # Call the recursive
[004000bc] 8fbff000 lw $31, 0($29) ; 63: lw $ra, 0($sp) # Load $ra on the stack
[004000c0] 27bd0004 addiu $29, $29, 4 ; 64: addu $sp, $sp, 4 # Move to the next element
[004000c4] 22100001 addi $16, $16, 1 ; 66: addi $s0, $s0, 1 # Increase counter
[004000c8] 03e00008 jr $31 ; 68: j $ra # Return the $ra
[004000cc] c4a20000 lwc1 $f2, 0($5) ; 71: lwc1 $f2, 0($a1)
[004000d0] 46001032 c.eq.s $f2, $f0 ; 73: c.eq.s $f2, $f0 # Check if the float is 0.0 (end of array)
[004000d4] 45010016 bclt 40 [END_FUNC-0x004000d4]; 74: bclt END_FUNC # If true, go to END_FUNC
[004000d8] 46001306 mov.s $f12, $f2 ; 76: mov.s $f12, $f2
[004000dc] 34020002 ori $2, $0, 2 ; 77: li $v0, 2
[004000e0] 0000000c syscall ; 78: syscall
[004000e4] 3c011001 lui $1, 4097 [strSpace] ; 80: la $a0, strSpace
[004000e8] 34240052 ori $4, $1, 82 [strSpace]
[004000ec] 34020004 ori $2, $0, 4 ; 81: li $v0, 4
[004000f0] 0000000c syscall ; 82: syscall
[004000f4] 20a50004 addi $5, $5, 4 ; 84: addi $a1, $a1, 4
[004000f8] 08100033 j 0x004000cc [DISPLAY] ; 85: j DISPLAY
[004000fc] 03e00008 jr $31 ; 88: j $ra

```

Kernel Text Segment [80000000]..[80010000]

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

## Exercise 2:

### Code (Bubble\_Sort):

```
● ● ●
1 .data
2 strNameId: .asciiz "Phan Tran Thanh Huy\nnITCSIU22086\n"
3 printInitialArray: .asciiz "The initial array: "
4 printSortedArray: .asciiz "The sorted array: "
5 strLine: .asciiz "\n"
6 strSpace: .asciiz " "
7 arr: .word 43, 6543, 34, 54, 4232, 64, 526, 643, 6435, 423, 4236, 566, 56, 0
8 sortedArr: .space 1024
9
10 .text
11 .global main
12
13 main:
14     la $a0, strNameId
15     li $v0, 4
16     syscall
17
18     la $a1, arr      # a1 = address of arr
19     li $s0, 0          # s0 = 0
20     jal LEN          # Compute the length of the array
21
22     la $a0, printInitialArray
23     li $v0, 4
24     syscall
25
26     la $a1, arr      # a1 = address of arr
27     jal DISPLAY      # Display the array
28
29     la $a0, strLine
30     li $v0, 4
31     syscall
32
33     la $a1, arr      # a1 = address of arr
34     li $t0, 0          # t0 = 0
35     jal SORT          # Sort the array
36
37     la $a0, printSortedArray
38     li $v0, 4
39     syscall
40
41     la $a1, arr      # a1 = address of arr
42     jal DISPLAY      # Display the array
43
44     la $a0, strLine
45     li $v0, 4
46     syscall
47
48     li $v0, 10
49     syscall
50
51 LEN:
52     lw $t2, 0($a1)      # Load t2 = a[i]
53     beq $t2, $0, END_FUNC  # Check if t2 = a[i] = 0, stop the LEN
54
55     addi $a1, $a1, 4      # Move the next element
56     addi $s0, $s0, 1      # Increase counter
57     j LEN
58
59 SORT:
60     j OUTER_LOOP        # Jump to OUTER_LOOP
61
62 OUTER_LOOP:
63     bge $t0, $s0, END_FUNC  # Check if t0 = s0 = length of the array, stop the OUTER_LOOP
64     li $t0, 0              # t0 = 0
65     li $t1, 0              # t1 = 0
66     j INNER_LOOP          # Jump to INNER_LOOP
67
68 INNER_LOOP:
69     sub $t2, $s0, $t0      # t2 = s0 - t0
70     addi $t2, $t2, -1      # t2 = t2 - 1 = the current index
71
72     bge $t1, $t2, CHECK_LOOP  # Check if t1 >= t2, go to CHECK_LOOP
73     li $t3, 4              # t3 = 4
74     mul $t3, $t3, $t1      # t3 = t3*t1
75     add $t3, $a1, $t3      # t3 = a1 + t3 = a[i]
76
77     addi $t1, $t1, 1      # Increase the counter of INNER_LOOP
78
79     lw $t4, 0($t3)
80     lw $t5, 4($t3)
81
82     bne $t4, $t5, INNER_LOOP  # if t4 <= t5, skip the swap
83     # Swap Variable
84     sw $t5, 0($t3)
85     sw $t4, 4($t3)
86     li $t9, 1              # t9 = 1
87     j INNER_LOOP
88
89 CHECK_LOOP:
90     beq $t9, $0, END_FUNC  # Check if t9 = 0, stop the sort
91     addi $t0, $t0, 1        # Increase the counter of OUTER_LOOP
92     j OUTER_LOOP
93
94 DISPLAY:
95     lw $t2, 0($a1)      # Load t2 = a[i]
96     beq $t2, $0, END_FUNC  # Check if t2 = 0, go to END_FUNC
97
98     move $a0, $t2          # Print the a[i]
99     li $v0, 1
100    syscall
101
102    la $a0, strSpace
103    li $v0, 4
104    syscall
105
106    addi $a1, $a1, 4      # Move to the next element
107    j DISPLAY
108
109 END_FUNC:
110    j $ra
```

## Test case:

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

## Single Step:

## OUTER\_LOOP:

```
Int Regs [16] Text
PC = 4000cc [00400005c] 3c011001 lui $1, 4097 [strLine] ; 29: la $a0, strLine
EPC = 0 [004000060] 34240048 ori $4, $1, 72 [strLine] ; 30: li $v0, 4
Cause = 0 [004000064] 34020004 ori $2, $0, 4 ; 31: syscall
BadVAddr = 0 [004000068] 00000000 syscall ; 31: syscall
Status = 3000ff10 [00400006c] 3c011001 lui $1, 4097 [arr] ; 33: la $a1, arr # $1 = address of arr
[004000070] 3425004c ori $5, $1, 76 [arr]
HI = 0 [004000074] 34080000 ori $8, $0, 0 ; 34: li $t0, 0 # t0 = 0
LO = 0 [004000078] 0c100031 jal 0x00400004 [SORT] ; 35: jal SORT # Sort the array
[00400007c] 34250040 ori $3, $0, 1
[004000080] 34240035 jal 0x00400004 [SORT] ; 37: jal SORT # Sort the array
[004000084] 34020004 ori $4, $1, 53 [printSortedArray]
[004000088] 34020004 ori $2, $0, 4 ; 38: li $v0, 4
[004000092] 0000000c syscall ; 39: syscall
[004000096] 3c011001 lui $1, 4097 [arr] ; 41: la $a1, arr # $1 = address of arr
R0 [r0] = 0 [00400009a] 3425004c ori $5, $1, 76 [arr]
R1 [a1] = 1 [00400009c] 34020004 ori $1, $0, 1
R2 [v0] = 4 [00400009e] 34020004 ori $2, $0, 4 ; 42: jal DISPLAY # Display the array
R3 [v1] = 0 [0040000a0] 34020004 ori $2, $0, 4 ; 43: jal DISPLAY # Display the array
R4 [a0] = 10010048 [0040000a4] 3425004c ori $5, $1, 76 [arr]
R5 [a1] = 1001004c [0040000a8] 34020004 ori $1, $0, 1
R6 [a2] = 7fffffa8 [0040000a9] 34020004 ori $2, $0, 4 ; 44: la $a0, strLine
R7 [t0] = 0 [0040000a9c] 34240040 ori $4, $1, 72 [strLine]
R8 [t1] = 0 [0040000a9e] 34020004 ori $2, $0, 4 ; 45: li $t0, 4
R9 [t1] = 0 [0040000a9f] 0000000c syscall ; 46: syscall
R10 [t2] = 0 [0040000b0] 34020004 ori $2, $0, 10 ; 48: li $v0, 10
R11 [t3] = 0 [0040000b4] 0000000c syscall ; 49: syscall
R12 [t4] = 0 [0040000b8] 8caa0000 lw $10, 0($5) ; 52: lw $t2, 0($a1) # Load t2 = a[i]
R13 [t5] = 0 [0040000b84] 11400028 beg $10, $0, 160 [END_FUNC-0x00400004]
R14 [t6] = 0 [0040000b9] 20a50004 addi $5, $5, 4 ; 55: addi $a1, $a1, 4 # Move the next element
R15 [t7] = 0 [0040000bc] 22100001 addi $16, $16, 1 ; 56: addi $s0, $s0, 1 # Increase counter
R16 [s0] = d [0040000c0] 0810002c j $0x040000b ; 57: j LEN
R17 [s1] = 0 [0040000c4] 0810002c j $0x04000c8 [OUTER_LOOP] ; 60: j OUTER_LOOP # Jump to OUTER_LOOP
R18 [s2] = 0 [0040000c8] 01100082 slt $1, $8, $16 ; 63: bge $t0, $s0, END_FUNC # Check if t0 = s0 = length of the array, stop the OUTER_LOOP
R19 [s3] = 0 [0040000cc] 10200022 sll $1, $0, 136 [END_FUNC-0x0040000cc]
R20 [s4] = 0 [0040000d0] 34190000 ori $25, $0, 0 ; 64: li $t9, 0 # t9 = 0
R21 [s5] = 0 [0040000d4] 34090000 ori $9, $0, 0 ; 65: li $t1, 0 # t1 = 0
```

## INNER\_LOOP:

In the first loop inside INNER\_LOOP:

Int Regs [16]	Text
PC = 4000e0	[00400068] 00000000 syscall ; 31: syscall
EPC = 0	[0040006c] 3c011001 lui \$1, 4097 [arr] ; 33: la \$a1, arr # a1 = address of arr
Cause = 0	[00400070] 3425004c ori \$5, \$1, 76 [arr]
BadAddr = 0	[00400074] 34080000 ori \$8, \$0, 0 ; 34: li \$t0, 0 # t0 = 0
Status = 3000ff10	[00400078] 0c100031 jal 0x040000c4 [SORT] ; 35: jal SORT # Sort the array
HI = 0	[0040007c] 3c011001 lui \$1, 4097 [printSortedArray] ; 37: la \$a0, printSortedArray
LO = 0	[00400080] 34240035 ori \$4, \$1, 53 [printSortedArray]
R0 [r0] = 0	[00400084] 34020004 ori \$2, \$0, 4 ; 38: li \$v0, 4
R1 [at1] = 1	[00400088] 00000000 syscall ; 39: syscall
R2 [v0] = 4	[00400090] 3425004c ori \$5, \$1, 76 [arr] ; 41: la \$a1, arr # a1 = address of arr
R3 [v1] = 0	[00400094] 0c10004a jal 0x04000028 [DISPLAY] ; 42: jal DISPLAY # Display the array
R4 [a01] = 10010048	[00400098] 3c011001 lui \$1, 4097 [strLine] ; 44: la \$a0, strLine
R5 [a1] = 1001004c	[0040009c] 34240048 ori \$4, \$1, 72 [strLine]
R6 [a2] = 7fffffa8	[004000a0] 34020004 ori \$2, \$0, 4 ; 45: li \$v0, 4
R7 [a3] = 0	[004000a8] 34020004a ori \$2, \$0, 10 ; 48: li \$v0, 10
R8 [t0] = 0	[004000ac] 00000000 syscall ; 49: syscall
R9 [t1] = 0	[004000b0] 8ca00000 lw \$10, 0(\$5) ; 52: lw \$t2, 0(\$a1) # Load t2 = a[i]
R10 [t2] = d	[004000b4] 11400028 beq \$10, \$0, 160 [END_FUNC-0x040000d4]
R11 [t3] = 0	[004000b8] 20a50004 add \$5, \$5, 4 ; 55: addi \$a1, \$a1, 4 # Move the next element
R12 [t4] = 0	[004000bc] 22100001 add \$16, \$16, 1 ; 56: addi \$s0, \$s0, 1 # Increase counter
R13 [t5] = 0	[004000cc] 0810002c j 0x040000b0 [LEN] ; 57: j LEN
R14 [t6] = 0	[004000c4] 08100032 j 0x040000c8 [OUTER_LOOP] ; 60: j OUTER_LOOP # Jump to OUTER_LOOP
R15 [t7] = 0	[004000c8] 01100082 slt \$1, \$s0, \$16 ; 63: bge \$t0, \$s0, END_FUNC # Check if t0 = s0 = length of the array, stop the OUTER_LOOP
R16 [s01] = d	[004000cc] 10200002 beq \$1, \$0, 136 [END_FUNC-0x040000cc]
R17 [s1] = 0	[004000d0] 34190000 ori \$25, \$0, 0 ; 64: li \$t9, 0 # t9 = 0
R18 [s2] = 0	[004000d4] 34090000 ori \$9, \$0, 0 ; 65: li \$t1, 0 # t1 = 0
R19 [s3] = 0	[004000d8] 08100037 j 0x040000dc [INNER_LOOP] ; 66: j INNER_LOOP # Jump to INNER_LOOP
R20 [s4] = 0	[004000dc] 02085022 sub \$10, \$16, \$8 ; 69: sub \$t2, \$s0, \$t0 # t2 = s0 - t0
R21 [s5] = 0	[004000e0] 214fffff addi \$10, \$10, -1 ; 70: addi \$t2, \$t2, -1 # t2 = t2 - 1 = the current index
...	[004000e4] 012a002a slt \$1, \$9, \$10 ; 72: bge \$t1, \$t3, CHECK_LOOP # Check if t1 >= t3, go to CHECK_LOOP

Int Regs [16]	Text
PC = 4000e4	[0040006c] 3c011001 lui \$1, 4097 [arr] ; 33: la \$a1, arr # a1 = address of arr
EPC = 0	[00400070] 3425004c ori \$5, \$1, 76 [arr]
Cause = 0	[00400074] 34080000 ori \$8, \$0, 0 ; 34: li \$t0, 0 # t0 = 0
BadAddr = 0	[00400078] 0c100031 jal 0x040000c4 [SORT] ; 35: jal SORT # Sort the array
Status = 3000ff10	[0040007c] 3c011001 lui \$1, 4097 [printSortedArray] ; 37: la \$a0, printSortedArray
HI = 0	[00400080] 34240035 ori \$4, \$1, 53 [printSortedArray]
LO = 0	[00400084] 34020004 ori \$2, \$0, 4 ; 38: li \$v0, 4
R0 [r0] = 0	[00400088] 00000000 syscall ; 39: syscall
R1 [at1] = 1	[00400090] 3425004c ori \$5, \$1, 76 [arr] ; 41: la \$a1, arr # a1 = address of arr
R2 [v0] = 4	[00400094] 0c10004a jal 0x04000028 [DISPLAY] ; 42: jal DISPLAY # Display the array
R3 [v1] = 0	[00400098] 3c011001 lui \$1, 4097 [strLine] ; 44: la \$a0, strLine
R4 [a01] = 10010048	[0040009c] 34240048 ori \$4, \$1, 72 [strLine]
R5 [a1] = 1001004c	[004000a0] 34020004 ori \$2, \$0, 4 ; 45: li \$v0, 4
R6 [a2] = 7fffffa8	[004000a8] 34020004a ori \$2, \$0, 10 ; 48: li \$v0, 10
R7 [a3] = 0	[004000ac] 00000000 syscall ; 49: syscall
R8 [t0] = 0	[004000b0] 8ca00000 lw \$10, 0(\$5) ; 52: lw \$t2, 0(\$a1) # Load t2 = a[i]
R9 [t1] = 0	[004000b4] 11400028 beq \$10, \$0, 160 [END_FUNC-0x040000d4]
R10 [t2] = d	[004000b8] 20a50004 add \$5, \$5, 4 ; 55: addi \$a1, \$a1, 4 # Move the next element
R11 [t3] = 0	[004000bc] 22100001 add \$16, \$16, 1 ; 56: addi \$s0, \$s0, 1 # Increase counter
R12 [t4] = 0	[004000cc] 0810002c j 0x040000b0 [LEN] ; 57: j LEN
R13 [t5] = 0	[004000c4] 08100032 j 0x040000c8 [OUTER_LOOP] ; 60: j OUTER_LOOP # Jump to OUTER_LOOP
R14 [t6] = 0	[004000c8] 01100082 slt \$1, \$s0, \$16 ; 63: bge \$t0, \$s0, END_FUNC # Check if t0 = s0 = length of the array, stop the OUTER_LOOP
R15 [t7] = 0	[004000cc] 10200002 beq \$1, \$0, 136 [END_FUNC-0x040000cc]
R16 [s01] = d	[004000d0] 34190000 ori \$25, \$0, 0 ; 64: li \$t9, 0 # t9 = 0
R17 [s1] = 0	[004000d4] 34090000 ori \$9, \$0, 0 ; 65: li \$t1, 0 # t1 = 0
R18 [s2] = 0	[004000d8] 08100037 j 0x040000dc [INNER_LOOP] ; 66: j INNER_LOOP # Jump to INNER_LOOP
R19 [s3] = 0	[004000dc] 02085022 sub \$10, \$16, \$8 ; 69: sub \$t2, \$s0, \$t0 # t2 = s0 - t0
R20 [s4] = 0	[004000e0] 214fffff addi \$10, \$10, -1 ; 70: addi \$t2, \$t2, -1 # t2 = t2 - 1 = the current index
R21 [s5] = 0	[004000e4] 012a002a slt \$1, \$9, \$10 ; 72: bge \$t1, \$t3, CHECK_LOOP # Check if t1 >= t2, go to CHECK_LOOP

Int Regs [16]	Text
BadAddr = 0	[0040009c] 34240048 ori \$4, \$1, 72 [strLine]
Status = 3000ff10	[004000a0] 34020004 ori \$2, \$0, 4 ; 45: li \$v0, 4
HI = 0	[004000a4] 00000000 syscall ; 46: syscall
LO = 0	[004000a8] 0c10004c ori \$9, \$0, 10 ; 48: li \$v0, 10
R0 [t0] = 0	[004000b2] 00000000 syscall ; 49: syscall
R1 [at1] = 1	[004000b6] 3425004c ori \$5, \$1, 76 [arr] ; 55: addi \$a1, \$a1, 4 # Move the next element
R2 [v0] = 4	[004000bc] 22100001 add \$16, \$16, 1 ; 56: addi \$s0, \$s0, 1 # Increase counter
R3 [v1] = 0	[004000cc] 0810002c j 0x040000b0 [LEN] ; 57: j LEN
R4 [a01] = 10010048	[004000c4] 08100032 j 0x040000c8 [OUTER_LOOP] ; 60: j OUTER_LOOP # Jump to OUTER_LOOP
R5 [a1] = 1001004c	[004000c8] 01100082 slt \$1, \$s0, \$16 ; 63: bge \$t0, \$s0, END_FUNC # Check if t0 = s0 = length of the array, stop the OUTER_LOOP
R6 [a2] = 7fffffa8	[004000cc] 10200002 beq \$1, \$0, 136 [END_FUNC-0x040000cc]
R7 [t1] = 0	[004000d0] 34190000 ori \$25, \$0, 0 ; 64: li \$t9, 0 # t9 = 0
R8 [t0] = 0	[004000d4] 34090000 ori \$9, \$0, 0 ; 65: li \$t1, 0 # t1 = 0
R9 [t1] = 0	[004000d8] 08100037 j 0x040000dc [INNER_LOOP] ; 66: j INNER_LOOP # Jump to INNER_LOOP
R10 [t2] = c	[004000dc] 02085022 sub \$10, \$16, \$8 ; 69: sub \$t2, \$s0, \$t0 # t2 = s0 - t0
R11 [t3] = 4	[004000e0] 012a002a slt \$1, \$9, \$10 ; 72: bge \$t1, \$t3, CHECK_LOOP # Check if t1 >= t2, go to CHECK_LOOP
R12 [t4] = 0	[004000e4] 34090000 ori \$9, \$0, 0 ; 73: addi \$t1, \$t1, 1 # t1 = t1 + 1
R13 [t5] = 0	[004000e8] 10200004 add \$1, \$1, \$12 ; 74: addi \$t3, \$t3, \$t3 # t3 = a[i]
R14 [t6] = 0	[004000f2] 00000004 syscall ; 75: addi \$t3, \$t3, \$t3 # t3 = a[i] + t3 = a[i + t3]
R15 [t7] = 0	[004000f6] 00000004 syscall ; 76: addi \$t3, \$t3, \$t3 # t3 = a[i + t3]
R16 [s01] = d	[004000f8] 00ab5820 add \$11, \$5, \$11 ; 77: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R17 [s1] = 0	[004000f8] 21200001 add \$9, \$9, \$1 ; 78: addi \$t4, \$t4, 0(\$t3)
R18 [s2] = 0	[004000f8] 00000004 syscall ; 79: addi \$t4, \$t4, 0(\$t3)
R19 [s3] = 0	[00400100] 8d6d0004 lw \$12, 0(\$11) ; 80: addi \$t4, \$t4, 4(\$t3)
R20 [s4] = 0	[00400104] 01ac082a sll \$1, \$13, \$12 ; 82: addi \$t4, \$t4, 4(\$t3)
R21 [s5] = 0	[00400108] 1020ffff5 beq \$1, \$0, -44 [INNER_LOOP-0x04000108]
R22 [s6] = 0	[0040010c] aded0004 sw \$13, 0(\$11) ; 84: sw \$t5, 0(\$t3)
R23 [s7] = 0	[00400110] aded0004 sw \$12, 4(\$11) ; 85: sw \$t4, 4(\$t3)
R24 [t8] = 0	[00400114] 34190001 ori \$25, \$0, 1 ; 86: li \$t9, 1 # t9 = 1
...	[00400118] 00000004 lw \$12, 0(\$11) ; 87: addi \$t1, \$t1, 1 # t1 = t1 + 1

Int Regs [16]	Text
BadAddr = 0	l0040009c 34240048 ori \$4, \$1, 72 [strLine]
Status = 3000ff10	[004000a0] 34020004 ori \$2, \$0, 4 ; 45: li \$v0, 4
HI = 0	[004000a4] 0000000c syscall ; 46: syscall
LO = 0	[004000a8] 34020004 addi \$1, \$0, 10 ; 47: addi \$t0, \$v0, 10
R0 [r0] = 0	[004000ac] 0000000c syscall ; 48: syscall
R1 [at] = 1	[004000a8] 20a50004 addi \$5, \$5, 4 ; 55: addi \$a1, \$a1, 4 # Move the next element
R2 [v0] = 4	[004000a0] 22100001 addi \$16, \$16, 1 ; 56: addi \$s0, \$s0, 1 # Increase counter
R3 [vr1] = 0	[004000c0] 0810002c j 0x040000b0 [LEN] ; 57: j LEN
R4 [a0] = 10010048	[004000a4] 11400028 beq \$10, \$0, 160 [END_FUNC-0x004000b4]
R5 [a1] = 1001004c	[004000a8] 20a50004 addi \$5, \$5, 4 ; 55: addi \$a1, \$a1, 4 # Move the next element
R6 [t0] = 7fffffa8	[004000a0] 22100001 addi \$16, \$16, 1 ; 56: addi \$s0, \$s0, 1 # Increase counter
R7 [a3] = 0	[004000c0] 34190000 ori \$25, \$0, 0 ; 64: li \$t9, 0 # t9 = 0
R8 [t0] = 0	[004000a4] 34090000 ori \$9, \$0, 0 ; 65: li \$t1, 0 # t1 = 0
R9 [t1] = 0	[004000a8] 08100037 j 0x040000b0 [INNER_LOOP] ; 66: j INNER_LOOP # Jump to INNER_LOOP
R10 [t2] = c	[004000a0] 02085022 sub \$10, \$16, \$8 ; 69: sub \$t2, \$s0, \$t0 # t2 = s0 - t0
R11 [t3] = 0	[004000a4] 214affff addi \$10, \$10, -1 ; 70: addi \$t3, \$t2, -1 # t3 = t2 - 1 = the current index
R12 [t4] = 0	[004000a8] 012a002a sub \$1, \$9, \$10 ; 73: li \$t4, 0 # t4 = 0
R13 [t5] = 0	[004000a0] 10200004 addi \$11, \$11, 4 ; 74: addi \$t5, \$t3, \$t1 # t5 = t3 + t1
R14 [t6] = 0	[004000a4] 00ab5820 addi \$11, \$9, \$11 ; 75: addi \$t6, \$t5, \$t2, \$t3 # t6 = a1 + t3 = a[i]
R15 [t7] = 0	[004000a8] 21290001 addi \$1, \$9, \$12 ; 77: addi \$t7, \$t6, \$t1, 1 # Increase the counter of INNER_LOOP
R16 [s2] = 0	[004000a0] 8d6c0000 lsr \$12, \$0, 511 ; 79: lr \$t4, 0(\$t3)
R17 [s3] = 0	[004000a4] 8d6d0000 lsr \$12, 4(\$t1) ; 85: sw \$t4, 4(\$t3)
R18 [s4] = 0	[004000a8] 34190001 ori \$25, \$0, 1 ; 86: li \$t9, 1 # t9 = 1
R19 [s5] = 0	[004000a0] 08100037 4 0x004000b4: ITNNRR LOOP1: R7: 4 ITNNRR LOOP1
R20 [s6] = 0	
R21 [s7] = 0	
R22 [s8] = 0	
R23 [s9] = 0	
R24 [t8] = 0	
acc r6=	

FP Regs	Int Regs [16]	Data	Text
Int Regs [16]	Text		
BadAddr = 0	l0040009c 34240048 ori \$4, \$1, 72 [strLine]		
Status = 3000ff10	[004000a0] 34020004 ori \$2, \$0, 4 ; 45: li \$v0, 4		
HI = 0	[004000a4] 0000000c syscall ; 46: syscall		
LO = 0	[004000a8] 34020004 addi \$1, \$0, 10 ; 48: li \$v0, 10		
R0 [r0] = 0	[004000ac] 0000000c syscall ; 49: syscall		
R1 [at] = 1	[004000a0] 0810002c j 0x040000b0 [LEN] ; 52: li \$t2, 0(\$a1) # Load t2 = a[i]		
R2 [v0] = 4	[004000a4] 11400028 beq \$10, \$0, 160 [END_FUNC-0x004000b4]		
R3 [vr1] = 0	[004000a8] 20a50004 addi \$5, \$5, 4 ; 55: addi \$a1, \$a1, 4 # Move the next element		
R4 [a0] = 10010048	[004000a0] 22100001 addi \$16, \$16, 1 ; 56: addi \$s0, \$s0, 1 # Increase counter		
R5 [a1] = 1001004c	[004000a4] 34090000 ori \$25, \$0, 0 ; 64: li \$t9, 0 # t9 = 0		
R6 [t0] = 7fffffa8	[004000a8] 34090000 ori \$9, \$0, 0 ; 65: li \$t1, 0 # t1 = 0		
R7 [t1] = 0	[004000a0] 02085022 addi \$10, \$16, \$8 ; 69: sub \$t2, \$s0, \$t0 # t2 = s0 - t0		
R8 [t2] = c	[004000a4] 214affff addi \$10, \$10, -1 ; 70: addi \$t3, \$t2, -1 # t3 = t2 - 1 = the current index		
R9 [t3] = 0	[004000a8] 012a002a sub \$1, \$9, \$10 ; 73: li \$t4, 0 # t4 = 0		
R10 [t4] = 0	[004000a0] 10200004 addi \$11, \$11, 4 ; 74: addi \$t5, \$t3, \$t1 # t5 = t3 + t1		
R11 [t5] = 0	[004000a4] 00ab5820 addi \$11, \$9, \$11 ; 75: addi \$t6, \$t5, \$t2, \$t3 # t6 = a1 + t3 = a[i]		
R12 [t6] = 0	[004000a8] 21290001 addi \$1, \$9, \$12 ; 77: addi \$t7, \$t6, \$t1, 1 # Increase the counter of INNER_LOOP		
R13 [t7] = 0	[004000a0] 8d6c0000 lsr \$12, \$0, 511 ; 79: lr \$t4, 0(\$t3)		
R14 [t8] = 0	[004000a4] 8d6d0000 lsr \$12, 4(\$t1) ; 85: sw \$t4, 4(\$t3)		
R15 [t9] = 0	[004000a8] 34190001 ori \$25, \$0, 1 ; 86: li \$t9, 1 # t9 = 1		
R16 [s0] = d	[004000a0] 08100037 4 0x004000b4: ITNNRR LOOP1: R7: 4 ITNNRR LOOP1		
R17 [s1] = 0			
R18 [s2] = 0			
R19 [s3] = 0			
R20 [s4] = 0			
R21 [s5] = 0			
R22 [s6] = 0			
R23 [s7] = 0			
R24 [t8] = 0			
acc r6=			

FP Regs	Int Regs [16]	Data	Text
Int Regs [16]	Text		
BadAddr = 0	l0040009c 34240048 ori \$4, \$1, 72 [strLine]		
Status = 3000ff10	[004000a0] 34020004 ori \$2, \$0, 4 ; 45: li \$v0, 4		
HI = 0	[004000a4] 0000000c syscall ; 46: syscall		
LO = 0	[004000a8] 34020004 addi \$1, \$0, 10 ; 48: li \$v0, 10		
R0 [r0] = 0	[004000ac] 0000000c syscall ; 49: syscall		
R1 [at] = 1	[004000a0] 0810002c j 0x040000b0 [LEN] ; 52: li \$t2, 0(\$a1) # Load t2 = a[i]		
R2 [v0] = 4	[004000a4] 11400028 beq \$10, \$0, 160 [END_FUNC-0x004000b4]		
R3 [vr1] = 0	[004000a8] 20a50004 addi \$5, \$5, 4 ; 55: addi \$a1, \$a1, 4 # Move the next element		
R4 [a0] = 10010048	[004000a0] 22100001 addi \$16, \$16, 1 ; 56: addi \$s0, \$s0, 1 # Increase counter		
R5 [a1] = 1001004c	[004000a4] 34090000 ori \$25, \$0, 0 ; 64: li \$t9, 0 # t9 = 0		
R6 [t0] = 7fffffa8	[004000a8] 34090000 ori \$9, \$0, 0 ; 65: li \$t1, 0 # t1 = 0		
R7 [t1] = 0	[004000a0] 02085022 addi \$10, \$16, \$8 ; 69: sub \$t2, \$s0, \$t0 # t2 = s0 - t0		
R8 [t2] = c	[004000a4] 214affff addi \$10, \$10, -1 ; 70: addi \$t3, \$t2, -1 # t3 = t2 - 1 = the current index		
R9 [t3] = 0	[004000a8] 012a002a sub \$1, \$9, \$10 ; 73: li \$t4, 0 # t4 = 0		
R10 [t4] = 0	[004000a0] 10200004 addi \$11, \$11, 4 ; 74: addi \$t5, \$t3, \$t1 # t5 = t3 + t1		
R11 [t5] = 0	[004000a4] 00ab5820 addi \$11, \$9, \$11 ; 75: addi \$t6, \$t5, \$t2, \$t3 # t6 = a1 + t3 = a[i]		
R12 [t6] = 0	[004000a8] 21290001 addi \$1, \$9, \$12 ; 77: addi \$t7, \$t6, \$t1, 1 # Increase the counter of INNER_LOOP		
R13 [t7] = 0	[004000a0] 8d6c0000 lsr \$12, \$0, 511 ; 79: lr \$t4, 0(\$t3)		
R14 [t8] = 0	[004000a4] 8d6d0000 lsr \$12, 4(\$t1) ; 85: sw \$t4, 4(\$t3)		
R15 [t9] = 0	[004000a8] 34190001 ori \$25, \$0, 1 ; 86: li \$t9, 1 # t9 = 1		
R16 [s0] = d	[004000a0] 08100037 4 0x004000b4: ITNNRR LOOP1: R7: 4 ITNNRR LOOP1		
R17 [s1] = 0			
R18 [s2] = 0			
R19 [s3] = 0			
R20 [s4] = 0			
R21 [s5] = 0			
R22 [s6] = 0			
R23 [s7] = 0			
R24 [t8] = 0			
acc r6=			

FP Regs Int Regs [16] Data Text

BadAddr = 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] = 7fffffa8 R7 [a3] = 0 R8 [t0] = 0 R9 [t1] = 1 R10 [t2] = c R11 [t3] = 1001004c R12 [t4] = 2b 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 nne r.o. = ^

0x0400000c 34240048 ori \$4, \$1, 1/2 [strLine]

[0x040000a0] 34020004 ori \$2, \$0, 4 ; 45: li \$v0, 4

[0x040000a4] 00000000 syscall ; 46: syscall

[0x040000a8] 34020004 ori \$2, \$0, 10 ; 48: li \$v0, 10

[0x040000ac] 00000000 syscall ; 49: syscall

[0x040000b0] 80ea0000 beq \$10, \$0, (\$5) ; 52: lr \$t2, 0(\$a1) # Load t2 = a[i]

[0x040000b4] 14000028 beq \$10, \$0, 160 [END\_FUNC-0x00400000]

[0x040000b8] 20a50004 addi \$5, \$0, 160 [END\_FUNC-0x00400000]

[0x040000c0] 22100001 addi \$16, \$16, 1 ; 55: addi \$s0, \$s0, 1 # Increase counter

[0x040000c4] 08100000 syscall ; 56: syscall

[0x040000c8] 00000000 beq \$10, \$0, (\$5) ; 57: j LEN

[0x040000cc] 08100032 j 0x0400000c [OUTER\_LOOP]; 60: j OUTER\_LOOP # Jump to OUTER\_LOOP

[0x040000d0] 01100024 beq \$1, \$0, 0 [END\_FUNC-0x00400000]

[0x040000d4] 34090001 ori \$25, \$0, 0 ; 63: bge \$t0, \$s0, END\_FUNC # Check if t0 = s0 = length of the array, stop the OUTER\_LOOP

[0x040000d8] 08100037 0r1 0x040000dc [INNER\_LOOP]; 66: j INNER\_LOOP # Jump to INNER\_LOOP

[0x040000e0] 10200000 beq \$1, \$0, 52 [CHECK\_LOOP-0x00400008]

[0x040000e4] 012a002a sit \$1, \$9, \$10 ; 68: li \$t3, 4 # t3 = 4

[0x040000e8] 340b0004 ori \$1, \$0, 4 ; 73: mul \$t3, \$t3, \$t3 # t3 = t3\*t1

[0x040000f0] 71695802 mul \$11, \$11, \$9 ; 74: add \$t3, \$s1, \$t3 # t3 = a1 + t3 = a[i]

[0x040000f4] 00ab5820 add \$11, \$5, \$11 ; 75: addi \$t3, \$t1, \$t1 # Increase the counter of INNER\_LOOP

[0x040000f8] 21290001 addi \$9, \$9, 1 ; 77: addi \$t1, \$t1, 1 # Increase the counter of INNER\_LOOP

[0x040000fc] 846c0000 lw \$12, 0(\$11) ; 79: ble \$t4, \$t5, 4(\$t3)

[0x04000100] 01ac002a alt \$1, \$13, \$12 ; 82: ble \$t4, \$t5, INNER\_LOOP # if t4

[0x04000104] 10200ff5 beq \$1, \$0, -44 [INNER\_LOOP-0x00400108]

[0x0400010c] ad6d0000 sw \$13, 0(\$11) ; 84: sw \$t5, 0(\$t3)

[0x04000110] ad6d0004 sw \$12, 4(\$11) ; 85: sw \$t4, 4(\$t3)

[0x04000114] 34190001 ori \$25, \$0, 1 ; 86: li \$t9, 1 # t9 = 1

[0x04000118] 08100037 4 0x040000dc [INNER\_LOOP]; 87: j INNER\_LOOP

0x0400000c 34240048 ori \$4, \$1, 1/2 [strLine]

[0x040000a0] 34020004 ori \$2, \$0, 4 ; 45: li \$v0, 4

[0x040000a4] 00000000 syscall ; 46: syscall

[0x040000a8] 34020004 ori \$2, \$0, 10 ; 48: li \$v0, 10

[0x040000ac] 00000000 syscall ; 49: syscall

[0x040000b0] 8caaa000 beq \$10, \$0, (\$5) ; 52: lr \$t2, 0(\$a1) # Load t2 = a[i]

[0x040000b4] 11400028 beq \$10, \$0, 160 [END\_FUNC-0x0040004b]

[0x040000b8] 20a50004 addi \$5, \$0, 160 [END\_FUNC-0x0040004b]

[0x040000c0] 22100001 addi \$16, \$16, 1 ; 55: addi \$s0, \$s0, 1 # Increase counter

[0x040000c4] 08100000 syscall ; 56: syscall

[0x040000c8] 00000000 beq \$10, \$0, (\$5) ; 57: j LEN

[0x040000cc] 08100032 j 0x0400000c [OUTER\_LOOP]; 60: j OUTER\_LOOP # Jump to OUTER\_LOOP

[0x040000d0] 01100024 beq \$1, \$0, 0 [END\_FUNC-0x00400000]

[0x040000d4] 34090001 ori \$25, \$0, 0 ; 63: bge \$t0, \$s0, END\_FUNC # Check if t0 = s0 = length of the array, stop the OUTER\_LOOP

[0x040000d8] 08100037 0r1 0x040000dc [INNER\_LOOP]; 66: j INNER\_LOOP # Jump to INNER\_LOOP

[0x040000e0] 10200000 beq \$1, \$0, 52 [CHECK\_LOOP-0x00400008]

[0x040000e4] 012a002a sit \$1, \$9, \$10 ; 68: sub \$t2, \$s0, \$t0 # t2 = s0 - t0

[0x040000e8] 340b0004 addi \$10, \$10, -1 ; 70: addi \$t2, \$t2, -1 # t2 = t2 - 1 = the current index

[0x040000f0] 714a0028 beq \$1, \$0, 136 [END\_FUNC-0x00400000]

[0x040000f4] 012a0082 sit \$1, \$9, \$10 ; 72: bge \$t1, \$t2, CHECK\_LOOP # Check if t1 >= t2, go to CHECK\_LOOP

[0x040000f8] 21290001 addi \$9, \$9, 1 ; 73: li \$t3, 4 # t3 = 4

[0x040000fc] 846c0000 lw \$12, 0(\$11) ; 74: mul \$t3, \$t3, \$t3 # t3 = t3\*t1

[0x04000100] 01ac002a alt \$1, \$13, \$12 ; 75: add \$t3, \$s1, \$t3 # t3 = a1 + t3 = a[i]

[0x04000104] 10200ff5 beq \$1, \$0, -44 [INNER\_LOOP-0x00400108]

[0x0400010c] ad6d0000 sw \$13, 0(\$11) ; 76: sw \$t5, 0(\$t3)

[0x04000110] ad6d0004 sw \$12, 4(\$11) ; 77: sw \$t4, 4(\$t3)

[0x04000114] 34190001 ori \$25, \$0, 1 ; 78: li \$t9, 1 # t9 = 1

[0x04000118] 08100037 4 0x040000dc [INNER\_LOOP]; 79: j INNER\_LOOP

0x0400000c 34240048 ori \$4, \$1, 1/2 [strLine]

[0x040000a0] 34020004 ori \$2, \$0, 4 ; 45: li \$v0, 4

[0x040000a4] 00000000 syscall ; 46: syscall

[0x040000a8] 34020004 ori \$2, \$0, 10 ; 48: li \$v0, 10

[0x040000ac] 00000000 syscall ; 49: syscall

[0x040000b0] 8caaa000 beq \$10, \$0, (\$5) ; 52: lr \$t2, 0(\$a1) # Load t2 = a[i]

[0x040000b4] 11400028 beq \$10, \$0, 160 [END\_FUNC-0x0040004b]

[0x040000b8] 20a50004 addi \$5, \$0, 160 [END\_FUNC-0x0040004b]

[0x040000c0] 22100001 addi \$16, \$16, 1 ; 55: addi \$s0, \$s0, 1 # Increase counter

[0x040000c4] 08100000 syscall ; 56: syscall

[0x040000c8] 00000000 beq \$10, \$0, (\$5) ; 57: j LEN

[0x040000cc] 08100032 j 0x0400000c [OUTER\_LOOP]; 60: j OUTER\_LOOP # Jump to OUTER\_LOOP

[0x040000d0] 01100024 beq \$1, \$0, 0 [END\_FUNC-0x00400000]

[0x040000d4] 34090001 ori \$25, \$0, 0 ; 63: bge \$t0, \$s0, END\_FUNC # Check if t0 = s0 = length of the array, stop the OUTER\_LOOP

[0x040000d8] 08100037 0r1 0x040000dc [INNER\_LOOP]; 66: j INNER\_LOOP # Jump to INNER\_LOOP

[0x040000e0] 10200000 beq \$1, \$0, 52 [CHECK\_LOOP-0x00400008]

[0x040000e4] 012a002a sit \$1, \$9, \$10 ; 68: sub \$t2, \$s0, \$t0 # t2 = s0 - t0

[0x040000e8] 340b0004 addi \$10, \$10, -1 ; 70: addi \$t2, \$t2, -1 # t2 = t2 - 1 = the current index

[0x040000f0] 714a0028 beq \$1, \$0, 136 [END\_FUNC-0x00400000]

[0x040000f4] 012a0082 sit \$1, \$9, \$10 ; 72: bge \$t1, \$t2, CHECK\_LOOP # Check if t1 >= t2, go to CHECK\_LOOP

[0x040000f8] 21290001 addi \$9, \$9, 1 ; 73: li \$t3, 4 # t3 = 4

[0x040000fc] 846c0000 lw \$12, 0(\$11) ; 74: mul \$t3, \$t3, \$t3 # t3 = t3\*t1

[0x04000100] 01ac002a alt \$1, \$13, \$12 ; 75: add \$t3, \$s1, \$t3 # t3 = a1 + t3 = a[i]

[0x04000104] 10200ff5 beq \$1, \$0, -44 [INNER\_LOOP-0x00400108]

[0x0400010c] ad6d0000 sw \$13, 0(\$11) ; 76: sw \$t5, 0(\$t3)

[0x04000110] ad6d0004 sw \$12, 4(\$11) ; 77: sw \$t4, 4(\$t3)

[0x04000114] 34190001 ori \$25, \$0, 1 ; 78: li \$t9, 1 # t9 = 1

[0x04000118] 08100037 4 0x040000dc [INNER\_LOOP]; 79: j INNER\_LOOP

0x0400000c 34240048 ori \$4, \$1, 1/2 [strLine]

[0x040000a0] 34020004 ori \$2, \$0, 4 ; 45: li \$v0, 4

[0x040000a4] 00000000 syscall ; 46: syscall

[0x040000a8] 34020004 ori \$2, \$0, 10 ; 48: li \$v0, 10

[0x040000ac] 00000000 syscall ; 49: syscall

[0x040000b0] 8caaa000 beq \$10, \$0, (\$5) ; 52: lr \$t2, 0(\$a1) # Load t2 = a[i]

[0x040000b4] 11400028 beq \$10, \$0, 160 [END\_FUNC-0x0040004b]

[0x040000b8] 20a50004 addi \$5, \$0, 160 [END\_FUNC-0x0040004b]

[0x040000c0] 22100001 addi \$16, \$16, 1 ; 55: addi \$s0, \$s0, 1 # Increase counter

[0x040000c4] 08100000 syscall ; 56: syscall

[0x040000c8] 00000000 beq \$10, \$0, (\$5) ; 57: j LEN

[0x040000cc] 08100032 j 0x0400000c [OUTER\_LOOP]; 60: j OUTER\_LOOP # Jump to OUTER\_LOOP

[0x040000d0] 01100024 beq \$1, \$0, 0 [END\_FUNC-0x00400000]

[0x040000d4] 34090001 ori \$25, \$0, 0 ; 63: bge \$t0, \$s0, END\_FUNC # Check if t0 = s0 = length of the array, stop the OUTER\_LOOP

[0x040000d8] 08100037 0r1 0x040000dc [INNER\_LOOP]; 66: j INNER\_LOOP # Jump to INNER\_LOOP

[0x040000e0] 10200000 beq \$1, \$0, 52 [CHECK\_LOOP-0x00400008]

[0x040000e4] 012a002a sit \$1, \$9, \$10 ; 68: sub \$t2, \$s0, \$t0 # t2 = s0 - t0

[0x040000e8] 340b0004 addi \$10, \$10, -1 ; 70: addi \$t2, \$t2, -1 # t2 = t2 - 1 = the current index

[0x040000f0] 714a0028 beq \$1, \$0, 136 [END\_FUNC-0x00400000]

[0x040000f4] 012a0082 sit \$1, \$9, \$10 ; 72: bge \$t1, \$t2, CHECK\_LOOP # Check if t1 >= t2, go to CHECK\_LOOP

[0x040000f8] 21290001 addi \$9, \$9, 1 ; 73: li \$t3, 4 # t3 = 4

[0x040000fc] 846c0000 lw \$12, 0(\$11) ; 74: mul \$t3, \$t3, \$t3 # t3 = t3\*t1

[0x04000100] 01ac002a alt \$1, \$13, \$12 ; 75: add \$t3, \$s1, \$t3 # t3 = a1 + t3 = a[i]

[0x04000104] 10200ff5 beq \$1, \$0, -44 [INNER\_LOOP-0x00400108]

[0x0400010c] ad6d0000 sw \$13, 0(\$11) ; 76: sw \$t5, 0(\$t3)

[0x04000110] ad6d0004 sw \$12, 4(\$11) ; 77: sw \$t4, 4(\$t3)

[0x04000114] 34190001 ori \$25, \$0, 1 ; 78: li \$t9, 1 # t9 = 1

[0x04000118] 08100037 4 0x040000dc [INNER\_LOOP]; 79: j INNER\_LOOP

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

## In the second loop inside INNER\_LOOP:

Int Regs [16]	Text
PC = 4000e0	[004000a8] 3402000a ori \$2, \$0, 10 ; 48: li \$v0, 10
EPC = 0	[004000ac] 00000000 syscall ; 49: syscall
Cause = 0	[004000b0] 8caaa000 lw \$10, 0(\$5) ; 52: lw \$t2, 0(\$a1) # Load t2 = a[1]
BadAddr = 0	[004000b4] 11400028 beq \$10, \$0, 160 [END_FUNC-0x004000b4]
Status = 3000ff10	[004000b8] 20a50004 addi \$16, \$16, 1 ; 55: addi \$a1, \$a1, 4 # Move the next element
HI = 0	[004000c1] 01000000 addi \$16, \$16, 1 ; 56: addi \$s0, \$s0, 1 # Increase counter
LO = 0	[004000c4] 08100002c 0x00400000 [LEN] ; 57: j LEN
R0 [r0] = 0	[004000cc] 10200022 beq \$10, \$0, 136 [END_FUNC-0x004000cc]
R1 [at] = 0	[004000d0] 341900000 ori \$25, \$0, 0 ; 64: li \$t9, 0 # t9 = 0
R2 [v0] = 4	[004000d4] 654000000 ori \$9, \$0, 0 ; 65: addi \$t1, \$t1, 0 # t1 = 0
R3 [v1] = 0	[004000d8] 08100037 j 0x004000dc [INNER_LOOP]; 66: j INNER_LOOP # Jump to INNER_LOOP
R4 [a0] = 10010048	[004000dc] 02085022 sub \$10, \$16, \$8 ; 69: sub \$t2, \$s0, \$t0 # t2 = 30 - t0
R5 [a1] = 1001004c	[004000e0] 214afffff addi \$10, \$10, -1 ; 70: addi \$t2, \$t2, -1 # t2 = t2 - 1 = the current index
R6 [a2] = 7fffffa8	[004000e4] 012aa082a slt \$1, \$9, \$10 ; 72: bge \$t1, \$t2, CHECK_LOOP # Check if t1 >= t2, go to CHECK_LOOP
R7 [a3] = 0	[004000e8] 102000000 beq \$1, \$0, 52 [CHECK_LOOP-0x004000e8]
R8 [t0] = 0	[004000ec] 340b0004 ori \$11, \$0, 4 ; 73: li \$t3, 4 # t3 = 4
R9 [t1] = 1	[004000f0] 71695802 mul \$11, \$11, \$9 ; 74: mul \$t3, \$t3, \$t1 # t3 = t3*t1
R10 [t2] = 0	[004000f4] 00ab5820 addi \$11, \$5, \$11 ; 75: addi \$t3, \$t3, \$t1 # t3 = a1 + t3 = a[1]
R11 [t3] = 1	[004000f8] 21290001 addi \$9, \$9, 1 ; 77: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R12 [t4] = 2b	[004000fc] 8d6c00000 lw \$12, 0(\$11) ; 79: lw \$t4, 0(\$t3)
R13 [t5] = 198f	[00400100] 8d6d00040 ori \$13, \$4(\$11) ; 80: lw \$t5, 4(\$t3)
R14 [t6] = 0	[00400104] 01a0802a slt \$1, \$13, \$12 ; 82: bne \$t4, \$t5, INNER_LOOP # if t4
R15 [t7] = 0	[00400108] 10200001 beq \$1, \$0, 160 [END_FUNC-0x00400001]
R16 [s0] = d	[0040010c] add6d0000 addi \$13, \$4(\$11) ; 84: j LEN, 0(\$t3)
R17 [s1] = 0	[00400110] add6e0004 sw \$12, 4(\$11) ; 85: sw \$t6, 4(\$t3)
R18 [s2] = 0	[00400114] 34190001 ori \$25, \$0, 1 ; 86: li \$t9, 1 # t9 = 1
R19 [s3] = 0	[00400118] 08100037 j 0x004000dc [INNER_LOOP]; 87: j INNER_LOOP
R20 [s4] = 0	[0040011c] 13200000 beq \$25, \$0, 56 [END_FUNC-0x0040001c]
R21 [s5] = 0	[00400120] 21080001 addi \$8, \$8, 1 ; 91: addi \$t0, \$t0, 1 # Increase the counter of OUTER_LOOP

Int Regs [16]	Text
PC = 4000e4	[004000a8] 3402000a ori \$2, \$0, 10 ; 48: li \$v0, 10
EPC = 0	[004000ac] 00000000 syscall ; 49: syscall
Cause = 0	[004000b0] 8caaa000 lw \$10, 0(\$5) ; 52: lw \$t2, 0(\$a1) # Load t2 = a[1]
BadAddr = 0	[004000b4] 11400028 beq \$10, \$0, 160 [END_FUNC-0x004000b4]
Status = 3000ff10	[004000b8] 20a50004 addi \$16, \$16, 1 ; 55: addi \$a1, \$a1, 4 # Move the next element
HI = 0	[004000c1] 01000000 addi \$16, \$16, 1 ; 56: addi \$s0, \$s0, 1 # Increase counter
LO = 0	[004000c4] 08100002c 0x00400000 [LEN] ; 57: j LEN
R0 [r0] = 0	[004000cc] 10200022 beq \$10, \$0, 136 [END_FUNC-0x004000cc]
R1 [at] = 0	[004000d0] 341900000 ori \$25, \$0, 0 ; 64: li \$t9, 0 # t9 = 0
R2 [v0] = 4	[004000d4] 654000000 ori \$9, \$0, 0 ; 65: addi \$t1, \$t1, 0 # t1 = 0
R3 [v1] = 0	[004000d8] 08100037 j 0x004000dc [INNER_LOOP]; 66: j INNER_LOOP # Jump to INNER_LOOP
R4 [a0] = 10010048	[004000dc] 02085022 sub \$10, \$16, \$8 ; 69: sub \$t2, \$s0, \$t0 # t2 = s0 - t0
R5 [a1] = 1001004c	[004000e0] 214afffff addi \$10, \$10, -1 ; 70: addi \$t2, \$t2, -1 # t2 = t2 - 1 = the current index
R6 [a2] = 7fffffa8	[004000e4] 012aa082a slt \$1, \$9, \$10 ; 72: bge \$t1, \$t2, CHECK_LOOP # Check if t1 >= t2, go to CHECK_LOOP
R7 [a3] = 0	[004000e8] 10200001 beq \$1, \$0, 52 [CHECK_LOOP-0x004000e8]
R8 [t0] = 0	[004000ec] 340b0004 ori \$11, \$0, 4 ; 73: li \$t3, 4 # t3 = 4
R9 [t1] = 1	[004000f0] 71695802 mul \$11, \$11, \$9 ; 74: mul \$t3, \$t3, \$t1 # t3 = t3*t1
R10 [t2] = 0	[004000f4] 00ab5820 addi \$11, \$5, \$11 ; 75: addi \$t3, \$t3, \$t1 # t3 = a1 + t3 = a[1]
R11 [t3] = 1	[004000f8] 21290001 addi \$9, \$9, 1 ; 77: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R12 [t4] = 2b	[00400100] 8d6c00000 lw \$12, 0(\$11) ; 79: lw \$t4, 0(\$t3)
R13 [t5] = 198f	[00400104] 8d6d00040 ori \$13, \$4(\$11) ; 80: lw \$t5, 4(\$t3)
R14 [t6] = 0	[00400108] 01a0802a slt \$1, \$13, \$12 ; 82: bne \$t4, \$t5, INNER_LOOP # if t4
R15 [t7] = 0	[00400110] 1020eff5 beq \$1, \$0, -44 [INNER_LOOP-0x00400108]
R16 [s0] = d	[00400114] add6d0000 sw \$12, 4(\$11) ; 84: sw \$t5, 0(\$t3)
R17 [s1] = 0	[00400118] add6e0004 sw \$12, 4(\$11) ; 85: sw \$t4, 4(\$t3)
R18 [s2] = 0	[0040011c] 34190001 ori \$25, \$0, 1 ; 86: li \$t9, 1 # t9 = 1
R19 [s3] = 0	[00400120] 08100037 j 0x004000dc [INNER_LOOP]; 87: j INNER_LOOP
R20 [s4] = 0	[00400124] 13200000 beq \$25, \$0, 56 [END_FUNC-0x0040001c]
R21 [s5] = 0	[00400128] 21080001 addi \$8, \$8, 1 ; 91: addi \$t0, \$t0, 1 # Increase the counter of OUTER_LOOP

Int Regs [16]	Text
Cause = 0	[004000a8] 3402000a ori \$2, \$0, 10 ; 48: li \$v0, 10
BadAddr = 0	[004000ac] 00000000 syscall ; 49: syscall
Status = 3000ff10	[004000b0] 8caaa000 lw \$10, 0(\$5) ; 52: lw \$t2, 0(\$a1) # Load t2 = a[1]
HI = 0	[004000b4] 11400028 beq \$10, \$0, 160 [END_FUNC-0x004000b4]
LO = 0	[004000b8] 20a50004 addi \$16, \$16, 1 ; 55: addi \$a1, \$a1, 4 # Move the next element
R0 [r0] = 0	[004000c1] 022100001 addi \$16, \$16, 1 ; 56: addi \$s0, \$s0, 1 # Increase counter
R1 [at] = 0	[004000c4] 08100002c 0x00400000 [LEN] ; 57: j LEN
R2 [v0] = 4	[004000cc] 10200001 beq \$1, \$0, 52 [CHECK_LOOP-0x00400001]
R3 [v1] = 0	[004000d0] 341900000 ori \$25, \$0, 0 ; 64: li \$t9, 0 # t9 = 0
R4 [a0] = 10010048	[004000d4] 654000000 ori \$9, \$0, 0 ; 65: addi \$t1, \$t1, 0 # t1 = 0
R5 [a1] = 1001004c	[004000d8] 08100037 j 0x004000dc [INNER_LOOP]; 66: j INNER_LOOP # Jump to INNER_LOOP
R6 [a2] = 7fffffa8	[004000dc] 02085022 sub \$10, \$16, \$8 ; 69: sub \$t2, \$s0, \$t0 # t2 = s0 - t0
R7 [a3] = 0	[004000e0] 214afffff addi \$10, \$10, -1 ; 70: addi \$t2, \$t2, -1 # t2 = t2 - 1 = the current index
R8 [t0] = 0	[004000e4] 012aa082a slt \$1, \$9, \$10 ; 72: bge \$t1, \$t2, CHECK_LOOP # Check if t1 >= t2, go to CHECK_LOOP
R9 [t1] = 1	[004000e8] 10200001 beq \$1, \$0, 52 [CHECK_LOOP-0x004000e8]
R10 [t2] = c	[004000fc] 340b0004 ori \$11, \$0, 4 ; 73: li \$t3, 4 # t3 = 4
R11 [t3] = 4	[00400100] 01a0802a mul \$11, \$11, \$9 ; 74: mul \$t3, \$t3, \$t1 # t3 = t3*t1
R12 [t4] = 2b	[00400104] 00ab5820 addi \$11, \$5, \$11 ; 75: addi \$t3, \$t3, \$t1 # t3 = a1 + t3 = a[1]
R13 [t5] = 198f	[00400108] 012aa082a addi \$9, \$9, 1 ; 77: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R14 [t6] = 0	[00400110] 01a0802a slt \$1, \$13, \$12 ; 79: bge \$t4, \$t5, INNER_LOOP # if t4
R15 [t7] = 0	[00400114] 1020eff5 beq \$1, \$0, -44 [INNER_LOOP-0x00400108]
R16 [s0] = d	[00400118] add6d0000 sw \$12, 4(\$11) ; 84: sw \$t5, 0(\$t3)
R17 [s1] = 0	[0040011c] add6e0004 sw \$12, 4(\$11) ; 85: sw \$t4, 4(\$t3)
R18 [s2] = 0	[00400120] 34190001 ori \$25, \$0, 1 ; 86: li \$t9, 1 # t9 = 1
R19 [s3] = 0	[00400124] 08100037 j 0x004000dc [INNER_LOOP]; 87: j INNER_LOOP
R20 [s4] = 0	[00400128] 13200000 beq \$25, \$0, 56 [END_FUNC-0x0040001c]
R21 [s5] = 0	[00400132] 21080001 addi \$8, \$8, 1 ; 91: addi \$t0, \$t0, 1 # Increase the counter of OUTER_LOOP

Int Regs [16]	S x Text
R2 [v0] = 4	[0040000a8] 3402000a ori \$2, \$0, 10 ; 48: li \$v0, 10
R3 [v1] = 0	[0040000ac] 00000000 syscall ; 49: syscall
R4 [a0] = 10010048	[0040000b0] 8caa0000 lw \$10, 0(\$5) ; 52: lr \$t2, 0(\$a1) # Load t2 = a[i]
R5 [a1] = 1001004c	[0040000b4] 11400028 beg \$10, \$0, 160 [END_FUNC-0x004000b4]
R6 [a2] = 7ffff1a8	[0040000b8] 20a50004 addi \$5, \$5, 4 ; 55: addi \$a1, \$a1, 4 # Move the next element
R7 [a3] = 0	[0040000c] 22100001 addi \$16, \$16, 1 ; 56: addi \$s0, \$s0, 1 # Increase counter
R8 [t0] = 0	[0040000c0] 0810002 addi \$1, \$0, 0 ; 57: j LEN
R9 [t1] = 1	[0040000c4] 08100032 j 0x00400008 [OUTER_LOOP] ; 60: j OUTER_LOOP # Jump to OUTER_LOOP
R10 [t2] = c	[0040000c8] 01100082a slt \$1, \$8, \$16 ; 63: bge \$t0, \$s0, END_FUNC # Check if t0 = s0 = length of the array, stop the OUTER_LOOP
R11 [t3] = 10010050	[0040000cc] 10200022 beg \$1, \$1, 136 [END_FUNC-0x004000cc]
R12 [t4] = 2b	[0040000d0] 34190000 ori \$25, \$0, 0 ; 64: li \$t9, 0 # t9 = 0
R13 [t5] = 198f	[0040000d4] 34090000 ori \$9, \$0, 0 ; 65: li \$t1, 0 # t1 = 0
R14 [t6] = 0	[0040000d8] 08100037 j 0x0040000d [INNER_LOOP] ; 66: j INNER_LOOP # Jump to INNER_LOOP
R15 [t7] = 0	[0040000dc] 02085022 sub \$10, \$16, \$8 ; 69: sub \$t2, \$s0, \$t0 # t2 = s0 - t0
R16 [s0] = d	[0040000e0] 214affff addi \$10, \$10, -1 ; 70: addi \$t2, \$t2, -1 # t2 = t2 - 1 = the current index
R17 [s1] = 0	[0040000e4] 012a082a slt \$1, \$9, \$10 ; 72: bge \$t1, \$t2, CHECK_LOOP # Check if t1 >= t2, go to CHECK_LOOP
R18 [s2] = 0	[0040000e8] 10200008 beg \$1, \$0, 56 [CHECK_LOOP-0x004000e8]
R19 [s3] = 0	[0040000ec] 340b0000 ori \$11, \$0, 4 ; 73: li \$t3, 4 # t3 = 4
R20 [sp] = 10008000	[0040000f0] 71695800 mul \$11, \$11, \$9 ; 74: mul \$t3, \$t3, \$t1 # t3 = a1 * t1
R21 [s8] = 0	[0040000f4] 09ab5820 addi \$11, \$5, \$11 ; 75: add \$t3, \$a1, \$t3 # a1 + t3 = a[i]
R22 [s9] = 0	[0040000f8] 21290001 addi \$9, \$9, 1 ; 77: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R23 [s7] = 0	[0040000f8c] 8d6d0000 lw \$12, 0(\$11) ; 79: lr \$t4, 0(\$t3)
R24 [t8] = 0	[00400100] 8d6d0004 lw \$13, 4(\$11) ; 80: lr \$t5, 4(\$t3)
R25 [t9] = 0	[00400104] 01ac082a slt \$1, \$13, \$12 ; 82: ble \$t4, \$t5, INNER_LOOP # if t4
R26 [x0] = 0	[00400108] 10200ff5 add \$1, \$0, -44 [INNER_LOOP-0x00400108]
R27 [x1] = 0	[0040010c] ad6d0000 sw \$13, 0(\$11) ; 84: sw \$t5, 0(\$t3)
R28 [gp] = 10008000	[00400110] ad6d0004 sw \$12, 4(\$11) ; 85: sw \$t4, 4(\$t3)
R29 [sp] = 7ffff19c	[00400114] 34190001 ori \$25, \$0, 1 ; 86: li \$t9, 1 # t9 = 1
R30 [s8] = 0	[00400118] 08100037 j 0x0040000d [INNER_LOOP] ; 87: j INNER_LOOP
R31 [ra] = 40007c	[0040011c] 1320000e beg \$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]	S x Text
R2 [v0] = 4	[0040000a8] 3402000a ori \$2, \$0, 10 ; 48: li \$v0, 10
R3 [v1] = 0	[0040000ac] 00000000 syscall ; 49: syscall
R4 [a0] = 10010048	[0040000b0] 8caa0000 lw \$10, 0(\$5) ; 52: lr \$t2, 0(\$a1) # Load t2 = a[i]
R5 [a1] = 1001004c	[0040000b4] 11400028 beg \$10, \$0, 160 [END_FUNC-0x004000b4]
R6 [a2] = 7ffff1a8	[0040000b8] 20a50004 addi \$5, \$5, 4 ; 55: addi \$a1, \$a1, 4 # Move the next element
R7 [a3] = 0	[0040000c] 22100001 addi \$16, \$16, 1 ; 56: addi \$s0, \$s0, 1 # Increase counter
R8 [t0] = 0	[0040000c0] 0810002 addi \$1, \$0, 0 ; 57: j LEN
R9 [t1] = 0	[0040000c4] 08100032 j 0x00400008 [OUTER_LOOP] ; 60: j OUTER_LOOP # Jump to OUTER_LOOP
R10 [t2] = c	[0040000c8] 01100082a slt \$1, \$8, \$16 ; 63: bge \$t0, \$s0, END_FUNC # Check if t0 = s0 = length of the array, stop the OUTER_LOOP
R11 [t3] = 10010050	[0040000cc] 10200022 beg \$1, \$1, 136 [END_FUNC-0x004000cc]
R12 [t4] = 2b	[0040000d0] 34190000 ori \$25, \$0, 0 ; 64: li \$t9, 0 # t9 = 0
R13 [t5] = 198f	[0040000d4] 34090000 ori \$9, \$0, 0 ; 65: li \$t1, 0 # t1 = 0
R14 [t6] = 0	[0040000d8] 08100037 j 0x0040000d [INNER_LOOP] ; 66: j INNER_LOOP # Jump to INNER_LOOP
R15 [t7] = 0	[0040000dc] 02085022 sub \$10, \$16, \$8 ; 69: sub \$t2, \$s0, \$t0 # t2 = s0 - t0
R16 [s0] = d	[0040000e0] 214affff addi \$10, \$10, -1 ; 70: addi \$t2, \$t2, -1 # t2 = t2 - 1 = the current index
R17 [s1] = 0	[0040000e4] 012a082a slt \$1, \$9, \$10 ; 72: bge \$t1, \$t2, CHECK_LOOP # Check if t1 >= t2, go to CHECK_LOOP
R18 [s2] = 0	[0040000e8] 10200008 beg \$1, \$0, 52 [CHECK_LOOP-0x004000e8]
R19 [s3] = 0	[0040000ec] 340b0004 ori \$11, \$0, 4 ; 73: li \$t3, 4 # t3 = 4
R20 [sp] = 10008000	[0040000f0] 71695802 mul \$11, \$11, \$9 ; 74: mul \$t3, \$t3, \$t1 # t3 = a1 * t1
R21 [s8] = 0	[0040000f4] 09ab5820 addi \$11, \$5, \$11 ; 75: add \$t3, \$a1, \$t3 # a1 + t3 = a[i]
R22 [s9] = 0	[0040000f8] 21290001 addi \$9, \$9, 1 ; 77: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R23 [s7] = 0	[0040000f8c] 8d6d0000 lw \$12, 0(\$11) ; 79: lr \$t4, 0(\$t3)
R24 [t8] = 0	[00400100] 8d6d0004 lw \$13, 4(\$11) ; 80: lr \$t5, 4(\$t3)
R25 [t9] = 0	[00400104] 01ac082a slt \$1, \$13, \$12 ; 82: ble \$t4, \$t5, INNER_LOOP # if t4
R26 [x0] = 0	[00400108] 10200ff5 add \$1, \$0, -44 [INNER_LOOP-0x00400108]
R27 [x1] = 0	[0040010c] ad6d0000 sw \$13, 0(\$11) ; 84: sw \$t5, 0(\$t3)
R28 [gp] = 10008000	[00400110] ad6d0004 sw \$12, 4(\$11) ; 85: sw \$t4, 4(\$t3)
R29 [sp] = 7ffff19c	[00400114] 34190001 ori \$25, \$0, 1 ; 86: li \$t9, 1 # t9 = 1
R30 [s8] = 0	[00400118] 08100037 j 0x0040000d [INNER_LOOP] ; 87: j INNER_LOOP
R31 [ra] = 40007c	[0040011c] 1320000e beg \$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]	S x Text
R2 [v0] = 4	[0040000a8] 3402000a ori \$2, \$0, 10 ; 48: li \$v0, 10
R3 [v1] = 0	[0040000ac] 00000000 syscall ; 49: syscall
R4 [a0] = 10010048	[0040000b0] 8caa0000 lw \$10, 0(\$5) ; 52: lr \$t2, 0(\$a1) # Load t2 = a[i]
R5 [a1] = 1001004c	[0040000b4] 11400028 beg \$10, \$0, 160 [END_FUNC-0x004000b4]
R6 [a2] = 7ffff1a8	[0040000b8] 20a50004 addi \$5, \$5, 4 ; 55: addi \$a1, \$a1, 4 # Move the next element
R7 [a3] = 0	[0040000c] 22100001 addi \$16, \$16, 1 ; 56: addi \$s0, \$s0, 1 # Increase counter
R8 [t0] = 0	[0040000c0] 0810002 addi \$1, \$0, 0 ; 57: j LEN
R9 [t1] = 0	[0040000c4] 08100032 j 0x00400008 [OUTER_LOOP] ; 60: j OUTER_LOOP # Jump to OUTER_LOOP
R10 [t2] = c	[0040000c8] 01100082a slt \$1, \$8, \$16 ; 63: bge \$t0, \$s0, END_FUNC # Check if t0 = s0 = length of the array, stop the OUTER_LOOP
R11 [t3] = 10010050	[0040000cc] 10200022 beg \$1, \$1, 136 [END_FUNC-0x004000cc]
R12 [t4] = 2b	[0040000d0] 34190000 ori \$25, \$0, 0 ; 64: li \$t9, 0 # t9 = 0
R13 [t5] = 198f	[0040000d4] 34090000 ori \$9, \$0, 0 ; 65: li \$t1, 0 # t1 = 0
R14 [t6] = 0	[0040000d8] 08100037 j 0x0040000d [INNER_LOOP] ; 66: j INNER_LOOP # Jump to INNER_LOOP
R15 [t7] = 0	[0040000dc] 02085022 sub \$10, \$16, \$8 ; 69: sub \$t2, \$s0, \$t0 # t2 = s0 - t0
R16 [s0] = d	[0040000e0] 214affff addi \$10, \$10, -1 ; 70: addi \$t2, \$t2, -1 # t2 = t2 - 1 = the current index
R17 [s1] = 0	[0040000e4] 012a082a slt \$1, \$9, \$10 ; 72: bge \$t1, \$t2, CHECK_LOOP # Check if t1 >= t2, go to CHECK_LOOP
R18 [s2] = 0	[0040000e8] 10200008 beg \$1, \$0, 52 [CHECK_LOOP-0x004000e8]
R19 [s3] = 0	[0040000ec] 340b0004 ori \$11, \$0, 4 ; 73: li \$t3, 4 # t3 = 4
R20 [sp] = 10008000	[0040000f0] 71695802 mul \$11, \$11, \$9 ; 74: mul \$t3, \$t3, \$t1 # t3 = a1 * t1
R21 [s8] = 0	[0040000f4] 09ab5820 addi \$11, \$5, \$11 ; 75: add \$t3, \$a1, \$t3 # a1 + t3 = a[i]
R22 [s9] = 0	[0040000f8] 21290001 addi \$9, \$9, 1 ; 77: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R23 [s7] = 0	[0040000f8c] 8d6d0000 lw \$12, 0(\$11) ; 79: lr \$t4, 0(\$t3)
R24 [t8] = 0	[00400100] 8d6d0004 lw \$13, 4(\$11) ; 80: lr \$t5, 4(\$t3)
R25 [t9] = 0	[00400104] 01ac082a slt \$1, \$13, \$12 ; 82: ble \$t4, \$t5, INNER_LOOP # if t4
R26 [x0] = 0	[00400108] 10200ff5 add \$1, \$0, -44 [INNER_LOOP-0x00400108]
R27 [x1] = 0	[0040010c] ad6d0000 sw \$13, 0(\$11) ; 84: sw \$t5, 0(\$t3)
R28 [gp] = 10008000	[00400110] ad6d0004 sw \$12, 4(\$11) ; 85: sw \$t4, 4(\$t3)
R29 [sp] = 7ffff19c	[00400114] 34190001 ori \$25, \$0, 1 ; 86: li \$t9, 1 # t9 = 1
R30 [s8] = 0	[00400118] 08100037 j 0x0040000d [INNER_LOOP] ; 87: j INNER_LOOP
R31 [ra] = 40007c	[0040011c] 1320000e beg \$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]	Text
R2 [v0] = 4	[0040000a8] 3402000a ori \$2, \$0, 10 ; 48: li \$v0, 10
R3 [v1] = 0	[0040000a9] 8caaa000 lw \$10, 0(\$5) ; 49: syscall
R4 [a0] = 10010048	[0040000b0] 11400028 beq \$10, \$0, 160 [END_FUNC-0x004000b4]
R5 [a1] = 1001004c	[0040000b1] 00a0b50004 addi \$5, \$5, 4 ; 52: addi \$a1, \$a1, 4 # Move the next element
R6 [a2] = 7fffffa8	[0040000b2] 22100001 addi \$1, \$1, 1 ; 53: addi \$s0, \$s0, 1 # Increase counter
R7 [a3] = 0	[0040000b3] 0811002c j 0x00400008 [LEN] ; 57: j LEN
R8 [t0] = 0	[0040000b4] 00100032 j 0x00400009 [OUTER_LOOP] ; 60: j OUTER_LOOP # Jump to OUTER_LOOP
R9 [t1] = 2	[0040000b5] 01110082a slt \$1, \$8, \$16 ; 63: bge \$t0, \$s0, END_FUNC # Check if t0 = s0 = length of the array, stop the OUTER_LOOP
R10 [t2] = c	[0040000b6] 10200022 beq \$1, \$0, 136 [END_FUNC-0x004000c0]
R11 [t3] = 10010050	[0040000b7] 00110082a slt \$1, \$8, \$16 ; 64: li \$t9, 0 # t9 = 0
R12 [t4] = 198f	[0040000b8] 34190000 ori \$9, \$0, 0 ; 65: li \$t1, 0 # t1 = 0
R13 [t5] = 22	[0040000b9] 08110037 j 0x0040000dc [INNER_LOOP] ; 66: j INNER_LOOP # Jump to INNER_LOOP
R14 [t6] = 0	[0040000ba] 02085022 sub \$10, \$16, \$8 ; 67: sub \$t2, \$s0, \$t0 # t2 = s0 - t0
R15 [t7] = 0	[0040000bb] 214affff addi \$10, \$10, -1 ; 68: addi \$t2, \$t2, -1 # t2 = t2 - 1 = the current index
R16 [s0] = d	[0040000bc] 012a082a slt \$1, \$9, \$10 ; 69: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R17 [s1] = 0	[0040000bd] 1020000d beq \$1, \$0, 52 [CHECK_LOOP-0x004000e8]
R18 [s2] = 0	[0040000be] 00110082a slt \$1, \$8, \$16 ; 70: bge \$t1, \$t2, CHECK_LOOP # Check if t1 >= t2, go to CHECK_LOOP
R19 [s3] = 0	[0040000bf] 340b0004 ori \$11, \$0, 4 ; 73: li \$t3, 4 # t3 = 4
R20 [s4] = 0	[0040000c0] 71695802 mul \$1, \$11, \$9 ; 74: mul \$t3, \$t3, \$t1 # t3 = t3*t1
R21 [s5] = 0	[0040000c1] 00ab5820 addi \$11, \$5, \$11 ; 75: addi \$s0, \$s0, 5 # t5 = a[i]
R22 [s6] = 0	[0040000c2] 21290001 addi \$9, \$9, 1 ; 76: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R23 [s7] = 0	[0040000c3] 0d6c0000 lw \$12, 0(\$11) ; 77: li \$t4, 0(\$t3)
R24 [t8] = 0	[0040000c4] 00110082a lw \$13, 4(\$11) ; 78: lw \$t5, 4(\$t3)
R25 [t9] = 0	[00400104] 01ac082a slt \$1, \$13, \$12 ; 79: blt \$t4, \$t5, 4(\$t3)
R26 [k0] = 0	[00400105] 1020000d beq \$1, \$0, 136 [END_FUNC-0x00400108]
R27 [k1] = 0	[00400106] ad6d0000 sw \$13, 0(\$11) ; 80: sw \$t5, 0(\$t3)
R28 [gp] = 10008000	[00400107] 00a0b50004 addi \$11, \$5, \$11 ; 81: addi \$s0, \$s0, 4 # t5 = 4(\$t3)
R29 [sp] = 7ffff19c	[00400108] 34190001 ori \$25, \$0, 1 ; 82: li \$t9, 1 # t9 = 1
R30 [s8] = 0	[00400118] 08100037 j 0x004000dc [INNER_LOOP] ; 87: j INNER_LOOP
R31 [ra] = 40007c	[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]	Text
R2 [v0] = 4	[0040000a8] 3402000a ori \$2, \$0, 10 ; 48: li \$v0, 10
R3 [v1] = 0	[0040000a9] 8caaa000 lw \$10, 0(\$5) ; 49: syscall
R4 [a0] = 10010048	[0040000b0] 11400028 beq \$10, \$0, 160 [END_FUNC-0x004000b4]
R5 [a1] = 1001004c	[0040000b1] 00a0b50004 addi \$5, \$5, 4 ; 52: addi \$a1, \$a1, 4 # Move the next element
R6 [a2] = 7fffffa8	[0040000b2] 22100001 addi \$1, \$1, 1 ; 53: addi \$s0, \$s0, 1 # Increase counter
R7 [a3] = 0	[0040000b3] 0811002c j 0x00400008 [LEN] ; 57: j LEN
R8 [t0] = 0	[0040000b4] 00110082a slt \$1, \$8, \$16 ; 60: j OUTER_LOOP # Jump to OUTER_LOOP
R9 [t1] = 2	[0040000b5] 10200022 beq \$1, \$0, 136 [END_FUNC-0x004000c0]
R10 [t2] = c	[0040000b6] 00110082a slt \$1, \$8, \$16 ; 64: li \$t9, 0 # t9 = 0
R11 [t3] = 10010050	[0040000b7] 34190000 ori \$9, \$0, 0 ; 65: li \$t1, 0 # t1 = 0
R12 [t4] = 198f	[0040000b8] 08110037 j 0x004000dc [INNER_LOOP] ; 66: j INNER_LOOP # Jump to INNER_LOOP
R13 [t5] = 22	[0040000b9] 02085022 sub \$10, \$16, \$8 ; 67: sub \$t2, \$s0, \$t0 # t2 = s0 - t0
R14 [t6] = 0	[0040000bb] 214affff addi \$10, \$10, -1 ; 68: addi \$t2, \$t2, -1 # t2 = t2 - 1 = the current index
R15 [t7] = 0	[0040000bc] 012a082a slt \$1, \$9, \$10 ; 69: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R16 [s0] = d	[0040000bd] 1020000d beq \$1, \$0, 52 [CHECK_LOOP-0x004000e8]
R17 [s1] = 0	[0040000be] 00110082a slt \$1, \$13, \$12 ; 70: bge \$t1, \$t2, CHECK_LOOP # Check if t1 >= t2, go to CHECK_LOOP
R18 [s2] = 0	[0040000bf] 340b0004 ori \$11, \$0, 4 ; 73: li \$t3, 4 # t3 = 4
R19 [s3] = 0	[0040000c0] 71695802 mul \$1, \$11, \$9 ; 74: mul \$t3, \$t3, \$t1 # t3 = t3*t1
R20 [s4] = 0	[0040000c1] 00ab5820 addi \$11, \$5, \$11 ; 75: addi \$s0, \$s0, 5 # t5 = a[i]
R21 [s5] = 0	[0040000c2] 21290001 addi \$9, \$9, 1 ; 76: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R22 [s6] = 0	[0040000c3] 0d6c0000 lw \$12, 0(\$11) ; 77: li \$t4, 0(\$t3)
R23 [s7] = 0	[0040000c4] 00110082a lw \$13, 4(\$11) ; 78: lw \$t5, 4(\$t3)
R24 [t8] = 0	[0040000c5] 01ac082a slt \$1, \$13, \$12 ; 79: blt \$t4, \$t5, 4(\$t3)
R25 [t9] = 0	[0040000c6] 1020000d beq \$1, \$0, -44 [INNER_LOOP-0x00400108]
R26 [k0] = 0	[0040000c7] ad6d0000 sw \$13, 0(\$11) ; 80: sw \$t5, 0(\$t3)
R27 [k1] = 0	[0040000c8] 00a0b50004 addi \$11, \$5, \$11 ; 81: addi \$s0, \$s0, 4 # t5 = 4(\$t3)
R28 [gp] = 10008000	[0040000c9] 34190001 ori \$25, \$0, 1 ; 82: li \$t9, 1 # t9 = 1
R29 [sp] = 7ffff19c	[004000118] 08100037 j 0x004000dc [INNER_LOOP] ; 87: j INNER_LOOP
R30 [s8] = 0	[00400011c] 1320000e beq \$25, \$0, 56 [END_FUNC-0x0040011c]
R31 [ra] = 40007c	[004000120] 21080001 addi \$8, \$8, 1 ; 91: addi \$t0, \$t0, 1 # Increase the counter of OUTER_LOOP

Those above steps happened when it satisfied the condition for swapping.

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

Int Regs [16]	Text
R2 [v0] = 4	[0040000a8] 8caaa000 lw \$10, 0(\$5) ; 52: lw \$t2, 0(\$a1) # Load t2 = a[1]
R3 [v1] = 0	[0040000a9] 0000000c syscall ; 49: syscall
R4 [a0] = 10010048	[0040000ba] 11400028 beq \$10, \$0, 160 [END_FUNC-0x004000b4]
R5 [a1] = 1001004c	[0040000bb] 00a0b50004 addi \$5, \$5, 4 ; 53: addi \$a1, \$a1, 4 # Move the next element
R6 [a2] = 7fffffa8	[0040000bc] 22100001 addi \$1, \$1, 1 ; 54: addi \$s0, \$s0, 1 # Increase counter
R7 [a3] = 0	[0040000bd] 0811002c j 0x00400008 [LEN] ; 57: j LEN
R8 [t0] = 0	[0040000be] 00110082a slt \$1, \$8, \$16 ; 60: j OUTER_LOOP # Jump to OUTER_LOOP
R9 [t1] = c	[0040000bf] 10200022 beq \$1, \$0, 136 [END_FUNC-0x004000c0]
R10 [t2] = c	[0040000c0] 00110082a slt \$1, \$8, \$16 ; 64: li \$t9, 0 # t9 = 0
R11 [t3] = 10010078	[0040000c1] 34190000 ori \$9, \$0, 0 ; 65: li \$t1, 0 # t1 = 0
R12 [t4] = 198f	[0040000c2] 08110037 j 0x004000dc [INNER_LOOP] ; 66: j INNER_LOOP # Jump to INNER_LOOP
R13 [t5] = 38	[0040000c3] 02085022 sub \$10, \$16, \$8 ; 67: sub \$t2, \$s0, \$t0 # t2 = s0 - t0
R14 [t6] = 0	[0040000c4] 214affff addi \$10, \$10, -1 ; 68: addi \$t2, \$t2, -1 # t2 = t2 - 1 = the current index
R15 [t7] = 0	[0040000c5] 012a082a slt \$1, \$9, \$10 ; 69: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R16 [s0] = d	[0040000c6] 1020000d beq \$1, \$0, 52 [CHECK_LOOP-0x004000e8]
R17 [s1] = 0	[0040000c7] 340b0004 ori \$11, \$0, 4 ; 73: li \$t3, 4 # t3 = 4
R18 [s2] = 0	[0040000c8] 71695802 mul \$1, \$11, \$9 ; 74: mul \$t3, \$t3, \$t1 # t3 = t3*t1
R19 [s3] = 0	[0040000c9] 00ab5820 addi \$11, \$5, \$11 ; 75: addi \$s0, \$s0, 5 # t5 = a[i]
R20 [s4] = 0	[0040000ca] 21290001 addi \$9, \$9, 1 ; 76: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R21 [s5] = 0	[0040000cb] 0d6c0000 lw \$12, 0(\$11) ; 77: li \$t4, 0(\$t3)
R22 [s6] = 0	[0040000cc] 00110082a lw \$13, 4(\$11) ; 78: lw \$t5, 4(\$t3)
R23 [s7] = 0	[0040000cd] 01ac082a slt \$1, \$13, \$12 ; 79: blt \$t4, \$t5, 4(\$t3)
R24 [t8] = 0	[0040000ce] 1020efff5 beq \$1, \$0, -44 [INNER_LOOP-0x00400108]
R25 [t9] = 1	[0040000cf] ad6d0000 sw \$13, 0(\$11) ; 80: sw \$t5, 0(\$t3)
R26 [k0] = 0	[0040000d0] 00a0b50004 addi \$11, \$5, \$11 ; 81: addi \$s0, \$s0, 4 # t5 = 4(\$t3)
R27 [k1] = 0	[0040000d1] 34190001 ori \$25, \$0, 1 ; 82: li \$t9, 1 # t9 = 1
R28 [gp] = 10008000	[0040000d2] 08100037 j 0x004000dc [INNER_LOOP] ; 87: j INNER_LOOP
R29 [sp] = 7ffff19c	[0040000d6] 1320000e beq \$25, \$0, 56 [END_FUNC-0x0040011c]
R30 [s8] = 0	[004000100] 21080001 addi \$8, \$8, 1 ; 91: addi \$t0, \$t0, 1 # Increase the counter of OUTER_LOOP
R31 [ra] = 40007c	[004000104] 00110082a slt \$1, \$8, \$16 ; 92: j OUTER_LOOP

When \$t1 >= \$t2, Increasing the counter \$t0 = \$t0 + 1. Then, it will jump back to OUTER\_LOOP and repeat those step until \$t0 = \$t0

## Code(Quick\_Sort):

## 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)

R2 [v0] = 4	[00400014] 0c100009 jal 0x00400024 [main] ; 188: jal main
R3 [v1] = 0	[00400018] 00000000 nop ; 189: nop
R4 [a0] = 1001004c	[0040001c] 3402000a ori \$2, \$0, 10 ; 191: li \$v0 10
R5 [a1] = 0	[00400020] 0000000c syscall ; 192: syscall # syscall 10 (exit)
R6 [a2] = d	[00400024] 3c041001 lui \$4, 4097 [strNameId] ; 14: la \$a0, strNameId
R7 [a3] = 0	[00400028] 34020004 ori \$2, \$0, 4 ; 15: li \$v0, 4
R8 [t0] = 0	[0040002c] 0000000c syscall ; 16: syscall
R9 [t1] = 0	[00400030] 3c011001 lui \$1, 4097 [printInitialArray]
R10 [t2] = 0	[00400034] 34240021 ori \$4, \$1, 33 [printInitialArray]
R11 [t3] = 0	[00400038] 34020004 ori \$2, \$0, 4 ; 19: li \$v0, 4
R12 [t4] = 0	[0040003c] 0000000c syscall ; 20: syscall
R13 [t5] = 0	[00400040] 3c011001 lui \$1, 4097 [arr] ; 22: la \$a1, arr # \$a1 = address of arr
R14 [t6] = 0	[00400044] 3425004c ori \$5, \$1, 76 [arr]
R15 [t7] = 0	[00400048] 0c100072 jal 0x004001c8 [DISPLAY] ; 23: jal DISPLAY # Print the array
R16 [s0] = 0	[0040004c] 3c011001 lui \$1, 4097 [strLine]
R17 [s1] = 0	[00400050] 34240048 ori \$4, \$1, 72 [strLine]
R18 [s2] = 0	[00400054] 34020004 ori \$2, \$0, 4 ; 26: li \$v0, 4
R19 [s3] = 0	[00400058] 0000000c syscall ; 27: syscall
R20 [s4] = 0	[0040005c] 3c011001 lui \$1, 4097 [arr] ; 29: la \$a0, arr # \$a0 = address of arr
R21 [s5] = 0	[00400060] 3424004c ori \$4, \$1, 76 [arr]
R22 [s6] = 0	[00400064] 34050000 ori \$5, \$0, 0 ; 30: li \$a1, 0 # Low index of the array which is 0
R23 [s7] = 0	[00400068] 34020004 ori \$6, \$0, 13 ; 31: li \$a2, 13 # High index of the array which is 13
R24 [t8] = 0	[00400070] 0c100070 jal 0x00400170 [QUICKSORT] ; 32:jal QSORT # Sort the array by Quicksort
R25 [t9] = 0	[00400070] 3c011001 lui \$1, 4097 [printSortedArray] ; 34: la \$a0, printSortedArray
R26 [k0] = 0	[00400074] 34240035 ori \$4, \$1, 53 [printSortedArray]
R27 [k1] = 0	[00400078] 34020004 ori \$2, \$0, 4 ; 35: li \$v0, 4
R28 [gp] = 10000000	[0040007c] 0000000c syscall ; 36: syscall
R29 [sp] = 7ffff19c	[00400080] 3c011001 lui \$1, 4097 [arr] ; 38: la \$a1, arr # \$a1 = address of arr
R30 [s8] = 0	[00400084] 3425004c ori \$5, \$1, 76 [arr]
R31 [ra] = 400070	[00400088] 0c100072 jal 0x004001c8 [DISPLAY] ; 39: jal DISPLAY # Print the array
	[0040008c] 3c011001 lui \$1, 4097 [strLine] ; 41: la \$a0, strLine

Int Regs [16]	Text
R2 [v0] = 4	[004000fc] 00069021 addi \$18, \$0, \$6 ; 79: move \$s2, \$a2 # \$s2 = \$a2
R3 [v1] = 0	[00400100] 00124880 sll \$1, \$18, 2 ; 81: sll \$t1, \$s2, 2 # t1 = high * 4
R4 [a0] = 1001004c	[00400104] 00894820 add \$9, \$4, \$9 ; 82: add \$t1, \$a0, \$t1 # t1 = arr + high * 4
R5 [a1] = 0	[00400108] 8d2a0000 lw \$10, 0(\$9) ; 83: lw \$t2, 0(\$t1) # pivot = t2 = arr[high]
R6 [a2] = d	[0040010c] 222bffff addi \$11, \$17, -1 ; 85: addi \$t3, \$s1, -1 # t3 = i = low - 1
R7 [a3] = 0	[00400110] 00116021 addu \$12, \$0, \$17 ; 86: move \$t4, \$s1 # t4 = j = low
R8 [t0] = 0	[00400114] 224dffff addi \$13, \$18, -1 ; 87: addi \$t5, \$s2, -1 # t5 = high - 1
R9 [t1] = 0	[00400118] 01ac702a slt \$14, \$13, \$12 ; 90: slt \$t6, \$t5, \$t4 # Check \$t6 = 1 if j > high - 1
R10 [t2] = 0	[0040011c] 15c0000e bne \$14, \$0, 56 [endifor-0x0040011c]
R11 [t3] = 0	[00400120] 000c4880 sll \$9, \$12, 2 ; 94: sll \$t1, \$t4, 2 # \$t1 = low * 4
R12 [t4] = 0	[00400124] 01244820 add \$9, \$9, \$4 ; 95: add \$t1, \$t1, \$a0 # \$t1 = arr + low * 4
R13 [t5] = 0	[00400128] 8d2f0000 lw \$10, 0(\$9) ; 96: lw \$t7, 0(\$t1) # \$t7 = arr[j] = arr[low]
R14 [t6] = 0	[0040012c] 01eac02a slt \$24, \$15, \$10 ; 98: slt \$t8, \$t7, \$t2 # Check \$t8 = 1 if arr[j] > pivot
R15 [t7] = 0	[00400130] 17000007 bne \$24, \$0, 28 [endifif-0x00400130]
R16 [s0] = 0	[00400134] 216b0001 addi \$11, \$11, 1 ; 102: addi \$t3, \$t3, 1 # Increase i++
R17 [s1] = 0	[00400138] 000b2821 addu \$5, \$0, \$11 ; 103: move \$a1, \$t3
R18 [s2] = 0	[0040013c] 000c3021 addu \$6, \$0, \$12 ; 104: move \$a2, \$t4
R19 [s3] = 0	[00400140] 0c100029 jal 0x00400044 [SWAP] ; 105: jal SWAP
R20 [s4] = 0	[00400144] 218c0001 addi \$12, \$12, 1 ; 107: addi \$t4, \$t4, 1 # j++
R21 [s5] = 0	[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
R22 [s6] = 0	[0040014c] 218c0001 addi \$12, \$12, 1 ; 111: addi \$t4, \$t4, 1 # j++
R23 [s7] = 0	[00400150] 08100046 j 0x00400118 [forloop] ; 112: j forloop
R24 [t8] = 0	[00400154] 21650001 addi \$5, \$11, 1 ; 115: addi \$a1, \$t3, 1 # \$a1 = i + 1
R25 [t9] = 0	[00400158] 00423021 addu \$6, \$0, \$18 ; 116: move \$a2, \$a2 # \$a2 = high
R26 [k0] = 0	[0040015c] 00051020 add \$2, \$0, \$5 ; 117: add \$v0, \$zero, \$a1 # \$v0 = \$a1
R27 [k1] = 0	[00400160] 0c100029 jal 0x00400044 [SWAP] ; 118: jal SWAP # Swap(arr[i+1], arr[high])
R28 [gp] = 10000000	[00400164] 8fbff000c lw \$31, 12(\$29) ; 120: la \$ra, 12(\$sp) # Return address
R29 [sp] = 7ffff19c	[00400168] 23bdf010 addi \$29, \$29, 16 ; 121: addi \$sp, \$sp, 16 # Restore the stack
R30 [s8] = 0	[0040016c] 03e00008 jr \$31 ; 122: jr \$ra
R31 [ra] = 400070	[00400170] 23bdf00 addi \$29, \$29, -16 # Allocate space on the stack
	[00400174] afa40000 sw \$4, 0(\$29) ; 127: sw \$a0, 0(\$sp) # save arr

Int Regs [16]		Text
PC	= 400188	[00400110] 00116021 addu \$12, \$0, \$17 ; 86: move \$t4, \$a1 # 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 [endifor-0x0400011c] [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, \$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 [endifif-0x0400130] [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 0x040000a4 [SWAP] ; 105: jal SWAP [00400144] 218c0001 addi \$12, \$12, 1 ; 107: addi \$t4, \$t4, 1 # j++ [00400148] 08100046 j 0x0400118 [forloop] ; 108: j forloop [0040014c] 218c0001 addi \$12, \$12, 1 ; 111: addi \$t4, \$t4, 1 # j++ [00400150] 08100046 j 0x0400118 [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 0x040000a4 [SWAP] ; 118: jal SWAP # Swap(arr[i+1], arr[high]) [00400164] 8fb0000c lw \$31, 12(\$29) ; 120: lw \$r4, 12(\$sp) # Return address [00400168] 23bd0010 addi \$29, \$29, 16 ; 121: addi \$sp, \$sp, 16 # Restore the stack [0040016c] 03e00008 jr \$31 ; 122: jr \$r4 [00400170] 23bdffff 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 \$r4, 12(\$sp) # Return address [00400184] 00064021 addu \$8, \$0, \$6 ; 132: move \$t0, \$a2 # \$t0 = high [00400188] 00a8482a sll \$9, \$5, \$8 ; 134: sll \$t1, \$a1, \$t0 # Check \$t1 = 1 if low > high 

Int Regs [16]		Text
PC	= 40018c	[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 [endifor-0x0400011c] [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, \$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 [endifif-0x0400130] [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 0x040000a4 [SWAP] ; 105: jal SWAP [00400144] 218c0001 addi \$12, \$12, 1 ; 107: addi \$t4, \$t4, 1 # j++ [00400148] 08100046 j 0x0400118 [forloop] ; 108: j forloop [0040014c] 218c0001 addi \$12, \$12, 1 ; 111: addi \$t4, \$t4, 1 # j++ [00400150] 08100046 j 0x0400118 [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 0x040000a4 [SWAP] ; 118: jal SWAP # Swap(arr[i+1], arr[high]) [00400164] 8fb0000c lw \$31, 12(\$29) ; 120: lw \$r4, 12(\$sp) # Return address [00400168] 23bd0010 addi \$29, \$29, 16 ; 121: addi \$sp, \$sp, 16 # Restore the stack [0040016c] 03e00008 jr \$31 ; 122: jr \$r4 [00400170] 23bdffff 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 \$r4, 12(\$sp) # Return address [00400184] 00064021 addu \$8, \$0, \$6 ; 132: move \$t0, \$a2 # \$t0 = high [00400188] 00a8482a sll \$9, \$5, \$8 ; 134: sll \$t1, \$a1, \$t0 # Check \$t1 = 1 if low > high [0040018c] 11200009 beq \$9, \$0, 36 [ENDIF-0x040018c]

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] = 7fffff18c		
R30	[s8] = 0		
R31	[ra] = 400194		
PC	= 400194		
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] = 7fffff18c		
R30	[s8] = 0		
R31	[ra] = 400194		
PC	= 400194		
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] = 7fffff18c		
R30	[s8] = 0		
R31	[ra] = 400194		
PC	= 400194		
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] = 7fffff18c		
R30	[s8] = 0		
R31	[ra] = 400194		
PC	= 400194		
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] = 7fffff18c		
R30	[s8] = 0		
R31	[ra] = 400194		
PC	= 400194		
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] = 7fffff18c		
R30	[s8] = 0		
R31	[ra] = 400194		
PC	= 400194		
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] = 7fffff18c		
R30	[s8] = 0		
R31	[ra] = 400194		
PC	= 400194		
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	</td	

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] = 7fffff17c  
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] 23bdffff 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 addi $17, $0, $5 ; 78: move $s1, $a1 # $s1 = $a1
[004000fc] 00069021 addi $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 addi $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 sll $14, $13, $12 ; 90: sll $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
[0040011c] 15c0000e bne $14, $0, $56 [endifor-0x0040011c]
[00400120] 000c4880 sll $9, $12, 2 ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400124] 01244820 add $9, $4, $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 sll $24, $15, $10 ; 98: sll $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400130] 17000007 bne $24, $0, $28 [endifif-0x00400130]
[00400134] 216b0001 addi $11, $1, 1 ; 102: addi $t3, $t3, 1 # Increase i++
[00400138] 00058281 add $9, $0, $11 ; 103: move $a1, $t3
[0040013c] 000c3021 add $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] = 7fffff17c  
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] 23bdffff 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 sll $14, $13, $12 ; 90: sll $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
[0040011c] 15c0000e bne $14, $0, $56 [endifor-0x0040011c]
[00400120] 000c4880 sll $9, $12, 2 ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400124] 01244820 add $9, $4, $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 sll $24, $15, $10 ; 98: sll $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400130] 17000007 bne $24, $0, $28 [endifif-0x00400130]
[00400134] 216b0001 addi $11, $1, 1 ; 102: addi $t3, $t3, 1 # Increase i++
[00400138] 00058281 add $9, $0, $11 ; 103: move $a1, $t3
[0040013c] 000c3021 add $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]

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] = 7fffff17c  
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] 23bdffff 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 sll $14, $13, $12 ; 90: sll $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
[0040011c] 15c0000e bne $14, $0, $56 [endifor-0x0040011c]
[00400120] 000c4880 sll $9, $12, 2 ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400124] 01244820 add $9, $4, $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 sll $24, $15, $10 ; 98: sll $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400130] 17000007 bne $24, $0, $28 [endifif-0x00400130]
[00400134] 216b0001 addi $11, $1, 1 ; 102: addi $t3, $t3, 1 # Increase i++
[00400138] 00058281 add $9, $0, $11 ; 103: move $a1, $t3
[0040013c] 000c3021 add $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]		Text
R2	[v0] = 4	[004000dc] 23bd0010 addi \$29, \$29, 16 ; 68: addi \$sp, \$sp, 16
R3	[v1] = 0	[004000e0] 03e00008 jr \$31 ; 69: jr \$ra
R4	[a0] = 1001004c	[004000e4] 23bdffff addi \$29, \$29, -16 ; 71: addi \$sp, \$sp, -16 # Allocate space on the stack
R5	[a1] = 0	[004000e8] afa40000 sw \$4, 0(\$29) ; 73: sw \$a0, 0(\$sp) # Save array on the stack
R6	[a2] = d	[004000ec] afa50004 sw \$5, 4(\$29) ; 74: sw \$a1, 4(\$sp) # Save low on the stack
R7	[a3] = 0	[004000f0] afa60008 sw \$6, 8(\$29) ; 75: sw \$a2, 8(\$sp) # Save high on the stack
R8	[t0] = d	[004000f4] afbf000c sw \$31, 12(\$29) ; 76: sw \$ra, 12(\$sp) # Save \$ra on the stack
R9	[t1] = 34	[004000f8] 00058821 addu \$17, \$0, \$5 ; 78: move \$s1, \$a1 # \$s1 = \$a1
R10	[t2] = 0	[004000fc] 00069021 addu \$18, \$0, \$6 ; 79: move \$s2, \$a2 # \$s2 = \$a2
R11	[t3] = 0	[00400100] 00124880 sll \$9, \$18, 2 ; 81: sll \$t1, \$s2, 2 # t1 = high * 4
R12	[t4] = 0	[00400104] 00894820 add \$9, \$4, \$9 ; 82: add \$t1, \$a0, \$t1 # t1 = arr + high * 4
R13	[t5] = 0	[00400108] 8d2a0000 lw \$10, 0(\$9) ; 83: lw \$t2, 0(\$t1) # pivot = t2 = arr[high]
R14	[t6] = 0	[0040010c] 222bffff addi \$11, \$17, -1 ; 85: addi \$t3, \$s1, -1 # t3 = i = low - 1
R15	[t7] = 0	[00400110] 00116021 addu \$12, \$0, \$17 ; 86: move \$t4, \$s1 # t4 = j = low
R16	[s0] = 0	[00400114] 224dffff addi \$13, \$18, -1 ; 87: addi \$t5, \$s2, -1 # t5 = high - 1
R17	[s1] = 0	[00400118] 01ac702a sll \$14, \$13, \$12 ; 90: sll \$t6, \$t5, \$t4 # Check \$t6 = 1 if j > high - 1
R18	[s2] = d	[0040011c] 15c0000e bne \$14, \$0, 56 [endifor-0x040011c]
R19	[s3] = 0	[00400120] 000e4880 sll \$9, \$12, 2 ; 94: sll \$t1, \$t4, 2 # \$t1 = low * 4
R20	[s4] = 0	[00400124] 01244880 add \$9, \$9, \$4 ; 95: add \$t1, \$t1, \$a0 # \$t1 = arr + low * 4
R21	[s5] = 0	[00400128] 8d2f0000 lw \$15, 0(\$9) ; 96: lw \$t7, 0(\$t1) # \$t7 = arr[j] = arr[low]
R22	[s6] = 0	[0040012c] 01eac02a sll \$24, \$15, \$10 ; 98: sll \$t8, \$t7, \$t2 # Check \$t8 = 1 if arr[j] > pivot
R23	[s7] = 0	[00400130] 17000007 bne \$24, \$0, 28 [endifif-0x0400130]
R24	[t8] = 0	[00400134] 216b0001 addi \$11, \$11, 1 ; 102: addi \$t3, \$t3, 1 # Increase i++
R25	[t9] = 0	[00400138] 000b2821 addu \$5, \$0, \$11 ; 103: move \$a1, \$t3
R26	[k0] = 0	[0040013c] 000c3021 addu \$6, \$0, \$12 ; 104: move \$a2, \$t4
R27	[k1] = 0	[00400140] 0c100029 jal 0x04000a4 [SWAP] ; 105: jal SWAP
R28	[gp] = 10008000	[00400144] 218c0001 addi \$12, \$12, 1 ; 107: addi \$t4, \$t4, 1 # j++
R29	[sp] = 7fffff17c	[00400148] 08100046 j 0x04000118 [forloop] ; 108: j forloop
R30	[s8] = 0	[0040014c] 218c0001 addi \$12, \$12, 1 ; 111: addi \$t4, \$t4, 1 # j++
R31	[ra] = 400194	[00400150] 08100046 j 0x04000118 [forloop] ; 112: j forloop
		[00400154] 21650001 addi \$5, \$11, 1 ; 115: addi \$a1, \$t3, 1 # \$a1 = i + 1

Int Regs [16]		Text
R2	[v0] = 4	[004000dc] 23bd0010 addi \$29, \$29, 16 ; 68: addi \$sp, \$sp, 16
R3	[v1] = 0	[004000e0] 03e00008 jr \$31 ; 69: jr \$ra
R4	[a0] = 1001004c	[004000e4] 23bdffff addi \$29, \$29, -16 ; 71: addi \$sp, \$sp, -16 # Allocate space on the stack
R5	[a1] = 0	[004000e8] afa40000 sw \$4, 0(\$29) ; 73: sw \$a0, 0(\$sp) # Save array on the stack
R6	[a2] = d	[004000ec] afa50004 sw \$5, 4(\$29) ; 74: sw \$a1, 4(\$sp) # Save low on the stack
R7	[a3] = 0	[004000f0] afa60008 sw \$6, 8(\$29) ; 75: sw \$a2, 8(\$sp) # Save high on the stack
R8	[t0] = d	[004000f4] afbf000c sw \$31, 12(\$29) ; 76: sw \$ra, 12(\$sp) # Save \$ra on the stack
R9	[t1] = 10010090	[004000f8] 00058821 addu \$17, \$0, \$5 ; 78: move \$s1, \$a1 # \$s1 = \$a1
R10	[t2] = 0	[004000fc] 00069021 addu \$18, \$0, \$6 ; 79: move \$s2, \$a2 # \$s2 = \$a2
R11	[t3] = 0	[00400100] 00124880 sll \$9, \$18, 2 ; 81: sll \$t1, \$s2, 2 # t1 = high * 4
R12	[t4] = 0	[00400104] 00894820 add \$9, \$4, \$9 ; 82: add \$t1, \$a0, \$t1 # t1 = arr + high * 4
R13	[t5] = 0	[00400108] 8d2a0000 lw \$10, 0(\$9) ; 83: lw \$t2, 0(\$t1) # pivot = t2 = arr[high]
R14	[t6] = 0	[0040010c] 222bffff addi \$11, \$17, -1 ; 85: addi \$t3, \$s1, -1 # t3 = i = low - 1
R15	[t7] = 0	[00400110] 00116021 addu \$12, \$0, \$17 ; 86: move \$t4, \$s1 # t4 = j = low
R16	[s0] = 0	[00400114] 224dffff addi \$13, \$18, -1 ; 87: addi \$t5, \$s2, -1 # t5 = high - 1
R17	[s1] = 0	[00400118] 01ac702a sll \$14, \$13, \$12 ; 90: sll \$t6, \$t5, \$t4 # Check \$t6 = 1 if j > high - 1
R18	[s2] = d	[0040011c] 15c0000e bne \$14, \$0, 56 [endifor-0x040011c]
R19	[s3] = 0	[00400120] 000e4880 sll \$9, \$12, 2 ; 94: sll \$t1, \$t4, 2 # \$t1 = low * 4
R20	[s4] = 0	[00400124] 01244880 add \$9, \$9, \$4 ; 95: add \$t1, \$t1, \$a0 # \$t1 = arr + low * 4
R21	[s5] = 0	[00400128] 8d2f0000 lw \$15, 0(\$9) ; 96: lw \$t7, 0(\$t1) # \$t7 = arr[j] = arr[low]
R22	[s6] = 0	[0040012c] 01eac02a sll \$24, \$15, \$10 ; 98: sll \$t8, \$t7, \$t2 # Check \$t8 = 1 if arr[j] > pivot
R23	[s7] = 0	[00400130] 17000007 bne \$24, \$0, 28 [endifif-0x0400130]
R24	[t8] = 0	[00400134] 216b0001 addi \$11, \$11, 1 ; 102: addi \$t3, \$t3, 1 # Increase i++
R25	[t9] = 0	[00400138] 000b2821 addu \$5, \$0, \$11 ; 103: move \$a1, \$t3
R26	[k0] = 0	[0040013c] 000c3021 addu \$6, \$0, \$12 ; 104: move \$a2, \$t4
R27	[k1] = 0	[00400140] 0c100029 jal 0x04000a4 [SWAP] ; 105: jal SWAP
R28	[gp] = 10008000	[00400144] 218c0001 addi \$12, \$12, 1 ; 107: addi \$t4, \$t4, 1 # j++
R29	[sp] = 7fffff17c	[00400148] 08100046 j 0x04000118 [forloop] ; 108: j forloop
R30	[s8] = 0	[0040014c] 218c0001 addi \$12, \$12, 1 ; 111: addi \$t4, \$t4, 1 # j++
R31	[ra] = 400194	[00400150] 08100046 j 0x04000118 [forloop] ; 112: j forloop
		[00400154] 21650001 addi \$5, \$11, 1 ; 115: addi \$a1, \$t3, 1 # \$a1 = i + 1

Int Regs [16]		Text
R2	[v0] = 4	[004000dc] 23bd0010 addi \$29, \$29, 16 ; 68: addi \$sp, \$sp, 16
R3	[v1] = 0	[004000e0] 03e00008 jr \$31 ; 69: jr \$ra
R4	[a0] = 1001004c	[004000e4] 23bdffff addi \$29, \$29, -16 ; 71: addi \$sp, \$sp, -16 # Allocate space on the stack
R5	[a1] = 0	[004000e8] afa40000 sw \$4, 0(\$29) ; 73: sw \$a0, 0(\$sp) # Save array on the stack
R6	[a2] = d	[004000ec] afa50004 sw \$5, 4(\$29) ; 74: sw \$a1, 4(\$sp) # Save low on the stack
R7	[a3] = 0	[004000f0] afa60008 sw \$6, 8(\$29) ; 75: sw \$a2, 8(\$sp) # Save high on the stack
R8	[t0] = d	[004000f4] afbf000c sw \$31, 12(\$29) ; 76: sw \$ra, 12(\$sp) # Save \$ra on the stack
R9	[t1] = 10010080	[004000f8] 00058821 addu \$17, \$0, \$5 ; 78: move \$s1, \$a1 # \$s1 = \$a1
R10	[t2] = 0	[004000fc] 00069021 addu \$18, \$0, \$6 ; 79: move \$s2, \$a2 # \$s2 = \$a2
R11	[t3] = 0	[00400100] 00124880 sll \$9, \$18, 2 ; 81: sll \$t1, \$s2, 2 # t1 = high * 4
R12	[t4] = 0	[00400104] 00894820 add \$9, \$4, \$9 ; 82: add \$t1, \$a0, \$t1 # t1 = arr + high * 4
R13	[t5] = 0	[00400108] 8d2a0000 lw \$10, 0(\$9) ; 83: lw \$t2, 0(\$t1) # pivot = t2 = arr[high]
R14	[t6] = 0	[0040010c] 222bffff addi \$11, \$17, -1 ; 85: addi \$t3, \$s1, -1 # t3 = i = low - 1
R15	[t7] = 0	[00400110] 00116021 addu \$12, \$0, \$17 ; 86: move \$t4, \$s1 # t4 = j = low
R16	[s0] = 0	[00400114] 224dffff addi \$13, \$18, -1 ; 87: addi \$t5, \$s2, -1 # t5 = high - 1
R17	[s1] = 0	[00400118] 01ac702a sll \$14, \$13, \$12 ; 90: sll \$t6, \$t5, \$t4 # Check \$t6 = 1 if j > high - 1
R18	[s2] = d	[0040011c] 15c0000e bne \$14, \$0, 56 [endifor-0x040011c]
R19	[s3] = 0	[00400120] 000e4880 sll \$9, \$12, 2 ; 94: sll \$t1, \$t4, 2 # \$t1 = low * 4
R20	[s4] = 0	[00400124] 01244880 add \$9, \$9, \$4 ; 95: add \$t1, \$t1, \$a0 # \$t1 = arr + low * 4
R21	[s5] = 0	[00400128] 8d2f0000 lw \$15, 0(\$9) ; 96: lw \$t7, 0(\$t1) # \$t7 = arr[j] = arr[low]
R22	[s6] = 0	[0040012c] 01eac02a sll \$24, \$15, \$10 ; 98: sll \$t8, \$t7, \$t2 # Check \$t8 = 1 if arr[j] > pivot
R23	[s7] = 0	[00400130] 17000007 bne \$24, \$0, 28 [endifif-0x0400130]
R24	[t8] = 0	[00400134] 216b0001 addi \$11, \$11, 1 ; 102: addi \$t3, \$t3, 1 # Increase i++
R25	[t9] = 0	[00400138] 000b2821 addu \$5, \$0, \$11 ; 103: move \$a1, \$t3
R26	[k0] = 0	[0040013c] 000c3021 addu \$6, \$0, \$12 ; 104: move \$a2, \$t4
R27	[k1] = 0	[00400140] 0c100029 jal 0x04000a4 [SWAP] ; 105: jal SWAP
R28	[gp] = 10008000	[00400144] 218c0001 addi \$12, \$12, 1 ; 107: addi \$t4, \$t4, 1 # j++
R29	[sp] = 7fffff17c	[00400148] 08100046 j 0x04000118 [forloop] ; 108: j forloop
R30	[s8] = 0	[0040014c] 218c0001 addi \$12, \$12, 1 ; 111: addi \$t4, \$t4, 1 # j++
R31	[ra] = 400194	[00400150] 08100046 j 0x04000118 [forloop] ; 112: j forloop
		[00400154] 21650001 addi \$5, \$11, 1 ; 115: addi \$a1, \$t3, 1 # \$a1 = i + 1

Int Regs [16]		Text
R2	[v0] = 4	[004000dc] 23bd0010 addi \$29, \$29, 16 ; 68: addi \$sp, \$sp, 16
R3	[v1] = 0	[004000e0] 03e00008 jr \$31 ; 69: jr \$ra
R4	[a0] = 1001004c	[004000e4] 23bdffff addi \$29, \$29, -16 ; 71: addi \$sp, \$sp, -16 # Allocate space on the stack
R5	[a1] = 0	[004000e8] afa40000 sw \$4, 0(\$29) ; 73: sw \$a0, 0(\$sp) # Save array on the stack
R6	[a2] = d	[004000ec] afa50004 sw \$5, 4(\$29) ; 74: sw \$a1, 4(\$sp) # Save low on the stack
R7	[a3] = 0	[004000f0] afa60008 sw \$6, 8(\$29) ; 75: sw \$a2, 8(\$sp) # Save high on the stack
R8	[t0] = d	[004000f4] afbf000c sw \$31, 12(\$29) ; 76: sw \$ra, 12(\$sp) # Save \$ra on the stack
R9	[t1] = 10010080	[004000f8] 00058821 addi \$17, \$0, \$5 ; 78: move \$s1, \$a1 # \$s1 = \$a1
R10	[t2] = 0	[004000fc] 00069021 addi \$18, \$0, \$6 ; 79: move \$s2, \$a2 # \$s2 = \$a2
R11	[t3] = ffffffff	[00400100] 00124880 sll \$9, \$18, 2 ; 81: sll \$t1, \$s2, 2 # t1 = high * 4
R12	[t4] = 0	[00400104] 00894820 add \$9, \$4, \$9 ; 82: add \$t1, \$a0, \$t1 # t1 = arr + high * 4
R13	[t5] = c	[00400108] 8d2a0000 lw \$10, 0(\$9) ; 83: lw \$t2, 0(\$t1) # pivot = t2 = arr[high]
R14	[t6] = 0	[0040010c] 222bffff addi \$11, \$17, -1 ; 85: addi \$t3, \$s1, -1 # t3 = i = low - 1
R15	[t7] = 0	[00400110] 00116021 addi \$12, \$0, \$17 ; 86: move \$t4, \$s1 # t4 = j = low
R16	[s0] = 0	[00400114] 224dffff addi \$13, \$18, -1 ; 87: addi \$t5, \$s2, -1 # t5 = high - 1
R17	[s1] = 0	[00400118] 01ac702a sll \$14, \$13, \$12 ; 89: sll \$t6, \$t5, \$t4 # Check \$t6 = 1 if j > high - 1
R18	[s2] = d	[0040011c] 15c0000e bne \$14, \$0, 56 [endifor-0x0040011c] ; 102: addi \$t3, \$t3, 1 # Increase i++
R19	[s3] = 0	[00400120] 000c4880 sll \$9, \$12, 2 ; 94: sll \$t1, \$t4, 2 # t1 = low * 4
R20	[s4] = 0	[00400124] 01244820 add \$9, \$4, \$4 ; 95: add \$t1, \$s1, \$a0 # \$t1 = arr + low * 4
R21	[s5] = 0	[00400128] 8d2f0000 lw \$15, 0(\$9) ; 96: lw \$t2, 0(\$t1) # \$t2 = arr[j] = arr[low]
R22	[s6] = 0	[0040012c] 01eac02a sll \$24, \$15, \$10 ; 98: sll \$t3, \$t7, \$t2 # Check \$t8 = 1 if arr[j] > pivot
R23	[s7] = 0	[00400130] 17000007 bne \$24, \$0, 28 [endifor-0x00400130] ; 102: addi \$t4, \$t4, 1 # Increase i++
R24	[t8] = 0	[00400134] 216b0001 addi \$11, \$11, 1 ; 103: move \$a1, \$t3
R25	[t9] = 0	[00400138] 000b2821 addu \$5, \$0, \$11 ; 104: move \$a2, \$t4
R26	[k0] = 0	[0040013c] 000c3021 addu \$6, \$0, \$12 ; 105: jal SWAP
R27	[k1] = 0	[00400140] 01c00029 jal 0x004000a4 [SWAP] ; 108: j forloop
R28	[gp] = 10008000	[00400144] 218c0001 addi \$12, \$12, 1 ; 109: addi \$t4, \$t4, 1 # j++
R29	[sp] = 7fffff17c	[00400148] 08100046 j 0x00400118 [forloop] ; 111: addi \$t4, \$t4, 1 # j++
R30	[s8] = 0	[0040014c] 218c0001 addi \$12, \$12, 1 ; 112: j forloop
R31	[ra] = 400194	[00400150] 08100046 j 0x00400118 [forloop] ; 113: addi \$a1, \$t3, 1 # \$a1 = i + 1
		[00400154] 21650001 addi \$5, \$11, 1 ; 115: addi \$a1, \$t3, 1 # \$a1 = i + 1

Int Regs [16]		Text
R2	[v0] = 4	[004000dc] 23bd0010 addi \$29, \$29, 16 ; 68: addi \$sp, \$sp, 16
R3	[v1] = 0	[004000e0] 03e00008 jr \$31 ; 69: jr \$ra
R4	[a0] = 1001004c	[004000e4] 23bdffff addi \$29, \$29, -16 ; 71: addi \$sp, \$sp, -16 # Allocate space on the stack
R5	[a1] = 0	[004000e8] afa40000 sw \$4, 0(\$29) ; 73: sw \$a0, 0(\$sp) # Save array on the stack
R6	[a2] = d	[004000ec] afa50004 sw \$5, 4(\$29) ; 74: sw \$a1, 4(\$sp) # Save low on the stack
R7	[a3] = 0	[004000f0] afa60008 sw \$6, 8(\$29) ; 75: sw \$a2, 8(\$sp) # Save high on the stack
R8	[t0] = d	[004000f4] afbf000c sw \$31, 12(\$29) ; 76: sw \$ra, 12(\$sp) # Save \$ra on the stack
R9	[t1] = 0	[004000f8] 00058821 addu \$17, \$0, \$5 ; 78: move \$s1, \$a1 # \$s1 = \$a1
R10	[t2] = 0	[004000fc] 00069021 addu \$18, \$0, \$6 ; 79: move \$s2, \$a2 # \$s2 = \$a2
R11	[t3] = ffffffff	[00400100] 00124880 sll \$9, \$18, 2 ; 81: sll \$t1, \$s2, 2 # t1 = high * 4
R12	[t4] = 0	[00400104] 00894820 add \$9, \$4, \$9 ; 82: add \$t1, \$a0, \$t1 # t1 = arr + high * 4
R13	[t5] = c	[00400108] 8d2a0000 lw \$10, 0(\$9) ; 83: lw \$t2, 0(\$t1) # pivot = t2 = arr[high]
R14	[t6] = 0	[0040010c] 222bffff addi \$11, \$17, -1 ; 85: addi \$t3, \$s1, -1 # t3 = i = low - 1
R15	[t7] = 0	[00400110] 00116021 addu \$12, \$0, \$17 ; 86: move \$t4, \$s1 # t4 = j = low
R16	[s0] = 0	[00400114] 224dffff addi \$13, \$18, -1 ; 87: addi \$t5, \$s2, -1 # t5 = high - 1
R17	[s1] = 0	[00400118] 01ac702a sll \$14, \$13, \$12 ; 89: sll \$t6, \$t5, \$t4 # Check \$t6 = 1 if j > high - 1
R18	[s2] = d	[0040011c] 15c0000e bne \$14, \$0, 56 [endifor-0x0040011c] ; 90: sll \$t6, \$t5, \$t4 # Check \$t6 = 1 if j > high - 1
R19	[s3] = 0	[00400120] 000e4880 sll \$9, \$12, 2 ; 94: sll \$t1, \$t4, 2 # t1 = low * 4
R20	[s4] = 0	[00400124] 01244820 add \$9, \$4, \$4 ; 95: add \$t1, \$t1, \$a0 # \$t1 = arr + low * 4
R21	[s5] = 0	[00400128] 8d2f0000 lw \$15, 0(\$9) ; 96: lw \$t2, 0(\$t1) # \$t2 = arr[j] = arr[low]
R22	[s6] = 0	[0040012c] 01eac02a sll \$24, \$15, \$10 ; 98: sll \$t3, \$t7, \$t2 # Check \$t8 = 1 if arr[j] > pivot
R23	[s7] = 0	[00400130] 17000007 bne \$24, \$0, 28 [endifor-0x00400130] ; 102: addi \$t4, \$t4, 1 # Increase i++
R24	[t8] = 0	[00400134] 216b0001 addi \$11, \$11, 1 ; 103: move \$a1, \$t3
R25	[t9] = 0	[00400138] 000b2821 addu \$6, \$0, \$11 ; 104: move \$a2, \$t4
R26	[k0] = 0	[0040013c] 000c3021 addu \$6, \$0, \$12 ; 105: jal SWAP
R27	[k1] = 0	[00400140] 01c00029 jal 0x004000a4 [SWAP] ; 108: j forloop
R28	[gp] = 10008000	[00400144] 218c0001 addi \$12, \$12, 1 ; 109: addi \$t4, \$t4, 1 # j++
R29	[sp] = 7fffff17c	[00400148] 08100046 j 0x00400118 [forloop] ; 111: addi \$t4, \$t4, 1 # j++
R30	[s8] = 0	[0040014c] 218c0001 addi \$12, \$12, 1 ; 112: j forloop
R31	[ra] = 400194	[00400150] 08100046 j 0x00400118 [forloop] ; 113: addi \$a1, \$t3, 1 # \$a1 = i + 1
		[00400154] 21650001 addi \$5, \$11, 1 ; 115: addi \$a1, \$t3, 1 # \$a1 = i + 1

Int Regs [16]		Text
R2	[v0] = 4	[004000dc] 23bd0010 addi \$29, \$29, 16 ; 68: addi \$sp, \$sp, 16
R3	[v1] = 0	[004000e0] 03e00008 jr \$31 ; 69: jr \$ra
R4	[a0] = 1001004c	[004000e4] 23bdffff addi \$29, \$29, -16 ; 71: addi \$sp, \$sp, -16 # Allocate space on the stack
R5	[a1] = 0	[004000e8] afa40000 sw \$4, 0(\$29) ; 73: sw \$a0, 0(\$sp) # Save array on the stack
R6	[a2] = d	[004000ec] afa50004 sw \$5, 4(\$29) ; 74: sw \$a1, 4(\$sp) # Save low on the stack
R7	[a3] = 0	[004000f0] afa60008 sw \$6, 8(\$29) ; 75: sw \$a2, 8(\$sp) # Save high on the stack
R8	[t0] = d	[004000f4] afbf000c sw \$31, 12(\$29) ; 76: sw \$ra, 12(\$sp) # Save \$ra on the stack
R9	[t1] = 1001004c	[004000f8] 00058821 addu \$17, \$0, \$5 ; 78: move \$s1, \$a1 # \$s1 = \$a1
R10	[t2] = 0	[004000fc] 00069021 addu \$18, \$0, \$6 ; 79: move \$s2, \$a2 # \$s2 = \$a2
R11	[t3] = ffffffff	[00400100] 00124880 sll \$9, \$18, 2 ; 81: sll \$t1, \$s2, 2 # t1 = high * 4
R12	[t4] = 0	[00400104] 00894820 add \$9, \$4, \$9 ; 82: add \$t1, \$a0, \$t1 # t1 = arr + high * 4
R13	[t5] = c	[00400108] 8d2a0000 lw \$10, 0(\$9) ; 83: lw \$t2, 0(\$t1) # pivot = t2 = arr[high]
R14	[t6] = 0	[0040010c] 222bffff addi \$11, \$17, -1 ; 85: addi \$t3, \$s1, -1 # t3 = i = low - 1
R15	[t7] = 0	[00400110] 00116021 addu \$12, \$0, \$17 ; 86: move \$t4, \$s1 # t4 = j = low
R16	[s0] = 0	[00400114] 224dffff addi \$13, \$18, -1 ; 87: addi \$t5, \$s2, -1 # t5 = high - 1
R17	[s1] = 0	[00400118] 01ac702a sll \$14, \$13, \$12 ; 89: sll \$t6, \$t5, \$t4 # Check \$t6 = 1 if j > high - 1
R18	[s2] = d	[0040011c] 15c0000e bne \$14, \$0, 56 [endifor-0x0040011c] ; 90: sll \$t6, \$t5, \$t4 # Check \$t6 = 1 if j > high - 1
R19	[s3] = 0	[00400120] 000e4880 sll \$9, \$12, 2 ; 94: sll \$t1, \$t4, 2 # t1 = low * 4
R20	[s4] = 0	[00400124] 01244820 add \$9, \$4, \$4 ; 95: add \$t1, \$t1, \$a0 # \$t1 = arr + low * 4
R21	[s5] = 0	[00400128] 8d2f0000 lw \$15, 0(\$9) ; 96: lw \$t2, 0(\$t1) # \$t2 = arr[j] = arr[low]
R22	[s6] = 0	[0040012c] 01eac02a sll \$24, \$15, \$10 ; 98: sll \$t3, \$t7, \$t2 # Check \$t8 = 1 if arr[j] > pivot
R23	[s7] = 0	[00400130] 17000007 bne \$24, \$0, 28 [endifor-0x00400130] ; 102: addi \$t4, \$t4, 1 # Increase i++
R24	[t8] = 0	[00400134] 216b0001 addi \$11, \$11, 1 ; 103: move \$a1, \$t3
R25	[t9] = 0	[00400138] 000b2821 addu \$6, \$0, \$11 ; 104: move \$a2, \$t4
R26	[k0] = 0	[0040013c] 000c3021 addu \$6, \$0, \$12 ; 105: jal SWAP
R27	[k1] = 0	[00400140] 01c00029 jal 0x004000a4 [SWAP] ; 108: j forloop
R28	[gp] = 10008000	[00400144] 218c0001 addi \$12, \$12, 1 ; 109: addi \$t4, \$t4, 1 # j++
R29	[sp] = 7fffff17c	[00400148] 08100046 j 0x00400118 [forloop] ; 111: addi \$t4, \$t4, 1 # j++
R30	[s8] = 0	[0040014c] 218c0001 addi \$12, \$12, 1 ; 112: j forloop
R31	[ra] = 400194	[00400150] 08100046 j 0x00400118 [forloop] ; 113: addi \$a1, \$t3, 1 # \$a1 = i + 1
		[00400154] 21650001 addi \$5, \$11, 1 ; 115: addi \$a1, \$t3, 1 # \$a1 = i + 1

Int Regs [16] Text

```

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] = 7fffff17c
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] 23bdffff 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] 00124890 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: sll $t1, $t4, 2 # $t1 = low * 4
[0040011c] 15c0000e bne $14, $0, 56 [endifor-0x0040011c] ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400120] 000c4880 sll $15, $0($9) ; 95: add $t1, $t1, $a0 # $t1 = arr + low * 4
[00400124] 01244820 add $9, $9, $4 ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
[00400128] 8d2f0000 lw $15, 0($9) ; 98: sll $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[0040013c] 01eac02a slt $24, $15, $10 ; 99: sll $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400140] 218c0001 addi $12, $12, 1 ; 102: addi $t3, $t3, 1 # Increase i++
[00400144] 218c0001 addi $12, $12, 1 ; 103: move $s1, $a1 # $s1 = $a1
[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
[00400152] 08100046 j 0x00400118 [forloop] ; 111: addi $t4, $t4, 1 # j++
[00400154] 21650001 addi $5, $11, 1 ; 112: j forloop
[00400158] 21650001 addi $5, $11, 1 ; 115: addi $a1, $t3, 1 # $a1 = i + 1

```

Int Regs [16] Text

```

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] = 7fffff17c
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] 23bdffff 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] 00124890 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: sll $t1, $t4, 2 # $t1 = low * 4
[0040011c] 15c0000e bne $14, $0, 56 [endifor-0x0040011c] ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400120] 000c4880 sll $15, $0($9) ; 95: add $t1, $t1, $a0 # $t1 = arr + low * 4
[00400124] 01244820 add $9, $9, $4 ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
[00400128] 8d2f0000 lw $15, 0($9) ; 98: sll $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[0040013c] 01eac02a slt $24, $15, $10 ; 99: sll $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400140] 218c0001 addi $12, $12, 1 ; 102: addi $t3, $t3, 1 # Increase i++
[00400144] 218c0001 addi $12, $12, 1 ; 103: move $s1, $a1 # $s1 = $a1
[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
[00400152] 08100046 j 0x00400118 [forloop] ; 111: addi $t4, $t4, 1 # j++
[00400154] 21650001 addi $5, $11, 1 ; 112: j forloop
[00400158] 21650001 addi $5, $11, 1 ; 115: addi $a1, $t3, 1 # $a1 = i + 1

```

Int Regs [16] Text

```

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] = 7fffff17c
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] 23bdffff 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] 00124890 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: sll $t1, $t4, 2 # $t1 = low * 4
[0040011c] 15c0000e bne $14, $0, 56 [endifor-0x0040011c] ; 94: sll $t1, $t4, 2 # $t1 = low * 4
[00400120] 000c4880 sll $15, $0($9) ; 95: add $t1, $t1, $a0 # $t1 = arr + low * 4
[00400124] 01244820 add $9, $9, $4 ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
[00400128] 8d2f0000 lw $15, 0($9) ; 98: sll $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[0040013c] 01eac02a slt $24, $15, $10 ; 99: sll $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
[00400140] 218c0001 addi $12, $12, 1 ; 102: addi $t3, $t3, 1 # Increase i++
[00400144] 218c0001 addi $12, $12, 1 ; 103: move $s1, $a1 # $s1 = $a1
[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
[00400152] 08100046 j 0x00400118 [forloop] ; 111: addi $t4, $t4, 1 # j++
[00400154] 21650001 addi $5, $11, 1 ; 112: j forloop
[00400158] 21650001 addi $5, $11, 1 ; 115: addi $a1, $t3, 1 # $a1 = i + 1

```

Int Regs [16]		Text
R2	[v0] = 4	[00400000] 000a5020 add \$v0, \$4, \$v0
R3	[v1] = 0	[004000cc] 8d540000 lw \$20, 0(\$10)
R4	[a0] = 1001004c	[004000d0] ad340000 sw \$20, 0(\$59)
R5	[a1] = 0	[004000d4] ad530000 sw \$19, 0(\$10)
R6	[a2] = 0	[004000d8] 8faa000c lw \$10, 12(\$29)
R7	[a3] = 0	[004000dc] 23bd0010 add \$29, \$29, 16
R8	[t0] = d	[004000e0] 03e00008 jr \$31
R9	[t1] = 1001004c	[004000e4] 23bdffff0 add \$29, \$29, -16
R10	[t2] = 0	[004000e8] afa40000 sw \$4, 0(\$29)
R11	[t3] = 0	[004000ec] afa50004 sw \$5, 4(\$29)
R12	[t4] = 0	[004000f0] afa60008 sw \$6, 8(\$29)
R13	[t5] = c	[004000f4] afbf000c sw \$31, 12(\$29)
R14	[t6] = 0	[004000f8] 00058821 add \$17, \$0, \$5
R15	[t7] = 2b	[004000fc] 00069021 add \$18, \$0, \$6
R16	[s0] = 0	[00400100] 00124880 sll \$19, \$18, 2
R17	[s1] = 0	[00400104] 00894820 add \$9, \$4, \$9
R18	[s2] = d	[00400108] 8d2a0000 lw \$10, 0(\$9)
R19	[s3] = 0	[0040010c] 222bffff add \$11, \$17, -1
R20	[s4] = 0	[00400110] 00116021 add \$12, \$0, \$17
R21	[s5] = 0	[00400114] 224dffff add \$13, \$18, -1
R22	[s6] = 0	[00400118] 01ac702a slt \$14, \$13, \$12
R23	[s7] = 0	[0040011c] 15c0000e bne \$14, \$0, \$5 [endifor-0x0040011c]
R24	[t8] = 0	[00400120] 000c4880 add \$9, \$12, 2
R25	[t9] = 0	[00400124] 01244820 add \$9, \$9, \$4
R26	[k0] = 0	[00400128] 8d2f0000 lw \$15, 0(\$9)
R27	[k1] = 0	[0040012c] 01eac02a slt \$24, \$15, \$10
R28	[gp] = 10008000	[00400130] 216b0001 add \$11, \$11, 1
R29	[sp] = 7fffff17c	[00400134] 216b0001 add \$11, \$11, 1 # Increase i++
R30	[s8] = 0	[00400138] 000c2821 addu \$5, \$0, \$11
R31	[ra] = 400144	[0040013c] 000c3021 addu \$6, \$0, \$12
		<b>[00400140] 01c00029 jal 0x00400044 [SWAP]</b>
		[105: jal SWAP]
		[00400144] 218c0001 addi \$12, \$12, 1
		; 107: addi \$t4, \$t4, 1 # j++

Int Regs [16]		Text
R2	[v0] = 4	[0040009c] 3402000a ori \$2, \$0, 10
R3	[v1] = 0	[004000a0] 00000000 syscall
R4	[a0] = 1001004c	<b>[004000a4] 23bdffff addi \$29, \$29, -16</b> ; 49: addi \$sp, \$sp, -16 # Allocate space on the stack
R5	[a1] = 0	[004000a8] afa40000 sw \$4, 0(\$29)
R6	[a2] = 0	[004000ac] afa50004 sw \$5, 4(\$29)
R7	[a3] = 0	[004000b0] afa60008 sw \$6, 8(\$29)
R8	[t0] = d	[004000b4] afaa000c sw \$10, 12(\$29)
R9	[t1] = 1001004c	[004000b8] 00054880 sll \$9, \$5, 2
R10	[t2] = 0	[004000bc] 00894820 add \$9, \$4, \$9
R11	[t3] = 0	[004000c0] 0d330000 lw \$19, 0(\$9)
R12	[t4] = 0	[004000c4] 00065080 sll \$10, \$6, 2
R13	[t5] = c	[004000c8] 008a5020 add \$10, \$4, \$10
R14	[t6] = 0	[004000cc] 8d540000 lw \$20, 0(\$10)
R15	[t7] = 2b	[004000d0] ad340000 sw \$20, 0(\$9)
R16	[s0] = 0	[004000d4] ad530000 sw \$19, 0(\$10)
R17	[s1] = 0	[004000d8] 8faa000c lw \$10, 12(\$29)
R18	[s2] = d	[004000dc] 23bd0010 addi \$29, \$29, 16
R19	[s3] = 0	[004000e0] 03e00008 jr \$31
R20	[s4] = 0	[004000e4] 23bdffff addi \$29, \$29, -16
R21	[s5] = 0	[004000e8] afa40000 sw \$4, 0(\$29)
R22	[s6] = 0	[004000ec] afa50004 sw \$5, 4(\$29)
R23	[s7] = 0	[004000f0] afa60008 sw \$6, 8(\$29)
R24	[t8] = 0	[004000f4] afbf000c sw \$31, 12(\$29)
R25	[t9] = 0	[004000f8] 00058821 add \$17, \$0, \$5
R26	[k0] = 0	[00400102] 00069021 addu \$18, \$0, \$6
R27	[k1] = 0	[00400106] 00124880 add \$9, \$18, 2
R28	[gp] = 10008000	[0040010a] 00894820 add \$9, \$4, \$9
R29	[sp] = 7fffff16c	[00400108] 8d2a0000 lw \$10, 0(\$9)
R30	[s8] = 0	[0040010c] 222bffff addi \$11, \$17, -1
R31	[ra] = 400144	[00400110] 00116021 addu \$12, \$0, \$17
		<b>[00400114] 224dffff addi \$13, \$18, -1</b>
		; 69: jr \$ra
		; 71: addi \$sp, \$sp, -16 # Allocate space on the stack
		; 73: sw \$a0, 0(\$sp) # Save array on the stack
		; 74: sw \$a1, 4(\$sp) # Save low on the stack
		; 75: sw \$a2, 8(\$sp) # Save high on the stack
		; 76: sw \$ra, 12(\$sp) # Save \$ra on the stack
		; 78: move \$s1, \$a1 # \$s1 = \$a1
		; 79: move \$s2, \$a2 # \$s2 = \$a2
		; 81: sll \$t1, \$s2, 2 # t1 = high * 4
		; 82: add \$t1, \$a0, \$t1 # t1 = arr + high * 4
		; 83: lw \$t2, 0(\$t1) # pivot = t2 = arr[high]
		; 85: add \$t3, \$s1, -1 # t3 = i = low - 1
		; 86: move \$t4, \$s1 # t4 = j = low
		; 87: addi \$t5, \$s2, -1 # t5 = high - 1

Int Regs [16]		Text
R2	[v0] = 4	[0040009c] 3402000a ori \$2, \$0, 10
R3	[v1] = 0	[004000a0] 0000000c syscall
R4	[a0] = 1001004c	<b>[004000a4] 23bdffff addi \$29, \$29, -16</b> ; 49: addi \$sp, \$sp, -16 # Allocate space on the stack
R5	[a1] = 0	[004000a8] afa40000 sw \$4, 0(\$29)
R6	[a2] = 0	[004000ac] afa50004 sw \$5, 4(\$29)
R7	[a3] = 0	[004000b0] afa60008 sw \$6, 8(\$29)
R8	[t0] = d	[004000b4] afaa000c sw \$10, 12(\$29)
R9	[t1] = 0	<b>[004000b8] 00054880 sll \$9, \$5, 2</b> ; 50: sll \$t1, \$a1, 2 # t1 = arr + Low * 4
R10	[t2] = 0	[004000bc] 00894820 add \$9, \$4, \$9
R11	[t3] = 0	[004000c0] 0d330000 lw \$19, 0(\$9)
R12	[t4] = 0	[004000c4] 00065080 sll \$10, \$6, 2
R13	[t5] = c	[004000c8] 008a5020 add \$10, \$4, \$10
R14	[t6] = 0	[004000cc] 8d540000 lw \$20, 0(\$10)
R15	[t7] = 2b	[004000d0] ad340000 sw \$20, 0(\$9)
R16	[s0] = 0	[004000d4] ad530000 sw \$19, 0(\$10)
R17	[s1] = 0	[004000d8] 8faa000c lw \$10, 12(\$29)
R18	[s2] = d	[004000dc] 23bd0010 addi \$29, \$29, 16
R19	[s3] = 0	[004000e0] 03e00008 jr \$31
R20	[s4] = 0	[004000e4] 23bdffff addi \$29, \$29, -16
R21	[s5] = 0	[004000e8] afa40000 sw \$4, 0(\$29)
R22	[s6] = 0	[004000ec] afa50004 sw \$5, 4(\$29)
R23	[s7] = 0	[004000f0] afa60008 sw \$6, 8(\$29)
R24	[t8] = 0	[004000f4] afbf000c sw \$31, 12(\$29)
R25	[t9] = 0	[004000f8] 00058821 add \$17, \$0, \$5
R26	[k0] = 0	[00400102] 00069021 addu \$18, \$0, \$6
R27	[k1] = 0	[00400106] 00124880 add \$9, \$18, 2
R28	[gp] = 10008000	[0040010a] 00894820 add \$9, \$4, \$9
R29	[sp] = 7fffff16c	[00400108] 8d2a0000 lw \$10, 0(\$9)
R30	[s8] = 0	[0040010c] 222bffff addi \$11, \$17, -1
R31	[ra] = 400144	[00400110] 00116021 addu \$12, \$0, \$17
		<b>[00400114] 224dffff addi \$13, \$18, -1</b>
		; 69: jr \$ra
		; 71: addi \$sp, \$sp, -16 # Allocate space on the stack
		; 73: sw \$a0, 0(\$sp) # Save array on the stack
		; 74: sw \$a1, 4(\$sp) # Save low on the stack
		; 75: sw \$a2, 8(\$sp) # Save high on the stack
		; 76: sw \$ra, 12(\$sp) # Save \$ra on the stack
		; 78: move \$s1, \$a1 # \$s1 = \$a1
		; 79: move \$s2, \$a2 # \$s2 = \$a2
		; 81: sll \$t1, \$s2, 2 # t1 = high * 4
		; 82: add \$t1, \$a0, \$t1 # t1 = arr + high * 4
		; 83: lw \$t2, 0(\$t1) # pivot = t2 = arr[high]
		; 85: add \$t3, \$s1, -1 # t3 = i = low - 1
		; 86: move \$t4, \$s1 # t4 = j = low
		; 87: addi \$t5, \$s2, -1 # t5 = high - 1

Int Regs [16]		Text
R2	[v0] = 4	[0040009c] 3402000a ori \$2, \$0, 10 ; 45: li \$v0, 10
R3	[v1] = 0	[004000a0] 0000000c syscall ; 46: syscall
R4	[a0] = 1001004c	[004000a4] 23bdffff addi \$29, \$29, -16 ; 49: addi \$sp, \$sp, -16 # Allocate space on the stack
R5	[a1] = 0	[004000a8] afa40000 sw \$4, 0(\$29) ; 51: sw \$a0, 0(\$sp) # Store array
R6	[a2] = 0	[004000ac] afa50004 sw \$5, 4(\$29) ; 52: sw \$a1, 4(\$sp) # Store Low
R7	[a3] = 0	[004000b0] afa60008 sw \$6, 8(\$29) ; 53: sw \$a2, 8(\$sp) # Store High
R8	[t0] = d	[004000b4] afaaa000c sw \$10, 12(\$29) ; 54: sw \$t2, 12(\$sp) # Store t2 for high
R9	[t1] = 1001004c	[004000b8] 00054880 sll \$9, \$5, 2 ; 56: sll \$t1, \$a1, 2 # \$t1 = Low * 4
R10	[t2] = 0	[004000bc] 00894820 add \$9, \$4, \$9 ; 57: add \$t1, \$a0, \$t1 # \$t1 = arr + Low * 4
R11	[t3] = 0	[004000c0] 8d330000 lw \$19, 0(\$9) ; 58: lw \$s3, 0(\$t1) # \$s3 = arr[\$t1]
R12	[t4] = 0	[004000c4] 00065080 sll \$10, \$6, 2 ; 60: sll \$t2, \$a2, 2 # \$t2 = High * 4
R13	[t5] = c	[004000cc] 008a5020 add \$10, \$4, \$10 ; 61: add \$t2, \$a0, \$t2 # \$t2 = arr + High * 4
R14	[t6] = 0	[004000cc] 8d540000 lw \$20, 0(\$10) ; 62: lw \$s4, 0(\$t2) # \$s4 = arr[\$t2]
R15	[t7] = 2b	[004000d0] ad340000 sw \$20, 0(\$9) ; 64: sw \$s4, 0(\$t1) # arr[\$t1] = \$s4 (swap a[\$t1] and a[\$t2])
R16	[s0] = 0	[004000d4] ad350000 sw \$19, 0(\$10) ; 65: sw \$s3, 0(\$t2) # arr[\$t2] = \$s3
R17	[s1] = 0	[004000d8] 8faa000c lw \$10, 12(\$29) ; 67: lw \$t2, 12(\$sp) # Load \$t2 on the stack
R18	[s2] = d	[004000dc] 23bd0010 addi \$29, \$29, 16 ; 68: addi \$sp, \$sp, 16
R19	[s3] = 0	[004000e0] 03e00008 jr \$31 ; 69: jr \$ra
R20	[s4] = 0	[004000e4] 23bdffff addi \$29, \$29, -16 ; 71: addi \$sp, \$sp, -16 # Allocate space on the stack
R21	[s5] = 0	[004000e8] afa40000 sw \$4, 0(\$29) ; 73: sw \$a0, 0(\$sp) # Save array on the stack
R22	[s6] = 0	[004000ec] afa50004 sw \$5, 4(\$29) ; 74: sw \$a1, 4(\$sp) # Save low on the stack
R23	[s7] = 0	[004000f0] afa60008 sw \$6, 8(\$29) ; 75: sw \$a2, 8(\$sp) # Save high on the stack
R24	[t8] = 0	[004000f4] afbf000c sw \$31, 12(\$29) ; 76: sw \$ra, 12(\$sp) # Save \$ra on the stack
R25	[t9] = 0	[004000f8] 00058821 addu \$17, \$0, \$5 ; 78: move \$s1, \$a1 # \$s1 = \$a1
R26	[k0] = 0	[004000fc] 00069021 addu \$18, \$0, \$6 ; 79: move \$s2, \$a2 # \$s2 = \$a2
R27	[k1] = 0	[00400100] 00124880 sll \$9, \$18, 2 ; 81: sll \$t1, \$s2, 2 # \$t1 = high * 4
R28	[gp] = 10008000	[00400104] 00894820 add \$9, \$4, \$9 ; 82: add \$t1, \$a0, \$t1 # \$t1 = arr + high * 4
R29	[sp] = 7fffff16c	[00400108] 8d2a0000 lw \$10, 0(\$9) ; 83: lw \$t2, 0(\$t1) # pivot = t2 = arr[high]
R30	[s8] = 0	[0040010c] 222bffff addi \$11, \$17, -1 ; 85: addi \$t3, \$s1, -1 # \$t3 = i = low - 1
R31	[ra] = 400144	[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]		Text
R2	[v0] = 4	[0040009c] 3402000a ori \$2, \$0, 10 ; 45: li \$v0, 10
R3	[v1] = 0	[004000a0] 0000000c syscall ; 46: syscall
R4	[a0] = 1001004c	[004000a4] 23bdffff addi \$29, \$29, -16 ; 49: addi \$sp, \$sp, -16 # Allocate space on the stack
R5	[a1] = 0	[004000a8] afa40000 sw \$4, 0(\$29) ; 51: sw \$a0, 0(\$sp) # Store array
R6	[a2] = 0	[004000ac] afa50004 sw \$5, 4(\$29) ; 52: sw \$a1, 4(\$sp) # Store Low
R7	[a3] = 0	[004000b0] afa60008 sw \$6, 8(\$29) ; 53: sw \$a2, 8(\$sp) # Store High
R8	[t0] = d	[004000b4] afaaa000c sw \$10, 12(\$29) ; 54: sw \$t2, 12(\$sp) # Store t2 for high
R9	[t1] = 1001004c	[004000b8] 00054880 sll \$9, \$5, 2 ; 56: sll \$t1, \$a1, 2 # \$t1 = Low * 4
R10	[t2] = 0	[004000bc] 00894820 add \$9, \$4, \$9 ; 57: add \$t1, \$a0, \$t1 # \$t1 = arr + Low * 4
R11	[t3] = 0	[004000c0] 8d330000 lw \$19, 0(\$9) ; 58: lw \$s3, 0(\$t1) # \$s3 = arr[\$t1]
R12	[t4] = 0	[004000c4] 00065080 sll \$10, \$6, 2 ; 60: sll \$t2, \$a2, 2 # \$t2 = High * 4
R13	[t5] = c	[004000cc] 008a5020 add \$10, \$4, \$10 ; 61: add \$t2, \$a0, \$t2 # \$t2 = arr + High * 4
R14	[t6] = 0	[004000cc] 8d540000 lw \$20, 0(\$10) ; 62: lw \$s4, 0(\$t2) # \$s4 = arr[\$t2]
R15	[t7] = 2b	[004000d0] ad340000 sw \$20, 0(\$9) ; 64: sw \$s4, 0(\$t1) # arr[\$t1] = \$s4 (swap a[\$t1] and a[\$t2])
R16	[s0] = 0	[004000d4] ad350000 sw \$19, 0(\$10) ; 65: sw \$s3, 0(\$t2) # arr[\$t2] = \$s3
R17	[s1] = 0	[004000d8] 8faa000c lw \$10, 12(\$29) ; 67: lw \$t2, 12(\$sp) # Load \$t2 on the stack
R18	[s2] = d	[004000dc] 23bd0010 addi \$29, \$29, 16 ; 68: addi \$sp, \$sp, 16
R19	[s3] = 2b	[004000e0] 03e00008 jr \$31 ; 69: jr \$ra
R20	[s4] = 0	[004000e4] 23bdffff addi \$29, \$29, -16 ; 71: addi \$sp, \$sp, -16 # Allocate space on the stack
R21	[s5] = 0	[004000e8] afa40000 sw \$4, 0(\$29) ; 73: sw \$a0, 0(\$sp) # Save array on the stack
R22	[s6] = 0	[004000ec] afa50004 sw \$5, 4(\$29) ; 74: sw \$a1, 4(\$sp) # Save low on the stack
R23	[s7] = 0	[004000f0] afa60008 sw \$6, 8(\$29) ; 75: sw \$a2, 8(\$sp) # Save high on the stack
R24	[t8] = 0	[004000f4] afbf000c sw \$31, 12(\$29) ; 76: sw \$ra, 12(\$sp) # Save \$ra on the stack
R25	[t9] = 0	[004000f8] 00058821 addu \$17, \$0, \$5 ; 78: move \$s1, \$a1 # \$s1 = \$a1
R26	[k0] = 0	[004000fc] 00069021 addu \$18, \$0, \$6 ; 79: move \$s2, \$a2 # \$s2 = \$a2
R27	[k1] = 0	[00400100] 00124880 sll \$9, \$18, 2 ; 81: sll \$t1, \$s2, 2 # \$t1 = high * 4
R28	[gp] = 10008000	[00400104] 00894820 add \$9, \$4, \$9 ; 82: add \$t1, \$a0, \$t1 # \$t1 = arr + high * 4
R29	[sp] = 7fffff16c	[00400108] 8d2a0000 lw \$10, 0(\$9) ; 83: lw \$t2, 0(\$t1) # pivot = t2 = arr[high]
R30	[s8] = 0	[0040010c] 222bffff addi \$11, \$17, -1 ; 85: addi \$t3, \$s1, -1 # \$t3 = i = low - 1
R31	[ra] = 400144	[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]		Text
R2	[v0] = 4	[0040009c] 3402000a ori \$2, \$0, 10 ; 45: li \$v0, 10
R3	[v1] = 0	[004000a0] 0000000c syscall ; 46: syscall
R4	[a0] = 1001004c	[004000a4] 23bdffff addi \$29, \$29, -16 ; 49: addi \$sp, \$sp, -16 # Allocate space on the stack
R5	[a1] = 0	[004000a8] afa40000 sw \$4, 0(\$29) ; 51: sw \$a0, 0(\$sp) # Store array
R6	[a2] = 0	[004000ac] afa50004 sw \$5, 4(\$29) ; 52: sw \$a1, 4(\$sp) # Store Low
R7	[a3] = 0	[004000b0] afa60008 sw \$6, 8(\$29) ; 53: sw \$a2, 8(\$sp) # Store High
R8	[t0] = d	[004000b4] afaaa000c sw \$10, 12(\$29) ; 54: sw \$t2, 12(\$sp) # Store t2 for high
R9	[t1] = 1001004c	[004000b8] 00054880 sll \$9, \$5, 2 ; 56: sll \$t1, \$a1, 2 # \$t1 = Low * 4
R10	[t2] = 0	[004000bc] 00894820 add \$9, \$4, \$9 ; 57: add \$t1, \$a0, \$t1 # \$t1 = arr + Low * 4
R11	[t3] = 0	[004000c0] 8d330000 lw \$19, 0(\$9) ; 58: lw \$s3, 0(\$t1) # \$s3 = arr[\$t1]
R12	[t4] = 0	[004000c4] 00065080 sll \$10, \$6, 2 ; 60: sll \$t2, \$a2, 2 # \$t2 = High * 4
R13	[t5] = c	[004000cc] 008a5020 add \$10, \$4, \$10 ; 61: add \$t2, \$a0, \$t2 # \$t2 = arr + High * 4
R14	[t6] = 0	[004000cc] 8d540000 lw \$20, 0(\$10) ; 62: lw \$s4, 0(\$t2) # \$s4 = arr[\$t2]
R15	[t7] = 2b	[004000d0] ad340000 sw \$20, 0(\$9) ; 64: sw \$s4, 0(\$t1) # arr[\$t1] = \$s4 (swap a[\$t1] and a[\$t2])
R16	[s0] = 0	[004000d4] ad350000 sw \$19, 0(\$10) ; 65: sw \$s3, 0(\$t2) # arr[\$t2] = \$s3
R17	[s1] = 0	[004000d8] 8faa000c lw \$10, 12(\$29) ; 67: lw \$t2, 12(\$sp) # Load \$t2 on the stack
R18	[s2] = d	[004000dc] 23bd0010 addi \$29, \$29, 16 ; 68: addi \$sp, \$sp, 16
R19	[s3] = 2b	[004000e0] 03e00008 jr \$31 ; 69: jr \$ra
R20	[s4] = 0	[004000e4] 23bdffff addi \$29, \$29, -16 ; 71: addi \$sp, \$sp, -16 # Allocate space on the stack
R21	[s5] = 0	[004000e8] afa40000 sw \$4, 0(\$29) ; 73: sw \$a0, 0(\$sp) # Save array on the stack
R22	[s6] = 0	[004000ec] afa50004 sw \$5, 4(\$29) ; 74: sw \$a1, 4(\$sp) # Save low on the stack
R23	[s7] = 0	[004000f0] afa60008 sw \$6, 8(\$29) ; 75: sw \$a2, 8(\$sp) # Save high on the stack
R24	[t8] = 0	[004000f4] afbf000c sw \$31, 12(\$29) ; 76: sw \$ra, 12(\$sp) # Save \$ra on the stack
R25	[t9] = 0	[004000f8] 00058821 addu \$17, \$0, \$5 ; 78: move \$s1, \$a1 # \$s1 = \$a1
R26	[k0] = 0	[004000fc] 00069021 addu \$18, \$0, \$6 ; 79: move \$s2, \$a2 # \$s2 = \$a2
R27	[k1] = 0	[00400100] 00124880 sll \$9, \$18, 2 ; 81: sll \$t1, \$s2, 2 # \$t1 = high * 4
R28	[gp] = 10008000	[00400104] 00894820 add \$9, \$4, \$9 ; 82: add \$t1, \$a0, \$t1 # \$t1 = arr + high * 4
R29	[sp] = 7fffff16c	[00400108] 8d2a0000 lw \$10, 0(\$9) ; 83: lw \$t2, 0(\$t1) # pivot = t2 = arr[high]
R30	[s8] = 0	[0040010c] 222bffff addi \$11, \$17, -1 ; 85: addi \$t3, \$s1, -1 # \$t3 = i = low - 1
R31	[ra] = 400144	[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		[0040009c] 3402000a ori \$2, \$0, 10	; 45: li \$v0, 10
R3 [v1] = 0		[004000a0] 0000000c syscall	; 46: syscall
R4 [a0] = 1001004c		[004000a4] 23bffff0 addi \$29, \$29, -16	; 49: addi \$sp, \$sp, -16 # Allocate space on the stack
R5 [a1] = 0		[004000a8] afa40000 sw \$4, 0(\$29)	; 51: sw \$a0, 0(\$sp) # Store array
R6 [a2] = 0		[004000ac] afa50004 sw \$5, 4(\$29)	; 52: sw \$a1, 4(\$sp) # Store Low
R7 [a3] = 0		[004000b0] afa60008 sw \$6, 8(\$29)	; 53: sw \$a2, 8(\$sp) # Store High
R8 [t0] = d		[004000b4] afaa000c sw \$10, 12(\$29)	; 54: sw \$t2, 12(\$sp) # Store t2 for high
R9 [t1] = 1001004c		[004000b8] 00054800 sll \$9, \$5, 2	; 56: sll \$t1, \$a1, 2 # \$t1 = Low * 4
R10 [t2] = 1001004c		[004000bc] 00894820 add \$9, \$4, \$9	; 57: add \$t1, \$a0, \$t1 # \$t1 = arr + Low * 4
R11 [t3] = 0		[004000cc] 8d330000 lw \$10, 12(\$29)	; 58: lw \$s3, 0(\$t1) # \$s3 = arr[\$t1]
R12 [t4] = 0		[004000c4] 00065080 addi \$29, \$29, 16	; 60: sll \$t2, \$a2, 2 # \$t2 = High * 4
R13 [t5] = c		[004000c8] 008a5020 add \$10, \$4, \$10	; 61: add \$t2, \$a0, \$t2 # \$t2 = arr + High * 4
R14 [t6] = 0		[004000d0] ad340000 sw \$20, 0(\$9)	; 62: 44: addi \$sp, \$sp, 16
R15 [t7] = 2b		[004000d4] 00054800 sll \$9, \$5, 2	; 64: sw \$s4, 0(\$t1) # \$s4 = swap a[\$t1] and a[\$t2]
R16 [s0] = 0		[004000d8] afa50004 sw \$4, 0(\$10)	; 65: sw \$s3, 0(\$t2) # arr[\$t2] = \$s3
R17 [s1] = d		[004000dc] 00894820 lw \$10, 12(\$29)	; 67: 44: lw \$t2, 12(\$sp) # Load \$t2 on the stack
R18 [s2] = d		[004000e0] 8d330000 addi \$29, \$29, 16	; 68: addi \$sp, \$sp, 16
R19 [s3] = 2b		[004000e4] 23bffff0 jr \$31	; 69: jr \$ra
R20 [s4] = 2b		[004000e8] afa40000 addi \$29, \$29, -16	; 71: addi \$sp, \$sp, -16 # Allocate space on the stack
R21 [s5] = 0		[004000ec] afa50004 sw \$4, 0(\$29)	; 73: sw \$a0, 0(\$sp) # Save array
R22 [s6] = 0		[004000f0] afa60008 sw \$5, 4(\$29)	; 74: sw \$a1, 4(\$sp) # Save low on the stack
R23 [s7] = 0		[004000f4] afaa000c sw \$6, 8(\$29)	; 75: sw \$a2, 8(\$sp) # Save high on the stack
R24 [t8] = 0		[004000f8] afb0000c sw \$31, 12(\$29)	; 76: sw \$ra, 12(\$sp) # Save \$ra on the stack
R25 [t9] = 0		[004000fc] 00058821 addi \$17, \$0, \$5	; 78: move \$s1, \$a1 # \$s1 = \$a1
R26 [k0] = 0		[00400100] 00069021 addi \$18, \$0, \$6	; 79: move \$s2, \$a2 # \$s2 = \$a2
R27 [k1] = 0		[00400104] 00894820 sll \$9, \$18, 2	; 81: sll \$t1, \$s2, 2 # \$t1 = high * 4
R28 [gp] = 10008000		[00400108] 8d2a0000 add \$9, \$4, \$9	; 82: add \$t1, \$a0, \$t1 # \$t1 = arr + high * 4
R29 [sp] = 7ffff16c		[0040010c] 222bffff addi \$11, \$17, -1	; 83: lw \$t2, 0(\$t1) # pivot = t2 = arr[high]
R30 [s8] = 0		[00400110] 00116021 addu \$12, \$0, \$17	; 85: addi \$t3, \$s1, -1 # \$t3 = i = low - 1
R31 [ra] = 400144		[00400114] 224dffff addi \$13, \$18, -1	; 86: move \$t4, \$s1 # \$t4 = j = low
			; 87: addi \$t5, \$s2, -1 # \$t5 = high - 1
Int Regs [16]			
R2 [v0] = 4		[0040009c] 3402000a ori \$2, \$0, 10	; 45: li \$v0, 10
R3 [v1] = 0		[004000a0] 0000000c syscall	; 46: syscall
R4 [a0] = 1001004c		[004000a4] 23bffff0 addi \$29, \$29, -16	; 49: addi \$sp, \$sp, -16 # Allocate space on the stack
R5 [a1] = 0		[004000a8] afa40000 sw \$4, 0(\$29)	; 51: sw \$a0, 0(\$sp) # Store array
R6 [a2] = 0		[004000ac] afa50004 sw \$5, 4(\$29)	; 52: sw \$a1, 4(\$sp) # Store Low
R7 [a3] = 0		[004000b0] afa60008 sw \$6, 8(\$29)	; 53: sw \$a2, 8(\$sp) # Store High
R8 [t0] = d		[004000b4] afaa000c sw \$10, 12(\$29)	; 54: sw \$t2, 12(\$sp) # Store t2 for high
R9 [t1] = 1001004c		[004000b8] 00054800 sll \$9, \$5, 2	; 56: sll \$t1, \$a1, 2 # \$t1 = Low * 4
R10 [t2] = 0		[004000bc] 00894820 add \$9, \$4, \$9	; 57: add \$t1, \$a0, \$t1 # \$t1 = arr + Low * 4
R11 [t3] = 0		[004000cc] 8d330000 lw \$19, 0(\$9)	; 58: lw \$s3, 0(\$t1) # \$s3 = arr[\$t1]
R12 [t4] = 0		[004000c4] 00065080 addi \$29, \$29, 16	; 60: sll \$t2, \$a2, 2 # \$t2 = High * 4
R13 [t5] = c		[004000c8] 008a5020 add \$10, \$4, \$10	; 61: add \$t2, \$a0, \$t2 # \$t2 = arr + High * 4
R14 [t6] = 0		[004000dc] 8d540000 lw \$20, 0(\$10)	; 62: 44: lw \$s4, 0(\$t2) # \$s4 = arr[\$t2]
R15 [t7] = 2b		[004000e0] ad340000 sw \$20, 0(\$9)	; 64: sw \$s4, 0(\$t1) # arr[\$t1] = \$s4 (swap a[\$t1] and a[\$t2])
R16 [s0] = 0		[004000e4] ad530000 sw \$19, \$0(\$10)	; 65: sw \$s3, 0(\$t2) # arr[\$t2] = \$s3
R17 [s1] = 0		[004000e8] 8faa000c lw \$10, 12(\$29)	; 67: 44: lw \$t2, 12(\$sp) # Load \$t2 on the stack
R18 [s2] = d		[004000ec] 23bffff0 addi \$29, \$29, 16	; 68: addi \$sp, \$sp, 16
R19 [s3] = 2b		[004000f0] 03e00008 jr \$31	; 69: jr \$ra
R20 [s4] = 2b		[004000f4] 23bffff0 addi \$29, \$29, -16	; 71: addi \$sp, \$sp, -16 # Allocate space on the stack
R21 [s5] = 0		[004000f8] afa40000 sw \$4, 0(\$29)	; 73: sw \$a0, 0(\$sp) # Save array on the stack
R22 [s6] = 0		[004000fc] afa50004 sw \$5, 4(\$29)	; 74: sw \$a1, 4(\$sp) # Save low on the stack
R23 [s7] = 0		[00400100] afa60008 sw \$6, 8(\$29)	; 75: sw \$a2, 8(\$sp) # Save high on the stack
R24 [t8] = 0		[00400104] afb0000c sw \$31, 12(\$29)	; 76: sw \$ra, 12(\$sp) # Save \$ra on the stack
R25 [t9] = 0		[00400108] 00058821 addu \$17, \$0, \$5	; 78: move \$s1, \$a1 # \$s1 = \$a1
R26 [k0] = 0		[0040010c] 00069021 addu \$18, \$0, \$6	; 79: move \$s2, \$a2 # \$s2 = \$a2
R27 [k1] = 0		[00400110] 00124880 sll \$9, \$18, 2	; 81: sll \$t1, \$s2, 2 # \$t1 = high * 4
R28 [gp] = 10008000		[00400114] 00894820 add \$9, \$4, \$9	; 82: add \$t1, \$a0, \$t1 # \$t1 = arr + high * 4
R29 [sp] = 7ffff17c		[00400118] 8d2a0000 lw \$10, 0(\$9)	; 83: lw \$t2, 0(\$t1) # pivot = t2 = arr[high]
R30 [s8] = 0		[0040011c] 222bffff addi \$11, \$17, -1	; 85: addi \$t3, \$s1, -1 # \$t3 = i = low - 1
R31 [ra] = 400144		[00400120] 00116021 addu \$12, \$0, \$17	; 86: move \$t4, \$s1 # \$t4 = j = low
		[00400124] 224dffff addi \$13, \$18, -1	; 87: addi \$t5, \$s2, -1 # \$t5 = high - 1

Int Regs [16]	Text		
R2 [v0] = 4	[0040000f0] afa60008	sw \$6, 8(\$29)	; 75: sw \$a2, 8(\$sp) # Save high on the stack
R3 [v1] = 0	[0040000f4] afbf000c	sw \$31, 12(\$29)	; 76: sw \$ra, 12(\$sp) # Save ra on the stack
R4 [a0] = 1001004c	[0040000f8] 00058821	addu \$17, \$0, \$5	; 78: move \$s1, \$a1 # \$s1 = \$a1
R5 [a1] = 0	[0040000fc] 00069021	addu \$18, \$0, \$6	; 79: move \$s2, \$a2 # \$s2 = \$a2
R6 [a2] = 0	[00400100] 00124880	sll \$9, \$18, 2	; 81: sll \$t1, \$s2, 2 # t1 = high * 4
R7 [a3] = 0	[00400104] 00894820	add \$9, \$4, \$9	; 82: add \$t1, \$a0, \$t1 # t1 = arr + high * 4
R8 [t0] = d	[00400108] 8d2a0000	lw \$10, 0(\$9)	; 83: lw \$t2, 0(\$t1) # pivot = t2 = arr[high]
R9 [t1] = 1001004c	[0040010c] 222bffff	addi \$11, \$17, -1	; 85: addi \$t3, \$s1, -1 # t3 = i = low - 1
R10 [t2] = 0	[00400110] 00116021	addu \$12, \$0, \$17	; 86: move \$t4, \$s1 # t4 = j = low
R11 [t3] = 0	[00400114] 224dffff	addi \$13, \$18, -1	; 87: addi \$t5, \$s2, -1 # t5 = high - 1
R12 [t4] = 0	[00400118] 01ac702a	slt \$14, \$13, \$12	; 90: sll \$t6, \$t5, \$t4 # Check \$t6 = 1 if j > high - 1
R13 [t5] = c	[0040011c] 15c00000	bne \$14, \$0, 56 [endifor-0x0040011c]	
R14 [t6] = 0	[00400120] 000c4880	sll \$9, \$12, 2	; 94: sll \$t1, \$t4, 2 # \$t1 = low * 4
R15 [t7] = 2b	[00400124] 01244880	add \$9, \$9, \$4	; 95: add \$t1, \$t1, \$a0 # \$t1 = arr + low * 4
R16 [s0] = 0	[00400128] 8d2f0000	lw \$15, 0(\$9)	; 96: lw \$t7, 0(\$t1) # \$t7 = arr[j] = arr[low]
R17 [s1] = 0	[0040012c] 01eac02a	slt \$24, \$15, \$10	; 98: sll \$t8, \$t7, \$t2 # Check \$t8 = 1 if arr[j] > pivot
R18 [s2] = d	[00400130] 17000007	bne \$24, \$0, 28 [endifif-0x00400130]	
R19 [s3] = 2b	[00400134] 216b0001	addi \$11, \$11, 1	; 102: addi \$t3, \$t3, 1 # Increase i++
R20 [s4] = 2b	[00400138] 000b2821	addu \$5, \$0, \$11	; 103: move \$a1, \$t3
R21 [s5] = 0	[0040013c] 000c3021	addu \$6, \$0, \$12	; 104: move \$a2, \$t4
R22 [s6] = 0	[00400140] 0c100029	jal 0x004000a4 [SWAP]	; 105: jal SWAP
R23 [s7] = 0	[00400144] 218c0001	addi \$12, \$12, 1	; 107: addi \$t4, \$t4, 1 # j++
R24 [t8] = 0	[00400148] 08100046	j 0x00400118 [forloop]	; 108: j forloop
R25 [t9] = 0	[0040014c] 218c0001	addi \$12, \$12, 1	; 111: addi \$t4, \$t4, 1 # j++
R26 [k0] = 0	[00400150] 08100046	j 0x00400118 [forloop]	; 112: j forloop
R27 [k1] = 0	[00400154] 21650001	addi \$5, \$11, 1	; 115: addi \$a1, \$t3, 1 # \$a1 = i + 1
R28 [gp] = 10008000	[00400158] 00123021	addu \$6, \$0, \$18	; 116: move \$a2, \$s2 # \$a2 = high
R29 [sp] = 7fffff17c	[0040015c] 00051020	add \$2, \$0, \$5	; 117: add \$v0, \$zero, \$a1 # \$v0 = \$a1
R30 [s8] = 0	[00400160] 0c100029	jal 0x004000a4 [SWAP]	; 118: jal SWAP # Swap(arr[i+1], arr[high])
R31 [ra] = 400144	[00400164] 8fbff00c	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

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		
R2 [v0] = 4	[0040000f4] afbf000c	sw \$31, 12(\$29)	; 76: sw \$ra, 12(\$sp) # Save ra on the stack
R3 [v1] = 0	[0040000f8] 00058821	addu \$17, \$0, \$5	; 78: move \$s1, \$a1 # \$s1 = \$a1
R4 [a0] = 1001004c	[0040000fc] 00069021	addu \$18, \$0, \$6	; 79: move \$s2, \$a2 # \$s2 = \$a2
R5 [a1] = d	[00400100] 00124880	sll \$9, \$18, 2	; 81: sll \$t1, \$s2, 2 # t1 = high * 4
R6 [a2] = c	[00400104] 00894820	add \$9, \$4, \$9	; 82: add \$t1, \$a0, \$t1 # t1 = arr + high * 4
R7 [a3] = 0	[00400108] 8d2a0000	lw \$10, 0(\$9)	; 83: lw \$t2, 0(\$t1) # pivot = t2 = arr[high]
R8 [t0] = d	[0040010c] 222bffff	addi \$11, \$17, -1	; 85: addi \$t3, \$s1, -1 # t3 = i = low - 1
R9 [t1] = 1001007c	[00400110] 00116021	addu \$12, \$0, \$17	; 86: move \$t4, \$s1 # t4 = j = low
R10 [t2] = 0	[00400114] 224dffff	addi \$13, \$18, -1	; 87: addi \$t5, \$s2, -1 # t5 = high - 1
R11 [t3] = c	[00400118] 01ac702a	slt \$14, \$13, \$12	; 90: sll \$t6, \$t5, \$t4 # Check \$t6 = 1 if j > high - 1
R12 [t4] = d	[0040011c] 15c00000	bne \$14, \$0, 56 [endifor-0x0040011c]	
R13 [t5] = c	[00400120] 000c4880	sll \$9, \$12, 2	; 94: sll \$t1, \$t4, 2 # \$t1 = low * 4
R14 [t6] = 1	[00400124] 01244880	add \$9, \$9, \$4	; 95: add \$t1, \$t1, \$a0 # \$t1 = arr + low * 4
R15 [t7] = 38	[00400128] 8d2f0000	lw \$15, 0(\$9)	; 96: lw \$t7, 0(\$t1) # \$t7 = arr[j] = arr[low]
R16 [s0] = 0	[0040012c] 01eac02a	slt \$24, \$15, \$10	; 98: sll \$t8, \$t7, \$t2 # Check \$t8 = 1 if arr[j] > pivot
R17 [s1] = 0	[00400130] 17000007	bne \$24, \$0, 28 [endifif-0x00400130]	
R18 [s2] = d	[00400134] 216b0001	addi \$11, \$11, 1	; 102: addi \$t3, \$t3, 1 # Increase i++
R19 [s3] = 38	[00400138] 000b2821	addu \$5, \$0, \$11	; 103: move \$a1, \$t3
R20 [s4] = 38	[0040013c] 000c3021	addu \$6, \$0, \$12	; 104: move \$a2, \$t4
R21 [s5] = 0	[00400140] 0c100029	jal 0x004000a4 [SWAP]	; 105: jal SWAP
R22 [s6] = 0	[00400144] 218c0001	addi \$12, \$12, 1	; 107: addi \$t4, \$t4, 1 # j++
R23 [s7] = 0	[00400148] 08100046	j 0x00400118 [forloop]	; 108: j forloop
R24 [t8] = 0	[0040014c] 218c0001	addi \$12, \$12, 1	; 111: addi \$t4, \$t4, 1 # j++
R25 [t9] = 0	[00400150] 08100046	j 0x00400118 [forloop]	; 112: j forloop
R26 [k0] = 0	[00400154] 21650001	addi \$5, \$11, 1	; 115: addi \$a1, \$t3, 1 # \$a1 = i + 1
R27 [k1] = 0	[00400158] 00123021	addu \$6, \$0, \$18	; 116: move \$a2, \$s2 # \$a2 = high
R28 [gp] = 10008000	[0040015c] 00051020	add \$2, \$0, \$5	; 117: add \$v0, \$zero, \$a1 # \$v0 = \$a1
R29 [sp] = 7fffff17c	[00400160] 0c100029	jal 0x004000a4 [SWAP]	; 118: jal SWAP # Swap(arr[i+1], arr[high])
R30 [s8] = 0	[00400164] 8fbff00c	lw \$31, 12(\$29)	; 120: lw \$ra, 12(\$sp) # Return address
R31 [ra] = 400144	[00400168] 23bd0010	addi \$29, \$29, 16	; 121: addi \$sp, \$sp, 16 # Restore the stack
	[0040016c] 03e00008	jr \$31	; 122: jr \$ra

Int Regs [16]	Text		
R2 [v0] = 4	[0040000f4] afbf000c	sw \$31, 12(\$29)	; 76: sw \$ra, 12(\$sp) # Save ra on the stack
R3 [v1] = 0	[0040000f8] 00058821	addu \$17, \$0, \$5	; 78: move \$s1, \$a1 # \$s1 = \$a1
R4 [a0] = 1001004c	[0040000fc] 00069021	addu \$18, \$0, \$6	; 79: move \$s2, \$a2 # \$s2 = \$a2
R5 [a1] = d	[00400100] 00124880	sll \$9, \$18, 2	; 81: sll \$t1, \$s2, 2 # t1 = high * 4
R6 [a2] = d	[00400104] 00894820	add \$9, \$4, \$9	; 82: add \$t1, \$a0, \$t1 # t1 = arr + high * 4
R7 [a3] = 0	[00400108] 8d2a0000	lw \$10, 0(\$9)	; 83: lw \$t2, 0(\$t1) # pivot = t2 = arr[high]
R8 [t0] = d	[0040010c] 222bffff	addi \$11, \$17, -1	; 85: addi \$t3, \$s1, -1 # t3 = i = low - 1
R9 [t1] = 1001007c	[00400110] 00116021	addu \$12, \$0, \$17	; 86: move \$t4, \$s1 # t4 = j = low
R10 [t2] = 0	[00400114] 224dffff	addi \$13, \$18, -1	; 87: addi \$t5, \$s2, -1 # t5 = high - 1
R11 [t3] = c	[00400118] 01ac702a	slt \$14, \$13, \$12	; 90: sll \$t6, \$t5, \$t4 # Check \$t6 = 1 if j > high - 1
R12 [t4] = d	[0040011c] 15c00000	bne \$14, \$0, 56 [endifor-0x0040011c]	
R13 [t5] = c	[00400120] 000c4880	sll \$9, \$12, 2	; 94: sll \$t1, \$t4, 2 # \$t1 = low * 4
R14 [t6] = 1	[00400124] 01244880	add \$9, \$9, \$4	; 95: add \$t1, \$t1, \$a0 # \$t1 = arr + low * 4
R15 [t7] = 38	[00400128] 8d2f0000	lw \$15, 0(\$9)	; 96: lw \$t7, 0(\$t1) # \$t7 = arr[j] = arr[low]
R16 [s0] = 0	[0040012c] 01eac02a	slt \$24, \$15, \$10	; 98: sll \$t8, \$t7, \$t2 # Check \$t8 = 1 if arr[j] > pivot
R17 [s1] = 0	[00400130] 17000007	bne \$24, \$0, 28 [endifif-0x00400130]	
R18 [s2] = d	[00400134] 216b0001	addi \$11, \$11, 1	; 102: addi \$t3, \$t3, 1 # Increase i++
R19 [s3] = 38	[00400138] 000b2821	addu \$5, \$0, \$11	; 103: move \$a1, \$t3
R20 [s4] = 38	[0040013c] 000c3021	addu \$6, \$0, \$12	; 104: move \$a2, \$t4
R21 [s5] = 0	[00400140] 0c100029	jal 0x004000a4 [SWAP]	; 105: jal SWAP
R22 [s6] = 0	[00400144] 218c0001	addi \$12, \$12, 1	; 107: addi \$t4, \$t4, 1 # j++
R23 [s7] = 0	[00400148] 08100046	j 0x00400118 [forloop]	; 108: j forloop
R24 [t8] = 0	[0040014c] 218c0001	addi \$12, \$12, 1	; 111: addi \$t4, \$t4, 1 # j++
R25 [t9] = 0	[00400150] 08100046	j 0x00400118 [forloop]	; 112: j forloop
R26 [k0] = 0	[00400154] 21650001	addi \$5, \$11, 1	; 115: addi \$a1, \$t3, 1 # \$a1 = i + 1
R27 [k1] = 0	[00400158] 00123021	addu \$6, \$0, \$18	; 116: move \$a2, \$s2 # \$a2 = high
R28 [gp] = 10008000	[0040015c] 00051020	add \$2, \$0, \$5	; 117: add \$v0, \$zero, \$a1 # \$v0 = \$a1
R29 [sp] = 7fffff17c	[00400160] 0c100029	jal 0x004000a4 [SWAP]	; 118: jal SWAP # Swap(arr[i+1], arr[high])
R30 [s8] = 0	[00400164] 8fbff00c	lw \$31, 12(\$29)	; 120: lw \$ra, 12(\$sp) # Return address
R31 [ra] = 400144	[00400168] 23bd0010	addi \$29, \$29, 16	; 121: addi \$sp, \$sp, 16 # Restore the stack
	[0040016c] 03e00008	jr \$31	; 122: jr \$ra

Int Regs [16] Text

```

R2 [v0] = d [004000f8] 00058821 addu $17, $0, $5 ; 76: sw $ra, 12($sp) # Save $ra on the stack
R3 [v1] = 0 [004000fc] 00069021 addu $18, $0, $6 ; 78: move $s1, $a1 # $s1 = $a1
R4 [a0] = 10010000 [00400100] 00124880 sll $9, $18, 2 ; 79: move $s2, $a2 # $s2 = $a2
R5 [a1] = d [00400104] 00894820 add $9, $4, $9 ; 81: sll $t1, $s2, 2 # $t1 = high * 4
R6 [a2] = d [0040010c] 222bffff add $11, $17, -1 ; 82: add $t1, $s0, $t1 # $t1 = arr + high * 4
R7 [a3] = 0 [00400110] 00116021 add $12, $0, $-1 ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
R8 [t0] = d [00400118] 01ac702a slt $14, $13, $12 ; 85: add $t3, $s1, -1 # $t3 = i = low - 1
R9 [t1] = 1001007c [00400120] 000c4880 sll $9, $12, 2 ; 86: move $t4, $s1 # $t4 = j = low
R10 [t2] = 0 [00400124] 01244820 add $9, $9, $4 ; 87: add $t5, $s2, -1 # $t5 = high - 1
R11 [t3] = c [00400128] 8d2f0000 lw $10, 0($9) ; 89: sll $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
R12 [t4] = d [00400134] 216b0001 addi $11, $11, 1 ; 90: sll $t1, $t4, 2 # $t1 = low * 4
R13 [t5] = c [00400138] 000b2821 addi $5, $0, $11 ; 94: sll $t1, $t4, 2 # $t1 = low * 4
R14 [t6] = 1 [0040013c] 000c3021 addi $6, $0, $12 ; 95: add $t1, $t1, $s0 # $t1 = arr + low * 4
R15 [t7] = 38 [00400140] 0c100029 jal 0x04000a04 [SWAP] ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
R16 [s0] = 0 [00400144] 218c0001 addi $12, $12, 1 ; 98: sll $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
R17 [s1] = 0 [00400148] 08100046 j 0x0400118 [forloop] ; 100: j forloop
R18 [s2] = d [0040014c] 218c0001 addi $12, $12, 1 ; 102: addi $t3, $t3, 1 # Increase i++
R19 [s3] = 38 [00400150] 08100046 j 0x0400118 [forloop] ; 103: move $a1, $t3
R20 [s4] = 38 [00400154] 21650001 addi $5, $11, 1 ; 104: move $a2, $t4
R21 [s5] = 0 [00400158] 00123021 addi $6, $0, $18 ; 105: jal SWAP
R22 [s6] = 0 [00400160] 0c100029 jal 0x04000a04 [SWAP] ; 107: addi $t4, $t4, 1 # j++
R23 [s7] = 0 [00400164] 8fb00000 lw $31, 12($29) ; 108: j forloop
R24 [t8] = 0 [00400168] 23bd0010 addi $29, $29, 16 ; 111: addi $t4, $t4, 1 # j++
R25 [t9] = 0 [0040016c] 03e00008 jr $31 ; 112: j forloop
R26 [k0] = 0 [00400170] 23bdffff addi $29, $29, -16 ; 115: addi $a1, $t3, 1 # $a1 = i + 1
R27 [k1] = 0 [00400174] afa40000 sw $4, 0($29) ; 116: move $a2, $s2 # $a2 = high
R28 [gp] = 10008000 [00400178] afa50004 sw $5, 4($29) ; 117: add $t0, $t0, $0 # $t0 = $a1
R29 [sp] = 7fffff17c [00400180] afbff000 lw $31, 12($29) ; 118: jal SWAP # Swap(arr[i+1], arr[high])
R30 [s8] = 0 [00400184] 00064021 addu $8, $0, $6 ; 120: lw $ra, 12($sp) # Return address
R31 [ra] = 400164 [00400188] 00a8482a sll $9, $5, $8 ; 121: addi $sp, $sp, 16 # Restore the stack
R32 [ra] = 11200009 beq $9, $0, 36 [ENDIF-0x040018c] ; 122: jr $ra

```

FP Regs Int Regs [16] Data Text

```

R2 [v0] = d [00400114] 2240ffff addi $15, $16, -1 ; 0: addi $t0, $s2, -1 # $t0 = high - 1
R3 [v1] = 0 [00400118] 01ac702a slt $14, $13, $12 ; 90: sll $t6, $t5, $t4 # Check $t6 = 1 if j > high - 1
R4 [a0] = 1001004c [0040011c] 15c00000 bne $14, $0, $5 ; 91: move $s1, $a1, $t3
R5 [a1] = d [00400120] 000c4880 sll $9, $12, 2 ; 94: sll $t1, $t4, 2 # $t1 = low * 4
R6 [a2] = d [00400124] 01244820 add $9, $9, $4 ; 95: add $t1, $t1, $s0 # $t1 = arr + low * 4
R7 [a3] = 0 [00400128] 8d2f0000 lw $15, 0($9) ; 96: lw $t7, 0($t1) # $t7 = arr[j] = arr[low]
R8 [t0] = d [0040012c] 01eac02a slt $24, $15, $10 ; 98: sll $t8, $t7, $t2 # Check $t8 = 1 if arr[j] > pivot
R9 [t1] = 1001007c [00400130] 17000007 bne $24, $0, 28 [endiff-0x0400130]
R10 [t2] = 0 [00400134] 216b0001 addi $11, $11, 1 ; 102: addi $t3, $t3, 1 # Increase i++
R11 [t3] = c [00400138] 000b2821 addu $5, $0, $11 ; 103: move $a1, $t3
R12 [t4] = d [0040013c] 000c3021 addu $6, $0, $12 ; 104: move $a2, $t4
R13 [t5] = c [00400140] 0c100029 jal 0x04000a04 [SWAP] ; 105: jal SWAP
R14 [t6] = 1 [00400144] 218c0001 addi $12, $12, 1 ; 107: addi $t4, $t4, 1 # j++
R15 [t7] = 38 [00400148] 08100046 j 0x0400118 [forloop] ; 108: j forloop
R16 [s0] = 0 [00400152] 218c0001 addi $12, $12, 1 ; 111: addi $t4, $t4, 1 # j++
R17 [s1] = 0 [00400154] 21650001 addi $5, $11, 1 ; 115: addi $a1, $t3, 1 # $a1 = i + 1
R18 [s2] = d [00400158] 00123021 addu $6, $0, $18 ; 116: move $a2, $s2 # $a2 = high
R19 [s3] = 38 [0040015c] 00051020 add $2, $0, $5 ; 117: add $t0, $t0, $0 # $t0 = $a1
R20 [s4] = 38 [00400160] 0c100029 jal 0x04000a04 [SWAP] ; 118: jal SWAP # Swap(arr[i+1], arr[high])
R21 [s5] = 0 [00400164] 8fb00000 lw $31, 12($29) ; 120: lw $ra, 12($sp) # Return address
R22 [s6] = 0 [00400168] 23bd0010 addi $29, $29, 16 ; 121: addi $sp, $sp, 16 # Restore the stack
R23 [s7] = 0 [0040016c] 03e00008 jr $31 ; 122: jr $ra
R24 [t8] = 0 [00400170] 23bdffff addi $29, $29, -16 ; 125: addi $sp, $sp, -16 # Allocate space on the stack
R25 [t9] = 0 [00400174] afa40000 sw $4, 0($29) ; 127: sw $a0, 0($sp) # save arr
R26 [k0] = 0 [00400178] afa50004 sw $5, 4($29) ; 128: sw $a1, 4($sp) # save low
R27 [k1] = 0 [0040017c] afa60004 sw $6, 8($29) ; 129: sw $a2, 8($sp) # save high
R28 [gp] = 10008000 [00400180] afbff000 lw $31, 12($29) ; 130: lw $ra, 12($sp) # Return address
R29 [sp] = 7fffff17c [00400184] 00064021 addu $8, $0, $6 ; 132: move $t0, $s2 # $t0 = high
R30 [s8] = 0 [00400188] 00a8482a sll $9, $5, $8 ; 134: sll $t1, $a1, $t0 # Check $t1 = 1 if low > high
R31 [ra] = 400164 [0040018c] 11200009 beq $9, $0, 36 [ENDIF-0x040018c]

```

Int Regs [16] Text

```

R2 [v0] = d [00400090] 34240048 ori $4, $1, 72 [strLine]
R3 [v1] = 0 [00400094] 34020004 ori $2, $0, 4 ; 42: li $v0, 4
R4 [a0] = 1001004c [00400098] 00000000 syscall ; 43: syscall
R5 [a1] = d [0040009c] 34020000 ori $2, $0, 10 ; 45: li $v0, 10
R6 [a2] = d [004000a0] 00000000 syscall ; 46: syscall
R7 [a3] = 0 [004000a4] 23bdffff addi $29, $29, -16 ; 49: addi $sp, $sp, -16 # Allocate space on the stack
R8 [t0] = d [004000a8] afa40000 sw $4, 0($29) ; 51: sw $a0, 0($sp) # Store array
R9 [t1] = 1001007c [004000ac] afa50004 sw $5, 4($29) ; 52: sw $a1, 4($sp) # Store Low
R10 [t2] = 0 [004000b0] afa60004 sw $6, 8($29) ; 53: sw $a2, 8($sp) # Store High
R11 [t3] = c [004000b4] afa0000c sw $10, $12($29) ; 54: sw $t2, 12($sp) # Store t2 for high
R12 [t4] = d [004000b8] 00548800 sll $9, $5, 2 ; 56: sll $t1, $a1, 2 # $t1 = Low * 4
R13 [t5] = c [004000bc] 00894820 add $9, $4, $9 ; 57: add $t1, $a0, $t1 # $t1 = arr + Low * 4
R14 [t6] = 1 [004000cc] 0d330000 lw $19, 0($9) ; 58: lw $s3, 0($t1) # $s3 = arr[t1]
R15 [t7] = 38 [004000d0] 00065080 sll $10, $6, 2 ; 60: sll $t2, $a2, 2 # $t2 = High * 4
R16 [s0] = 0 [004000d4] 00a50200 add $10, $4, $10 ; 61: add $t2, $a0, $t2 # $t2 = arr + High * 4
R17 [s1] = 0 [004000d8] 0d340000 lw $20, 0($10) ; 62: lw $s4, 0($t2) # $s4 = arr[t2]
R18 [s2] = d [004000e0] 00a60000 sw $20, 0($9) ; 64: sw $s4, 0($t1) # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
R19 [s3] = 38 [004000e4] ad530000 sw $19, 0($10) ; 65: sw $s3, 0($t2) # arr[$t2] = $s3
R20 [s4] = 38 [004000e8] 0f8aa000c lw $10, 12($29) ; 67: lw $t2, 12($sp) # Load $t2 on the stack
R21 [s5] = 0 [004000ec] 23bd0010 addi $29, $29, 16 ; 68: addi $sp, $sp, 16
R22 [s6] = 0 [004000f0] 03e00008 jr $31 ; 69: jr $ra
R23 [s7] = 0 [004000f4] 23bdffff addi $29, $29, -16 ; 71: addi $sp, $sp, -16 # Allocate space on the stack
R24 [t8] = 0 [004000f8] afa40000 sw $4, 0($29) ; 73: sw $a0, 0($sp) # Save array on the stack
R25 [t9] = 0 [004000fc] afa50004 sw $5, 4($29) ; 74: sw $a1, 4($sp) # Save low on the stack
R26 [k0] = 0 [00400100] afa60004 sw $6, 8($29) ; 75: sw $a2, 8($sp) # Save high on the stack
R27 [k1] = 0 [00400104] afbff000 sw $31, 12($29) ; 76: sw $ra, 12($sp) # Save $ra on the stack
R28 [gp] = 10008000 [00400108] 00058821 addu $17, $0, $5 ; 78: move $s1, $a1 # $s1 = $a1
R29 [sp] = 7fffff17c [0040010c] 00069021 addu $18, $0, $6 ; 79: move $s2, $a2 # $s2 = $a2
R30 [s8] = 0 [00400110] 00124880 sll $9, $18, 2 ; 81: sll $t1, $s2, 2 # $t1 = high * 4
R31 [ra] = 400164 [00400114] 00894820 add $9, $4, $9 ; 82: add $t1, $a0, $t1 # $t1 = arr + high * 4
R32 [ra] = 11200009 lw $10, 0($9) ; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]

```

Int Regs [16]		Text
R2	[v0] = d	[00400090] 34240048 ori \$4, \$1, 72 [strLine]
R3	[v1] = 0	[00400094] 34020004 ori \$2, \$0, 4 ; 42: li \$v0, 4
R4	[a0] = 1001004c	[00400098] 00000000 syscall ; 43: syscall
R5	[a1] = d	[0040009c] 3402000a ori \$2, \$0, 10 ; 45: li \$v0, 10
R6	[a2] = d	[004000a0] 0000000c syscall ; 46: syscall
R7	[a3] = 0	[004000a4] 23bdfff0 addi \$29, \$29, -16 ; 49: addi \$sp, \$sp, -16 # Allocate space on the stack
R8	[t0] = d	[004000a8] afa40000 sw \$4, 0(\$29) ; 51: sw \$a0, 0(\$sp) # Store array
R9	[t1] = 34	[004000ac] afa50004 sw \$5, 4(\$29) ; 52: sw \$a1, 4(\$sp) # Store Low
R10	[t2] = 0	[004000b0] afa60008 sw \$6, 8(\$29) ; 53: sw \$a2, 8(\$sp) # Store High
R11	[t3] = c	[004000b4] afaaa000c sw \$10, 12(\$29) ; 54: sw \$t2, 12(\$sp) # Store t2 for high
R12	[t4] = d	[004000b8] 00054880 sll \$9, \$5, 2 ; 56: sll \$t1, \$a1, 2 # \$t1 = low * 4
R13	[t5] = c	[004000bc] 00894820 add \$9, \$4, \$9 ; 57: add \$t1, \$a0, \$t1 # \$t1 = arr + low * 4
R14	[t6] = 1	[004000c0] 8d330000 lw \$19, 0(\$9) ; 58: lw \$s3, 0(\$t1) # \$s3 = arr[\$t1]
R15	[t7] = 38	[004000c4] 00065080 sll \$10, \$6, 2 ; 60: sll \$t2, \$a2, 2 # \$t2 = high * 4
R16	[s0] = 0	[004000c8] 00065020 add \$10, \$4, \$10 ; 61: add \$t2, \$a0, \$t2 # \$t2 = arr + high * 4
R17	[s1] = 0	[004000cc] 8d540000 lw \$20, 0(\$10) ; 62: lw \$s4, 0(\$t2) # \$s4 = arr[\$t2]
R18	[s2] = d	[004000d0] ad340000 sw \$20, 0(\$9) ; 64: sw \$s4, 0(\$t1) # arr[\$t1] = \$s4 (swap a[\$t1] and a[\$t2])
R19	[s3] = 38	[004000d4] 8faao000c sw \$10, 12(\$29) ; 65: sw \$s3, 0(\$t2) # arr[\$t2] = \$s3
R20	[s4] = 38	[004000d8] afa60008 addu \$17, \$0, \$5 ; 67: lw \$t2, 12(\$sp) # Load \$t2 on the stack
R21	[s5] = 0	[004000dc] 23bd0010 addi \$29, \$29, 16 ; 68: addi \$sp, \$sp, 16
R22	[s6] = 0	[004000e0] 03e00008 jr \$31 ; 69: jr \$ra
R23	[s7] = 0	[004000e4] 23bdfff0 addi \$29, \$29, -16 ; 71: addi \$sp, \$sp, -16 # Allocate space on the stack
R24	[t8] = 0	[004000e8] afa40000 sw \$4, 0(\$29) ; 73: sw \$a0, 0(\$sp) # Save array on the stack
R25	[t9] = 0	[004000ec] afa50004 sw \$5, 4(\$29) ; 74: sw \$a1, 4(\$sp) # Save low on the stack
R26	[k0] = 0	[004000f0] afa60008 sw \$6, 8(\$29) ; 75: sw \$a2, 8(\$sp) # Save high on the stack
R27	[k1] = 0	[004000f4] afbf000c sw \$31, 12(\$29) ; 76: sw \$ra, 12(\$sp) # Save \$ra on the stack
R28	[gp] = 10008000	[004000f8] 00058821 addu \$17, \$0, \$5 ; 78: move \$s1, \$a1 # \$s1 = \$a1
R29	[sp] = 7ffff16c	[004000fc] 00069021 addu \$18, \$0, \$6 ; 79: move \$s2, \$a2 # \$s2 = \$a2
R30	[s8] = 0	[00400100] 00124880 sll \$9, \$18, 2 ; 81: sll \$t1, \$s2, 2 # \$t1 = high * 4
R31	[ra] = 400164	[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]		Text
R2	[v0] = d	[00400090] 34240048 ori \$4, \$1, 72 [strLine]
R3	[v1] = 0	[00400094] 34020004 ori \$2, \$0, 4 ; 42: li \$v0, 4
R4	[a0] = 1001004c	[00400098] 00000000 syscall ; 43: syscall
R5	[a1] = d	[0040009c] 3402000a ori \$2, \$0, 10 ; 45: li \$v0, 10
R6	[a2] = d	[004000a0] 0000000c syscall ; 46: syscall
R7	[a3] = 0	[004000a4] 23bdfff0 addi \$29, \$29, -16 ; 49: addi \$sp, \$sp, -16 # Allocate space on the stack
R8	[t0] = d	[004000a8] afa40000 sw \$4, 0(\$29) ; 51: sw \$a0, 0(\$sp) # Store array
R9	[t1] = 10010080	[004000ac] afa50004 sw \$5, 4(\$29) ; 52: sw \$a1, 4(\$sp) # Store Low
R10	[t2] = 0	[004000cc] afa60008 sw \$6, 8(\$29) ; 53: sw \$a2, 8(\$sp) # Store High
R11	[t3] = c	[004000dc] afaaa000c sw \$10, 12(\$29) ; 54: sw \$t2, 12(\$sp) # Store t2 for high
R12	[t4] = d	[004000e0] 00054880 sll \$9, \$5, 2 ; 56: sll \$t1, \$a1, 2 # \$t1 = low * 4
R13	[t5] = c	[004000ec] 00894920 add \$9, \$4, \$9 ; 57: add \$t1, \$a0, \$t1 # \$t1 = arr + low * 4
R14	[t6] = 1	[004000c0] 8d330000 lw \$19, 0(\$9) ; 58: lw \$s3, 0(\$t1) # \$s3 = arr[\$t1]
R15	[t7] = 38	[004000c4] 00065080 sll \$10, \$6, 2 ; 60: sll \$t2, \$a2, 2 # \$t2 = high * 4
R16	[s0] = 0	[004000c8] 00065020 add \$10, \$4, \$10 ; 61: add \$t2, \$a0, \$t2 # \$t2 = arr + high * 4
R17	[s1] = 0	[004000cc] 8d540000 lw \$20, 0(\$10) ; 62: lw \$s4, 0(\$t2) # \$s4 = arr[\$t2]
R18	[s2] = d	[004000dc] ad340000 sw \$20, 0(\$9) ; 64: sw \$s4, 0(\$t1) # arr[\$t1] = \$s4 (swap a[\$t1] and a[\$t2])
R19	[s3] = 38	[004000e0] 8faao000c sw \$10, 12(\$29) ; 65: sw \$s3, 0(\$t2) # arr[\$t2] = \$s3
R20	[s4] = 38	[004000e4] afa60008 addu \$17, \$0, \$5 ; 67: lw \$t2, 12(\$sp) # Load \$t2 on the stack
R21	[s5] = 0	[004000e8] afbf000c sw \$31, 12(\$29) ; 68: addi \$sp, \$sp, 16
R22	[s6] = 0	[004000ec] 03e00008 jr \$31 ; 69: jr \$ra
R23	[s7] = 0	[004000f0] 23bdfff0 addi \$29, \$29, -16 ; 71: addi \$sp, \$sp, -16 # Allocate space on the stack
R24	[t8] = 0	[004000f4] afa40000 sw \$4, 0(\$29) ; 73: sw \$a0, 0(\$sp) # Save array on the stack
R25	[t9] = 0	[004000f8] afa50004 sw \$5, 4(\$29) ; 74: sw \$a1, 4(\$sp) # Save low on the stack
R26	[k0] = 0	[004000fc] afa60008 sw \$6, 8(\$29) ; 75: sw \$a2, 8(\$sp) # Save high on the stack
R27	[k1] = 0	[00400100] 00124880 sll \$9, \$18, 2 ; 76: sw \$ra, 12(\$sp) # Save \$ra on the stack
R28	[gp] = 10008000	[00400104] 00894820 add \$9, \$4, \$9 ; 78: move \$s1, \$a1 # \$s1 = \$a1
R29	[sp] = 7ffff16c	[00400108] 8d2a0000 lw \$10, 0(\$9) ; 79: move \$s2, \$a2 # \$s2 = \$a2
R30	[s8] = 0	[0040010c] 00069021 addu \$18, \$0, \$6 ; 81: sll \$t1, \$s2, 2 # \$t1 = high * 4
R31	[ra] = 400164	[00400110] 00894820 add \$9, \$4, \$9 ; 82: add \$t1, \$a0, \$t1 # pivot = t2 = arr[high]

Int Regs [16]		Text
R2	[v0] = d	[00400090] 34240048 ori \$4, \$1, 72 [strLine]
R3	[v1] = 0	[00400094] 34020004 ori \$2, \$0, 4 ; 42: li \$v0, 4
R4	[a0] = 1001004c	[00400098] 00000000 syscall ; 43: syscall
R5	[a1] = d	[0040009c] 3402000a ori \$2, \$0, 10 ; 45: li \$v0, 10
R6	[a2] = d	[004000a0] 0000000c syscall ; 46: syscall
R7	[a3] = 0	[004000a4] 23bdfff0 addi \$29, \$29, -16 ; 49: addi \$sp, \$sp, -16 # Allocate space on the stack
R8	[t0] = d	[004000a8] afa40000 sw \$4, 0(\$29) ; 51: sw \$a0, 0(\$sp) # Store array
R9	[t1] = 10010080	[004000ac] afa50004 sw \$5, 4(\$29) ; 52: sw \$a1, 4(\$sp) # Store Low
R10	[t2] = 0	[004000cc] afa60008 sw \$6, 8(\$29) ; 53: sw \$a2, 8(\$sp) # Store High
R11	[t3] = c	[004000dc] afaaa000c sw \$10, 12(\$29) ; 54: sw \$t2, 12(\$sp) # Store t2 for high
R12	[t4] = d	[004000e0] 00054880 sll \$9, \$5, 2 ; 56: sll \$t1, \$a1, 2 # \$t1 = low * 4
R13	[t5] = c	[004000ec] 00894920 add \$9, \$4, \$9 ; 57: add \$t1, \$a0, \$t1 # \$t1 = arr + low * 4
R14	[t6] = 1	[004000c0] 8d330000 lw \$19, 0(\$9) ; 58: lw \$s3, 0(\$t1) # \$s3 = arr[\$t1]
R15	[t7] = 38	[004000c4] 00065080 sll \$10, \$6, 2 ; 60: sll \$t2, \$a2, 2 # \$t2 = high * 4
R16	[s0] = 0	[004000c8] 00065020 add \$10, \$4, \$10 ; 61: add \$t2, \$a0, \$t2 # \$t2 = arr + high * 4
R17	[s1] = 0	[004000cc] 8d540000 lw \$20, 0(\$10) ; 62: lw \$s4, 0(\$t2) # \$s4 = arr[\$t2]
R18	[s2] = d	[004000dc] ad340000 sw \$20, 0(\$9) ; 64: sw \$s4, 0(\$t1) # arr[\$t1] = \$s4 (swap a[\$t1] and a[\$t2])
R19	[s3] = 38	[004000e0] 8faao000c sw \$10, 12(\$29) ; 65: sw \$s3, 0(\$t2) # arr[\$t2] = \$s3
R20	[s4] = 38	[004000e4] afa60008 addu \$17, \$0, \$5 ; 67: lw \$t2, 12(\$sp) # Load \$t2 on the stack
R21	[s5] = 0	[004000e8] afbf000c sw \$31, 12(\$29) ; 68: addi \$sp, \$sp, 16
R22	[s6] = 0	[004000ec] 03e00008 jr \$31 ; 69: jr \$ra
R23	[s7] = 0	[004000f0] 23bdfff0 addi \$29, \$29, -16 ; 71: addi \$sp, \$sp, -16 # Allocate space on the stack
R24	[t8] = 0	[004000f4] afa40000 sw \$4, 0(\$29) ; 73: sw \$a0, 0(\$sp) # Save array on the stack
R25	[t9] = 0	[004000f8] afa50004 sw \$5, 4(\$29) ; 74: sw \$a1, 4(\$sp) # Save low on the stack
R26	[k0] = 0	[004000fc] afa60008 sw \$6, 8(\$29) ; 75: sw \$a2, 8(\$sp) # Save high on the stack
R27	[k1] = 0	[00400100] 00124880 sll \$9, \$18, 2 ; 76: sw \$ra, 12(\$sp) # Save \$ra on the stack
R28	[gp] = 10008000	[00400104] 00894820 add \$9, \$4, \$9 ; 78: move \$s1, \$a1 # \$s1 = \$a1
R29	[sp] = 7ffff16c	[00400108] 8d2a0000 lw \$10, 0(\$9) ; 79: move \$s2, \$a2 # \$s2 = \$a2
R30	[s8] = 0	[0040010c] 00069021 addu \$18, \$0, \$6 ; 81: sll \$t1, \$s2, 2 # \$t1 = high * 4
R31	[ra] = 400164	[00400110] 00894820 add \$9, \$4, \$9 ; 82: add \$t1, \$a0, \$t1 # pivot = t2 = arr[high]

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] = 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

[004000090] 34240048 ori $4, $1, 72 [strLine]
[004000941] 34020004 ori $2, $0, 4
[004000981] 0000000c syscall
[0040009c1] 3402000a ori $2, $0, 10
[004000a01] 0000000c syscall
[004000a41] 23bdfff0 addi $29, $29, -16
[004000a81] afa40000 sw $4, 0($29)
[004000b41] afaaa00c sw $10, 12($29)
[004000b81] 00054880 sll $9, $5, 2
[004000bc1] 00894820 add $9, $4, $9
[004000c01] 8d330000 lw $19, 0($9)
[004000c41] 00065080 sll $10, $6, 2
[004000c81] 008a5020 add $10, $4, $10
[004000d01] 8d540000 lw $20, 0($10)
[004000d41] ad340000 sw $20, 0($9)
[004000d81] ad530000 sw $19, 0($10)
[004000e01] 03e00008 00058821 addu $17, $0, $5
[004000e41] afa50004 sw $5, 4($29)
[004000e81] afa60008 sw $6, 8($29)
[004000f01] afa60008 sw $6, 8($29)
[004000f41] afb000c sw $31, 12($29)
[004000f81] 00058821 addu $17, $0, $5
[004000fc1] 00069021 addu $18, $0, $6
[004001001] 00124880 sll $9, $18, 2
[004001041] 00894820 add $9, $4, $9
[004001081] 8d2a0000 lw $10, 0($9)

; 42: li $v0, 4
; 43: syscall
; 45: li $v0, 10
; 46: syscall
; 49: addi $sp, $sp, -16 # Allocate space on the stack
; 51: sw $a0, 0($sp) # Store array
; 52: sw $a1, 4($sp) # Store Low
; 53: sw $a2, 8($sp) # Store High
; 54: sw $t2, 12($sp) # Store t2 for high
; 56: sll $t1, $a0, 2 # $t1 = Low * 4
; 57: add $t1, $a0, $t1 # $t1 = arr + Low * 4
; 58: lw $s3, 0($t1) # $s3 = arr[$t1]
; 60: sll $t2, $a2, 2 # $t2 = High * 4
; 61: add $t2, $a0, $t2 # $t2 = arr + High * 4
; 62: lw $s4, 0($t2) # $s4 = arr[$t2]
; 64: sw $s4, 0($t1) # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
; 65: sw $s3, 0($t2) # arr[$t2] = $s3
; 67: lw $t2, 12($sp) # Load $t2 on the stack
; 68: addi $sp, $sp, 16
; 69: jr $ra
; 71: addi $sp, $sp, -16 # Allocate space on the stack
; 73: sw $a0, 0($sp) # Save array on the stack
; 74: sw $a1, 4($sp) # Save low on the stack
; 75: sw $a2, 8($sp) # Save high on the stack
; 76: sw $ra, 12($sp) # Save $ra on the stack
; 78: move $s1, $a1 # $s1 = $a1
; 79: move $s2, $a2 # $s2 = $a2
; 81: sll $t1, $s2, 2 # $t1 = high * 4
; 82: add $t1, $a0, $t1 # $t1 = arr + high * 4
; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
```

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] = 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

[004000090] 34240048 ori $4, $1, 72 [strLine]
[004000941] 34020004 ori $2, $0, 4
[004000981] 0000000c syscall
[0040009c1] 3402000a ori $2, $0, 10
[004000a01] 0000000c syscall
[004000a41] 23bdfff0 addi $29, $29, -16
[004000a81] afa40000 sw $4, 0($29)
[004000b41] afaaa00c sw $10, 12($29)
[004000b81] 00054880 sll $9, $5, 2
[004000bc1] 00894820 add $9, $4, $9
[004000c01] 8d330000 lw $19, 0($9)
[004000c41] 00065080 sll $10, $6, 2
[004000c81] 008a5020 add $10, $4, $10
[004000d01] 8d540000 lw $20, 0($10)
[004000d41] ad340000 sw $20, 0($9)
[004000d81] ad530000 sw $19, 0($10)
[004000e01] 03e00008 00058821 addu $17, $0, $5
[004000e41] afa50004 sw $5, 4($29)
[004000e81] afa60008 sw $6, 8($29)
[004000f01] afa60008 sw $6, 8($29)
[004000f41] afb000c sw $31, 12($29)
[004000f81] 00058821 addu $17, $0, $5
[004000fc1] 00069021 addu $18, $0, $6
[004001001] 00124880 sll $9, $18, 2
[004001041] 00894820 add $9, $4, $9
[004001081] 8d2a0000 lw $10, 0($9)

; 42: li $v0, 4
; 43: syscall
; 45: li $v0, 10
; 46: syscall
; 49: addi $sp, $sp, -16 # Allocate space on the stack
; 51: sw $a0, 0($sp) # Store array
; 52: sw $a1, 4($sp) # Store Low
; 53: sw $a2, 8($sp) # Store High
; 54: sw $t2, 12($sp) # Store t2 for high
; 56: sll $t1, $a0, 2 # $t1 = Low * 4
; 57: add $t1, $a0, $t1 # $t1 = arr + Low * 4
; 58: lw $s3, 0($t1) # $s3 = arr[$t1]
; 60: sll $t2, $a2, 2 # $t2 = High * 4
; 61: add $t2, $a0, $t2 # $t2 = arr + High * 4
; 62: lw $s4, 0($t2) # $s4 = arr[$t2]
; 64: sw $s4, 0($t1) # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
; 65: sw $s3, 0($t2) # arr[$t2] = $s3
; 67: lw $t2, 12($sp) # Load $t2 on the stack
; 68: addi $sp, $sp, 16
; 69: jr $ra
; 71: addi $sp, $sp, -16 # Allocate space on the stack
; 73: sw $a0, 0($sp) # Save array on the stack
; 74: sw $a1, 4($sp) # Save low on the stack
; 75: sw $a2, 8($sp) # Save high on the stack
; 76: sw $ra, 12($sp) # Save $ra on the stack
; 78: move $s1, $a1 # $s1 = $a1
; 79: move $s2, $a2 # $s2 = $a2
; 81: sll $t1, $s2, 2 # $t1 = high * 4
; 82: add $t1, $a0, $t1 # $t1 = arr + high * 4
; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
```

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] = 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

[004000090] 34240048 ori $4, $1, 72 [strLine]
[004000941] 34020004 ori $2, $0, 4
[004000981] 0000000c syscall
[0040009c1] 3402000a ori $2, $0, 10
[004000a01] 0000000c syscall
[004000a41] 23bdfff0 addi $29, $29, -16
[004000a81] afa40000 sw $4, 0($29)
[004000b41] afaaa00c sw $10, 12($29)
[004000b81] 00054880 sll $9, $5, 2
[004000bc1] 00894820 add $9, $4, $9
[004000c01] 8d330000 lw $19, 0($9)
[004000c41] 00065080 sll $10, $6, 2
[004000c81] 008a5020 lw $20, 0($10)
[004000d01] ad340000 sw $20, 0($9)
[004000d41] ad530000 sw $19, 0($10)
[004000d81] 00058821 addu $17, $0, $5
[004000e01] 03e00008 00058821 addu $17, $0, $5
[004000e41] afa50004 sw $5, 4($29)
[004000e81] 00065080 sll $10, $6, 2
[004000f01] 008a5020 add $10, $4, $10
[004000f41] 8d540000 lw $20, 0($10)
[004000f81] 00058821 addu $17, $0, $5
[004000fc1] 00069021 addu $18, $0, $6
[004001001] 00124880 sll $9, $18, 2
[004001041] 00894820 add $9, $4, $9
[004001081] 8d2a0000 lw $10, 0($9)

; 42: li $v0, 4
; 43: syscall
; 45: li $v0, 10
; 46: syscall
; 49: addi $sp, $sp, -16 # Allocate space on the stack
; 51: sw $a0, 0($sp) # Store array
; 52: sw $a1, 4($sp) # Store Low
; 53: sw $a2, 8($sp) # Store High
; 54: sw $t2, 12($sp) # Store t2 for high
; 56: sll $t1, $a0, 2 # $t1 = Low * 4
; 57: add $t1, $a0, $t1 # $t1 = arr + Low * 4
; 58: lw $s3, 0($t1) # $s3 = arr[$t1]
; 60: sll $t2, $a2, 2 # $t2 = High * 4
; 61: add $t2, $a0, $t2 # $t2 = arr + High * 4
; 62: lw $s4, 0($t2) # $s4 = arr[$t2]
; 64: sw $s4, 0($t1) # arr[$t1] = $s4 (swap a[$t1] and a[$t2])
; 65: sw $s3, 0($t2) # arr[$t2] = $s3
; 67: lw $t2, 12($sp) # Load $t2 on the stack
; 68: addi $sp, $sp, 16
; 69: jr $ra
; 71: addi $sp, $sp, -16 # Allocate space on the stack
; 73: sw $a0, 0($sp) # Save array on the stack
; 74: sw $a1, 4($sp) # Save low on the stack
; 75: sw $a2, 8($sp) # Save high on the stack
; 76: sw $ra, 12($sp) # Save $ra on the stack
; 78: move $s1, $a1 # $s1 = $a1
; 79: move $s2, $a2 # $s2 = $a2
; 81: sll $t1, $s2, 2 # $t1 = high * 4
; 82: add $t1, $a0, $t1 # $t1 = arr + high * 4
; 83: lw $t2, 0($t1) # pivot = t2 = arr[high]
```

Int Regs [16]		Text
R2	[v0] = d	[004000090] 34240048 ori \$4, \$1, 72 [strLine]
R3	[v1] = 0	[004000094] 34020004 ori \$2, \$0, 4 ; 42: li \$v0, 4
R4	[a0] = 1001004c	[004000098] 0000000c syscall ; 43: syscall
R5	[a1] = d	[00400009c] 3402000a ori \$2, \$0, 10 ; 45: li \$v0, 10
R6	[a2] = d	[0040000a0] 0000000c syscall ; 46: syscall
R7	[a3] = 0	[0040000a4] 23bffff0 addi \$29, \$29, -16 ; 49: addi \$sp, \$sp, -16 # Allocate space on the stack
R8	[t0] = d	[0040000a8] afa40000 sw \$4, 0(\$29) ; 51: sw \$a0, 0(\$sp) # Store array
R9	[t1] = 10010080	[0040000ac] afa50004 sw \$5, 4(\$29) ; 52: sw \$a1, 4(\$sp) # Store Low
R10	[t2] = 0	[0040000b0] afa60008 sw \$6, 8(\$29) ; 53: sw \$a2, 8(\$sp) # Store High
R11	[t3] = c	[0040000b4] afaa000c sw \$10, 12(\$29) ; 54: sw \$t2, 12(\$sp) # Store t2 for high
R12	[t4] = d	[0040000b8] 00054880 sll \$9, \$5, 2 ; 56: sll \$t1, \$a1, 2 # \$t1 = Low * 4
R13	[t5] = c	[0040000bc] 00894820 add \$9, \$4, \$9 ; 57: add \$t1, \$a0, \$t1 # \$t1 = arr + Low * 4
R14	[t6] = 1	[0040000c0] 8d330000 lw \$19, 0(\$9) ; 58: lw \$s3, 0(\$t1) # \$s3 = arr[\$t1]
R15	[t7] = 38	[0040000c4] 00065080 lw \$10, \$6, 2 ; 60: sll \$t2, \$a2, 2 # \$t2 = High * 4
R16	[s0] = 0	[0040000c8] 008a5020 add \$10, \$4, \$10 ; 61: add \$t2, \$a0, \$t2 # \$t2 = arr + High * 4
R17	[s1] = 0	[0040000cc] 8d540000 lw \$20, 0(\$10) ; 62: lw \$s4, 0(\$t2) # \$s4 = arr[\$t2]
R18	[s2] = d	[0040000d0] ad340000 sw \$20, 0(\$9) ; 64: sw \$s4, 0(\$t1) # arr[\$t1] = \$s4 (swap a[\$t1] and a[\$t2])
R19	[s3] = 0	[0040000d4] ad530000 sw \$19, 0(\$10) ; 65: sw \$s3, 0(\$t2) # arr[\$t2] = \$s3
R20	[s4] = 0	[0040000d8] 8faa000c lw \$10, 12(\$29) ; 67: lw \$t2, 12(\$sp) # Load \$t2 on the stack
R21	[s5] = 0	[0040000dc] 23bd0010 addi \$29, \$29, 16 ; 68: addi \$sp, \$sp, 16
R22	[s6] = 0	[0040000e0] 03e00008 jr \$31 ; 69: jr \$ra
R23	[s7] = 0	[0040000e4] 23bffff0 addi \$29, \$29, -16 ; 71: addi \$sp, \$sp, -16 # Allocate space on the stack
R24	[t8] = 0	[0040000e8] afa40000 sw \$4, 0(\$29) ; 73: sw \$a0, 0(\$sp) # Save array on the stack
R25	[t9] = 0	[0040000ec] afa50004 sw \$5, 4(\$29) ; 74: sw \$a1, 4(\$sp) # Save low on the stack
R26	[k0] = 0	[0040000f0] afa60008 sw \$6, 8(\$29) ; 75: sw \$a2, 8(\$sp) # Save high on the stack
R27	[k1] = 0	[0040000f4] afbff00c sw \$31, 12(\$29) ; 76: sw \$ra, 12(\$sp) # Save \$ra on the stack
R28	[gp] = 10008000	[0040000f8] 00058821 addu \$17, \$0, \$5 ; 78: move \$s1, \$a1 # \$s1 = \$a1
R29	[sp] = 7fffff17c	[0040000fc] 00069021 addu \$18, \$0, \$6 ; 79: move \$s2, \$a2 # \$s2 = \$a2
R30	[s8] = 0	[00400100] 00124880 sll \$9, \$18, 2 ; 81: sll \$t1, \$s2, 2 # \$t1 = high * 4
R31	[ra] = 400164	[00400104] 00894820 add \$9, \$4, \$9 ; 82: add \$t1, \$a0, \$t1 # \$t1 = arr + high * 4
		[004001081] 8d2a0000 lw \$10, 0(\$9) ; 83: lw \$t2, 0(\$t1) # pivot = t2 = arr[high]

Int Regs [16]		Text
R2	[v0] = d	[004000090] 34240048 ori \$4, \$1, 72 [strLine]
R3	[v1] = 0	[004000094] 34020004 ori \$2, \$0, 4 ; 42: li \$v0, 4
R4	[a0] = 1001004c	[004000098] 0000000c syscall ; 43: syscall
R5	[a1] = d	[00400009c] 3402000a ori \$2, \$0, 10 ; 45: li \$v0, 10
R6	[a2] = d	[0040000a0] 0000000c syscall ; 46: syscall
R7	[a3] = 0	[0040000a4] 23bffff0 addi \$29, \$29, -16 ; 49: addi \$sp, \$sp, -16 # Allocate space on the stack
R8	[t0] = d	[0040000a8] afa40000 sw \$4, 0(\$29) ; 51: sw \$a0, 0(\$sp) # Store array
R9	[t1] = 10010080	[0040000ac] afa50004 sw \$5, 4(\$29) ; 52: sw \$a1, 4(\$sp) # Store Low
R10	[t2] = 0	[0040000b0] afa60008 sw \$6, 8(\$29) ; 53: sw \$a2, 8(\$sp) # Store High
R11	[t3] = c	[0040000b4] afaa000c sw \$10, 12(\$29) ; 54: sw \$t2, 12(\$sp) # Store t2 for high
R12	[t4] = d	[0040000b8] 00054880 sll \$9, \$5, 2 ; 56: sll \$t1, \$a1, 2 # \$t1 = Low * 4
R13	[t5] = c	[0040000bc] 00894820 add \$9, \$4, \$9 ; 57: add \$t1, \$a0, \$t1 # \$t1 = arr + Low * 4
R14	[t6] = 1	[0040000c0] 8d330000 lw \$19, 0(\$9) ; 58: lw \$s3, 0(\$t1) # \$s3 = arr[\$t1]
R15	[t7] = 38	[0040000c4] 00065080 lw \$10, \$6, 2 ; 60: sll \$t2, \$a2, 2 # \$t2 = High * 4
R16	[s0] = 0	[0040000c8] 008a5020 add \$10, \$4, \$10 ; 61: add \$t2, \$a0, \$t2 # \$t2 = arr + High * 4
R17	[s1] = 0	[0040000cc] 8d540000 lw \$20, 0(\$10) ; 62: lw \$s4, 0(\$t2) # \$s4 = arr[\$t2]
R18	[s2] = d	[0040000d0] ad340000 sw \$20, 0(\$9) ; 64: sw \$s4, 0(\$t1) # arr[\$t1] = \$s4 (swap a[\$t1] and a[\$t2])
R19	[s3] = 0	[0040000d4] ad530000 sw \$19, 0(\$10) ; 65: sw \$s3, 0(\$t2) # arr[\$t2] = \$s3
R20	[s4] = 0	[0040000d8] 8faa000c lw \$10, 12(\$29) ; 67: lw \$t2, 12(\$sp) # Load \$t2 on the stack
R21	[s5] = 0	[0040000dc] 23bd0010 addi \$29, \$29, 16 ; 68: addi \$sp, \$sp, 16
R22	[s6] = 0	[0040000e0] 03e00008 jr \$31 ; 69: jr \$ra
R23	[s7] = 0	[0040000e4] 23bffff0 addi \$29, \$29, -16 ; 71: addi \$sp, \$sp, -16 # Allocate space on the stack
R24	[t8] = 0	[0040000e8] afa40000 sw \$4, 0(\$29) ; 73: sw \$a0, 0(\$sp) # Save array on the stack
R25	[t9] = 0	[0040000ec] afa50004 sw \$5, 4(\$29) ; 74: sw \$a1, 4(\$sp) # Save low on the stack
R26	[k0] = 0	[0040000f0] afa60008 sw \$6, 8(\$29) ; 75: sw \$a2, 8(\$sp) # Save high on the stack
R27	[k1] = 0	[0040000f4] afbff00c sw \$31, 12(\$29) ; 76: sw \$ra, 12(\$sp) # Save \$ra on the stack
R28	[gp] = 10008000	[0040000f8] 00058821 addu \$17, \$0, \$5 ; 78: move \$s1, \$a1 # \$s1 = \$a1
R29	[sp] = 7fffff17c	[0040000fc] 00069021 addu \$18, \$0, \$6 ; 79: move \$s2, \$a2 # \$s2 = \$a2
R30	[s8] = 0	[00400100] 00124880 sll \$9, \$18, 2 ; 81: sll \$t1, \$s2, 2 # \$t1 = high * 4
R31	[ra] = 400164	[00400104] 00894820 add \$9, \$4, \$9 ; 82: add \$t1, \$a0, \$t1 # \$t1 = arr + high * 4
		[004001081] 8d2a0000 lw \$10, 0(\$9) ; 83: lw \$t2, 0(\$t1) # pivot = t2 = arr[high]

Int Regs [16]		Text
R2	[v0] = d	[00400110] 00116021 addu \$12, \$0, \$17 ; 86: move \$t4, \$s1 # \$t4 = j = low
R3	[v1] = 0	[00400114] 224dffff addi \$13, \$18, -1 ; 87: addi \$t5, \$s2, -1 # \$t5 = high - 1
R4	[a0] = 1001004c	[00400118] 01ac702a slt \$14, \$13, \$12 ; 90: slt \$t6, \$t5, \$t4 # Check \$t6 = 1 if j > high - 1
R5	[a1] = d	[0040011c] 15c0000e bne \$14, \$0, \$56 [endifor-0x0040011c]
R6	[a2] = d	[00400120] 000c4880 sll \$9, \$12, 2 ; 94: sll \$t1, \$t4, 2 # \$t1 = low * 4
R7	[a3] = 0	[00400124] 01244820 add \$9, \$9, \$4 ; 95: add \$t1, \$t1, \$a0 # \$t1 = arr + low * 4
R8	[t0] = d	[00400128] 8d2f0000 lw \$15, \$0(\$9) ; 96: lw \$t7, \$0(\$t1) # \$t7 = arr[j] = arr[low]
R9	[t1] = 10010080	[0040012c] 01eac02a slt \$24, \$15, \$10 ; 98: slt \$t8, \$t7, \$t2 # Check \$t8 = 1 if arr[j] > pivot
R10	[t2] = 0	[00400130] 17000007 bne \$24, \$0, \$28 [endifor-0x00400130]
R11	[t3] = c	[00400134] 216b0001 addi \$11, \$11, 1 ; 102: addi \$t3, \$t3, 1 # Increase it++
R12	[t4] = d	[00400138] 000b2821 addu \$5, \$0, \$11 ; 103: move \$s1, \$t3
R13	[t5] = c	[0040013c] 000c3021 addu \$6, \$0, \$12 ; 104: move \$s2, \$t4
R14	[t6] = 1	[00400140] 01c00029 jal 0x0040004a [SWAP] ; 105: jal SWAP
R15	[t7] = 38	[00400144] 218c0001 addi \$12, \$12, 1 ; 107: addi \$t4, \$t4, 1 # j++
R16	[s0] = 0	[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
R17	[s1] = 0	[0040014c] 218c0001 addi \$12, \$12, 1 ; 111: addi \$t4, \$t4, 1 # j++
R18	[s2] = d	[00400150] 08100046 j 0x00400118 [forloop] ; 112: j forloop
R19	[s3] = 0	[00400154] 21650001 addi \$5, \$11, 1 ; 115: addi \$s1, \$t3, 1 # \$s1 = i + 1
R20	[s4] = 0	[00400158] 00123021 addu \$6, \$0, \$18 ; 116: move \$s2, \$s2 # \$s2 = high
R21	[s5] = 0	[0040015c] 00051020 add \$2, \$0, \$5 ; 117: add \$t0, \$zero, \$s1 # \$t0 = \$s1
R22	[s6] = 0	[00400160] 0c100029 jal 0x00400004 [SWAP] ; 118: jal SWAP # Swap(arr[i-1], arr[high])
R23	[s7] = 0	[00400164] 8fbff00c lw \$s1, 12(\$29) ; 120: lw \$ra, 12(\$sp) # Return address
R24	[t8] = 0	[00400168] 23bd0010 addi \$29, \$29, 16 ; 121: addi \$sp, \$sp, 16 # Restore the stack
R25	[t9] = 0	[0040016c] 03e00008 jr \$31 ; 122: jr \$ra
R26	[k0] = 0	[00400170] 23bffff0 addi \$29, \$29, -16 # Allocate space on the stack
R27	[k1] = 0	[00400174] afa40000 sw \$4, 0(\$29) ; 125: addi \$sp, \$sp, -16 # save low
R28	[gp] = 10008000	[00400178] afa50004 sw \$5, 4(\$29) ; 128: sw \$a1, 4(\$sp) # save low
R29	[sp] = 7fffff17c	[0040017c] afa60008 sw \$6, 8(\$29) ; 129: sw \$a2, 8(\$sp) # save high
R30	[s8] = 0	[00400180] afbff00c sw \$31, 12(\$29) ; 130: sw \$ra, 12(\$sp) # Return address
R31	[ra] = 400194	[00400184] 00064021 addu \$8, \$0, \$6 ; 132: move \$t0, \$s2 # \$t0 = high
		[00400188] 00a8482a slt \$9, \$5, \$8 ; 134: sll \$t1, \$a1, \$t0 # Check \$t1 = 1 if low > high

Int Regs [16]

	Text
R2 [v0] = d	[00400110] 00116021 addu \$12, \$0, \$17 ; 86: move \$t4, \$s1 # t4 = j = low
R3 [v1] = 0	[00400114] 224dffff addi \$13, \$18, -1 ; 87: addi \$t5, \$s2, -1 # t5 = high - 1
R4 [a0] = 1001004c	[00400118] 01ac702a slt \$14, \$13, \$12 ; 90: slt \$t6, \$t5, \$t4 # Check \$t6 = 1 if j > high - 1
R5 [a1] = d	[0040011c] 15c0000a bne \$14, \$0, 56 [endifor-0x0040011c]
R6 [a2] = d	[00400120] 000c4880 sll \$9, \$12, 2 ; 94: sll \$t1, \$t4, 2 # \$t1 = low * 4
R7 [a3] = 0	[00400124] 01244820 add \$9, \$9, \$4 ; 95: add \$t1, \$t1, \$s0 # \$t1 = arr + low * 4
R8 [t0] = d	[00400128] 8d2f0000 lw \$15, 0(\$9) ; 96: lw \$t7, 0(\$t1) # \$t7 = arr[j] = arr[low]
R9 [t1] = 10010080	[0040012c] 01eac02a slt \$24, \$15, \$10 ; 98: slt \$t8, \$t7, \$t2 # Check \$t8 = 1 if arr[j] > pivot
R10 [t2] = 0	[00400130] 17000007 bne \$24, \$0, 28 [endifif-0x00400130]
R11 [t3] = c	[00400134] 21650001 addi \$11, \$11, 1 ; 102: addi \$t3, \$t3, 1 # Increase i++
R12 [t4] = d	[00400138] 000b2821 addu \$5, \$0, \$11 ; 103: move \$a1, \$t3
R13 [t5] = c	[0040013c] 000c3021 addu \$6, \$0, \$12 ; 104: move \$a2, \$t4
R14 [t6] = 1	[00400140] 01230201 jal 0x004000a4 [SWAP] ; 105: jal SWAP
R15 [t7] = 38	[00400144] 218c0001 addi \$12, \$12, 1 ; 107: addi \$t4, \$t4, 1 # j++
R16 [s0] = 0	[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
R17 [s1] = 0	[0040014c] 218c0001 addi \$12, \$12, 1 ; 111: addi \$t4, \$t4, 1 # j++
R18 [s2] = d	[00400150] 08100046 j 0x00400118 [forloop] ; 112: j forloop
R19 [s3] = 0	[00400154] 21650001 addi \$5, \$11, 1 ; 115: addi \$a1, \$t3, 1 # \$a1 = i + 1
R20 [s4] = 0	[00400158] 00123021 addu \$6, \$0, \$18 ; 116: move \$a2, \$s2 # \$a2 = high
R21 [s5] = 0	[0040015c] 00051020 add \$2, \$0, \$5 ; 117: add \$t0, \$zero, \$a1 # \$v0 = \$a1
R22 [s6] = 0	[00400160] 0c100029 jal 0x004000a4 [SWAP] ; 118: jal SWAP # Swap(arr[i+1], arr[high])
R23 [s7] = 0	[00400164] 8fbff000 lw \$31, 12(\$29) ; 120: lw \$ra, 12(\$sp) # Return address
R24 [t8] = 0	[00400168] 23bd0010 addi \$29, \$29, 16 ; 121: addi \$sp, \$sp, 16 # Restore the stack
R25 [t9] = 0	[0040016c] 03e00008 jr \$31 ; 122: jr \$ra
R26 [k0] = 0	[00400170] 23bffff0 addi \$29, \$29, -16 ; 125: addi \$sp, \$sp, -16 # Allocate space on the stack
R27 [k1] = 0	[00400174] afa40000 sw \$4, 0(\$29) ; 127: sw \$a0, 0(\$sp) # save arr
R28 [gp] = 10008000	[00400178] afa50004 sw \$5, 4(\$29) ; 128: sw \$a1, 4(\$sp) # save low
R29 [sp] = 7fffff18c	[0040017c] afa60008 sw \$6, 8(\$29) ; 129: sw \$a2, 8(\$sp) # save high
R30 [s8] = 0	[00400180] afbf0008 sw \$31, 12(\$29) ; 130: sw \$ra, 12(\$sp) # Return address
R31 [ra] = 400194	[00400184] 00064021 addu \$8, \$0, \$6 ; 132: move \$t0, \$a2 # \$t0 = high
	[00400188] 00a482a slt \$9, \$5, \$8 ; 134: slt \$t1, \$a1, \$t0 # Check \$t1 = 1 if low > high
	... 00000000 00000000 00000000 00000000-1

Int Regs [16]

	Text
R2 [v0] = d	[00400140] 0c100029 jal 0x004000a4 [SWAP] ; 105: jal SWAP
R3 [v1] = 0	[00400144] 218c0001 addi \$12, \$12, 1 ; 107: addi \$t4, \$t4, 1 # j++
R4 [a0] = 1001004c	[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
R5 [a1] = d	[0040014c] 218c0001 addi \$12, \$12, 1 ; 111: addi \$t4, \$t4, 1 # j++
R6 [a2] = d	[00400150] 08100046 j 0x00400118 [forloop] ; 112: j forloop
R7 [a3] = 0	[00400154] 21650001 addi \$5, \$11, 1 ; 115: addi \$a1, \$t3, 1 # \$a1 = i + 1
R8 [t0] = d	[00400158] 00123021 addu \$6, \$0, \$18 ; 116: move \$a2, \$s2 # \$a2 = high
R9 [t1] = 10010080	[0040015c] 00051020 add \$2, \$0, \$5 ; 117: add \$t0, \$zero, \$a1 # \$v0 = \$a1
R10 [t2] = 0	[00400160] 0c100029 jal 0x004000a4 [SWAP] ; 118: jal SWAP # Swap(arr[i+1], arr[high])
R11 [t3] = c	[00400164] 8fbff00c lw \$31, 12(\$29) ; 120: lw \$ra, 12(\$sp) # Return address
R12 [t4] = d	[00400168] 23bd0010 addi \$29, \$29, 16 ; 121: addi \$sp, \$sp, 16 # Restore the stack
R13 [t5] = c	[0040016c] 03e00008 jr \$31 ; 122: jr \$ra
R14 [t6] = 1	[00400170] 23bffff0 addi \$29, \$29, -16 ; 125: addi \$sp, \$sp, -16 # Allocate space on the stack
R15 [t7] = 38	[00400174] afa40000 sw \$4, 0(\$29) ; 127: sw \$a0, 0(\$sp) # save arr
R16 [s0] = d	[00400178] afa50004 sw \$5, 4(\$29) ; 128: sw \$a1, 4(\$sp) # save low
R17 [s1] = 0	[0040017c] afa60008 sw \$6, 8(\$29) ; 129: sw \$a2, 8(\$sp) # save high
R18 [s2] = d	[00400180] afbf0008 sw \$31, 12(\$29) ; 130: sw \$ra, 12(\$sp) # Return address
R19 [s3] = 0	[00400184] 00064021 addu \$8, \$0, \$6 ; 132: move \$t0, \$a2 # \$t0 = high
R20 [s4] = 0	[00400188] 00a482a slt \$9, \$5, \$8 ; 134: slt \$t1, \$a1, \$t0 # Check \$t1 = 1 if low > high
R21 [s6] = 0	[00400190] 0c100039 beq \$9, \$0, 36 [ENDIF-0x0040018c]
R22 [s7] = 0	[00400194] 00028021 addu \$16, \$0, \$2 ; 141: move \$s0, \$v0 # pivot \$s0 = \$v0
R23 [s8] = 0	[00400198] 8fa50004 lw \$5, 4(\$29) ; 143: lw \$a1, 4(\$sp) # load low
R24 [t8] = 0	[0040019c] 2206ffff add \$6, \$16, -1 ; 144: add \$a2, \$s0, -1 # pivot - 1
R25 [t9] = 0	[004001a0] 0c10005c jal 0x00400170 [QUICKSORT]; 145: jal QUICKSORT # quicksort(array, low, pivot - 1)
R26 [k0] = 0	[004001a4] 22050001 addi \$5, \$16, 1 ; 147: addi \$a1, \$s0, 1 # pivot + 1
R27 [k1] = 0	[004001a8] 8fa40000 lw \$6, 8(\$29) ; 148: lw \$a2, 8(\$sp) # load high
R28 [gp] = 10008000	[004001ac] 0c10005c jal 0x00400170 [QUICKSORT]; 149: jal QUICKSORT # quicksort(array, pivot + 1, high)
R29 [sp] = 7fffff18c	[004001b0] 8fa40000 lw \$4, 0(\$29) ; 153: lw \$a0, 0(\$sp) # Restore a0
R30 [s8] = 0	[004001b4] 8fa50004 lw \$5, 4(\$29) ; 154: lw \$a1, 4(\$sp) # Restore a1
R31 [ra] = 400194	[004001b8] 8fa60008 lw \$6, 8(\$29) ; 155: lw \$a2, 8(\$sp) # Restore a2
	... 00000000 00000000 00000000 00000000-1

Int Regs [16]

	Text
PC = 40019c	[00400140] 0c100029 jal 0x004000a4 [SWAP] ; 105: jal SWAP
EPC = 0	[00400144] 218c0001 addi \$12, \$12, 1 ; 107: addi \$t4, \$t4, 1 # j++
Cause = 0	[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
BadVAddr = 0	[0040014c] 01ac702a slt \$14, \$13, \$12 ; 111: addi \$t4, \$t4, 1 # j++
Status = 3000ff10	[00400150] 08100046 j 0x00400118 [forloop] ; 112: j forloop
HI = 0	[00400154] 21650001 addi \$5, \$11, 1 ; 115: addi \$a1, \$t3, 1 # \$a1 = i + 1
LO = 0	[00400158] 00123021 addu \$6, \$0, \$18 ; 116: move \$a2, \$s2 # \$a2 = high
R0 [r0] = 0	[0040015c] 00051020 add \$2, \$0, \$5 ; 117: add \$t0, \$zero, \$a1 # \$v0 = \$a1
R1 [at] = 10010000	[00400160] 0c100029 jal 0x004000a4 [SWAP] ; 118: jal SWAP # Swap(arr[i+1], arr[high])
R2 [v0] = d	[00400164] 8fbff00c lw \$31, 12(\$29) ; 120: lw \$ra, 12(\$sp) # Return address
R3 [v1] = 0	[00400168] 23bd0010 addi \$29, \$29, 16 ; 121: addi \$sp, \$sp, 16 # Restore the stack
R4 [a0] = 1001004c	[0040016c] 03e00008 jr \$31 ; 122: jr \$ra
R5 [a1] = 0	[00400170] 23bffff0 addi \$29, \$29, -16 ; 125: addi \$sp, \$sp, -16 # Allocate space on the stack
R6 [a2] = d	[00400174] afa40000 sw \$4, 0(\$29) ; 127: sw \$a0, 0(\$sp) # save arr
R7 [a3] = 0	[00400178] afa50004 sw \$5, 4(\$29) ; 128: sw \$a1, 4(\$sp) # save low
R8 [t0] = d	[0040017c] afa60008 sw \$6, 8(\$29) ; 129: sw \$a2, 8(\$sp) # save high
R9 [t1] = 10010080	[00400180] 00064021 addu \$8, \$0, \$6 ; 130: sw \$ra, 12(\$sp) # Return address
R10 [t2] = 0	[00400184] 00a482a slt \$9, \$5, \$8 ; 134: slt \$t1, \$a1, \$t0 # Check \$t1 = 1 if low > high
R11 [t3] = c	[00400188] 00028021 beq \$9, \$0, 36 [ENDIF-0x0040018c]
R12 [t4] = d	[00400190] 0c100039 jal 0x004000e4 [PARTITION]; 139: jal PARTITION
R13 [t5] = c	[00400194] 00028021 addu \$16, \$0, \$2 ; 141: move \$s0, \$v0 # pivot \$s0 = \$v0
R14 [t6] = 1	[00400198] 8fa50004 lw \$5, 4(\$29) ; 143: lw \$a1, 4(\$sp) # load low
R15 [t7] = 38	[0040019c] 2206ffff add \$6, \$16, -1 ; 144: add \$a2, \$s0, -1 # pivot - 1
R16 [s0] = d	[004001a0] 0c10005c jal 0x00400170 [QUICKSORT]; 145: jal QUICKSORT # quicksort(array, low, pivot - 1)
R17 [s1] = 0	[004001a4] 22050001 addi \$5, \$16, 1 ; 147: add \$a1, \$s0, 1 # pivot + 1
R18 [s2] = d	[004001a8] 8fa40000 lw \$6, 8(\$29) ; 148: lw \$a2, 8(\$sp) # load high
R19 [s3] = 0	[004001ac] 0c10005c jal 0x00400170 [QUICKSORT]; 149: jal QUICKSORT # quicksort(array, pivot + 1, high)
R20 [s4] = 0	[004001b0] 8fa40000 lw \$4, 0(\$29) ; 153: lw \$a0, 0(\$sp) # Restore a0
R21 [s5] = 0	[004001b4] 8fa50004 lw \$5, 4(\$29) ; 154: lw \$a1, 4(\$sp) # Restore a1
	... 00000000 00000000 00000000 00000000-1

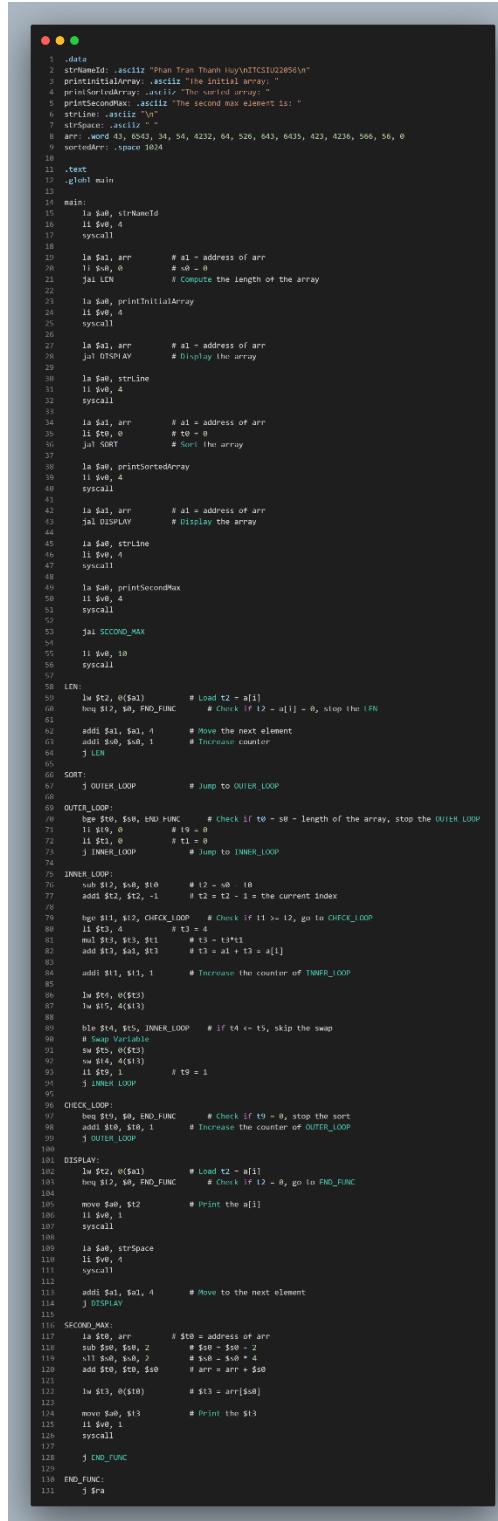
Reg [16]		Text
PC	= 4001a0	[00400140] 0c100029 jal 0x004000a4 [SWAP] ; 105: jal SWAP
EPC	= 0	[00400144] 218c0001 addi \$12, \$12, 1 ; 107: addi \$t4, \$t4, 1 # j++
Cause	= 0	[00400148] 08100046 j 0x00400118 [forloop] ; 108: j forloop
BadVAddr	= 0	[0040014c] 218c0001 addi \$12, \$12, 1 ; 111: addi \$t4, \$t4, 1 # j++
Status	= 3000ff10	[00400150] 08100046 j 0x00400118 [forloop] ; 112: j forloop
HI	= 0	[00400154] 21650001 addi \$5, \$11, 1 ; 115: addi \$a1, \$t3, 1 # \$a1 = i + 1
LO	= 0	[00400158] 00123021 addu \$6, \$0, \$18 ; 116: move \$a2, \$s2 # \$a2 = high
R0	[r0] = 0	[0040015c] 00051020 add \$2, \$0, \$5 ; 117: add \$v0, \$zero, \$a1 # \$v0 = \$a1
R1	[at] = 10010000	[00400160] 0c100025 jal 0x004000a4 [SWAP] ; 118: jal SWAP # Swap(\$arr[i+1], \$arr[high])
R2	[v0] = d	[00400164] 8fbff00c lw \$31, 12(\$29) ; 120: lw \$ra, 12(\$sp) # Return address
R3	[v1] = 0	[00400168] 23bd0010 addi \$29, \$29, 16 ; 121: addi \$sp, \$sp, 16 # Restore the stack
R4	[a0] = 1001004c	[0040016c] 03e00008 jr \$31 ; 122: jr \$ra
R5	[a1] = 0	[00400170] 23bdff0 addi \$29, \$29, -16 ; 125: addi \$sp, \$sp, -16 # Allocate space on the stack
R6	[a2] = c	[00400174] afa40000 sw \$4, 0(\$29) ; 127: sw \$a0, 0(\$sp) # save arr
R7	[a3] = 0	[00400178] afa50004 sw \$5, 4(\$29) ; 128: sw \$a1, 4(\$sp) # save low
R8	[t0] = d	[00400180] afbf000c sw \$31, 12(\$29) ; 129: sw \$a2, 8(\$sp) # save high
R9	[t1] = 10010080	[00400184] 00064021 addu \$8, \$0, \$6 ; 130: sw \$ra, 12(\$sp) # Return address
R10	[t2] = 0	[00400188] 00a4982a s1t \$9, \$5, \$8 ; 132: move \$t0, \$a2 # \$t0 = high
R11	[t3] = c	[00400190] 0c100039 beq \$9, \$0, 36 [ENDIF-0x0040018c] ; 134: s1t \$t1, \$a1, \$t0 # Check \$t1 = 1 if low > high
R12	[t4] = d	[00400194] 00028021 addu \$16, \$0, \$2 ; 139: jal PARTITION
R13	[t5] = c	[00400198] 8fa50004 lw \$5, 4(\$29) ; 141: move \$s0, \$v0 # pivot \$s0 = \$v0
R14	[t6] = 1	[0040019c] 2206ffff addi \$6, \$16, -1 ; 143: lw \$a1, 4(\$sp) # load low
R15	[t7] = 38	[004001a0] 0c10005c jal 0x00400170 [QUICKSORT] ; 144: addi \$a2, \$s0, -1 # pivot - 1
R16	[s0] = d	[004001a4] 22050001 addi \$5, \$16, 1 ; 147: addi \$a1, \$s0, 1 # pivot + 1
R17	[s1] = 0	[004001a8] 8fa60008 lw \$6, 8(\$29) ; 148: lw \$a2, 8(\$sp) # load high
R18	[s2] = d	[004001ac] 0c10005c jal 0x00400170 [QUICKSORT] ; 149: jal QUICKSORT # quicksort(array, pivot + 1, high)
R19	[s3] = 0	[004001b0] 8fa40000 lw \$4, 0(\$29) ; 153: lw \$a0, 0(\$sp) # Restore a0
R20	[s4] = 0	[004001b4] 8fa50004 lw \$5, 4(\$29) ; 154: lw \$a1, 4(\$sp) # Restore a1
R21	[s5] = 0	[004001b8] 8fa60008 lw \$6, 8(\$29) ; 155: lw \$a2, 8(\$sp) # Restore a2
		... 00000000-00000000 ... 0000-10/0001 # previous return address

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

And QuickSort (Array, Pivot + 1, High)

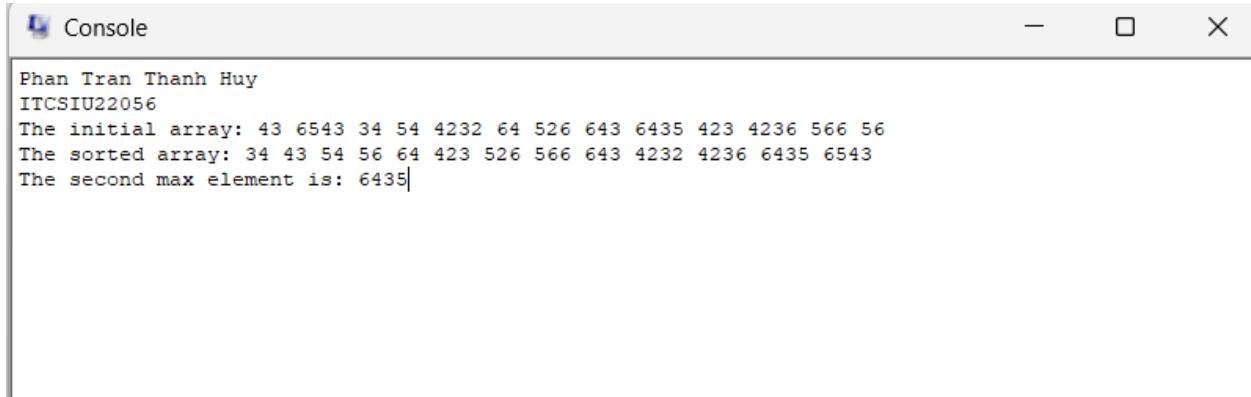
## Exercise 3:

### Code:



```
1 .data
2 strNameId: .ascii "Phan Tran Thanh Huy\nnITCSU2020\n"
3 printInitialArray: .ascii "The initial array: "
4 printSecondMax: .ascii "The sorted array: "
5 printSecondMaxX: .ascii "The second max element is: "
6 strTime: .ascii " "
7 strSpace: .ascii " "
8 arr .word 45, 6043, 34, 54, 4232, 64, 526, 643, 6435, 423, 4235, 566, 56, 0
9 sortedArr .space 3034
10
11 .text
12 .globl main
13
14 main:
15 la $a0, strNameId
16 li $v0, 4
17 syscall
18
19 la $a1, arr      # $a1 = address of arr
20 li $v0, 0          # $v0 = 0
21 jal LCN          # Compute the length of the array
22
23 la $a0, printInitialArray
24 li $v0, 4
25 syscall
26
27 la $a1, arr      # $a1 = address of arr
28 jal DSPLAY        # Display the array
29
30 la $a0, strTime
31 li $v0, 4
32 syscall
33
34 la $a1, arr      # $a1 = address of arr
35 li $v0, 0          # $v0 = 0
36 jal SORT          # Sort the array
37
38 la $a0, printSortedArray
39 li $v0, 4
40 syscall
41
42 la $a1, arr      # $a1 = address of arr
43 jal DISPLAY        # Display the array
44
45 la $a0, strTime
46 li $v0, 4
47 syscall
48
49 la $a0, printSecondMax
50 li $v0, 4
51 syscall
52
53 jal SECOND_MAX
54
55 li $v0, 10
56 syscall
57
58 LCN:
59     la $t2, 0($a1)      # Load t2 = a[i]
60     bne $t2, $0, END_FUNC    # Check if t2 = a[1] = 0, stop the LCN
61
62     addi $a1, $a1, 4      # Move the next element
63     addi $v0, $v0, 1      # Increase counter
64     j LCN
65
66 SORT:
67     j OUTER_LOOP        # Jump to OUTER_LOOP
68
69 OUTER_LOOP:
70     bge $t0, $v0, END_FUNC    # Check if t0 = v0 = length of the array, stop the OUTER_LOOP
71     li $v0, 0          # t0 = 0
72     li $t1, 0          # t1 = 0
73     j INNER_LOOP        # Jump to INNER_LOOP
74
75 INNER_LOOP:
76     sub $t2, $a0, $t0      # t2 = a0 - t0
77     addi $t2, $t2, -1      # t2 = t2 - 1 = the current index
78
79     bge $t1, $t2, CHECK_LOOP    # Check if t1 >= t2, go to CHECK_LOOP
80     li $t3, 4          # t3 = 4
81     mul $t3, $t3, $t1      # t3 = t3*t1
82     add $t3, $a0, $t3      # t3 = a1 + t3 = a[1]
83
84     addi $t1, $t1, 1      # Increase the counter of INNER_LOOP
85
86     la $t4, 0($t2)      # Load t4 = a[i]
87     la $t5, 0($t2)
88
89     ble $t4, $t5, INNER_LOOP    # If t4 <= t5, skip the swap
90     # Swap Variable
91     sw $t4, 0($t2)
92     sw $t5, 0($t2)
93     li $t5, 1          # t5 = 1
94     j INNER_LOOP        # Jump to INNER_LOOP
95
96 CHECK_LOOP:
97     bge $t0, $v0, END_FUNC    # Check if t0 = 0, stop the sort
98     addi $t0, $t0, 1      # Increase the counter of OUTER_LOOP
99     j OUTER_LOOP
100
101 DISPLAY:
102     la $t2, 0($a1)      # Load t2 = a[i]
103     bne $t2, $0, END_FUNC    # Check if t2 = 0, go to END_FUNC
104
105     move $a0, $t2      # Print the a[i]
106     li $v0, 1
107     syscall
108
109     la $a0, strSpace
110     li $v0, 4
111     syscall
112
113     addi $a1, $a1, 4      # Move to the next element
114     j DISPLAY
115
116 SECOND_MAX:
117     la $t0, arr      # $t0 = address of arr
118     sub $t0, $t0, 2      # $t0 = $t0 - 2
119     li $t1, 0          # t1 = 0
120     add $t0, $t0, $t0      # arr = arr + $t0
121
122     li $t3, 0($t0)      # $t3 = arr[$t0]
123
124     move $a0, $t3      # Print the $t3
125     li $v0, 1
126     syscall
127
128     j END_FUNC
129
130 END_FUNC:
131     j $ra
```

## Test case:



```

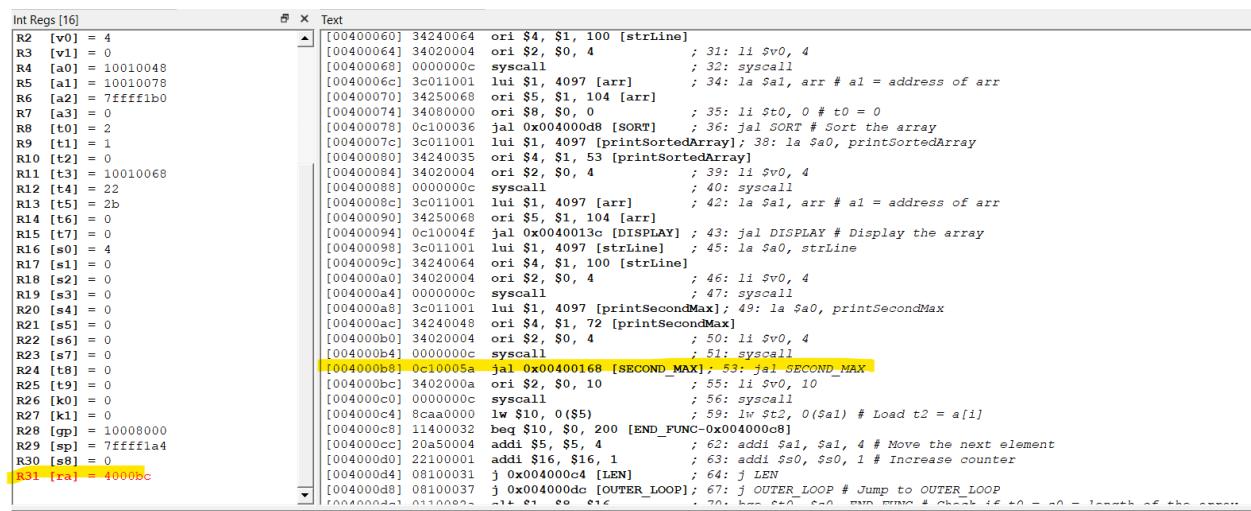
Phan Tran Thanh Huy
ITCSIU22056
The initial array: 43 6543 34 54 4232 64 526 643 6435 423 4236 566 56
The sorted array: 34 43 54 56 64 423 526 566 643 4232 4236 6435 6543
The second max element is: 6435

```

## Single Step:

It is the same steps with Bubble\_Sort

## SECOND\_MAX steps:



Int Regs [16]	Text
R2 [v0] = 4	[00400060] 34240064 ori \$4, \$1, 100 [strLine]
R3 [v1] = 0	[00400064] 34020004 ori \$2, \$0, 4 ; 31: li \$v0, 4
R4 [a0] = 10010048	[00400068] 0000000c syscall ; 32: syscall
R5 [a1] = 10010078	[0040006c] 3c011001 lui \$1, 4097 [arr] ; 34: la \$a1, arr # a1 = address of arr
R6 [a2] = 7fffff1b0	[00400070] 34250068 ori \$5, \$1, 104 [arr]
R7 [a3] = 0	[00400074] 34080000 ori \$8, \$0, 0 ; 35: li \$t0, 0 # t0 = 0
R8 [t0] = 2	[00400078] 0c100036 jal 0x004000d8 [SORT] ; 36: jal SORT # Sort the array
R9 [t1] = 1	[0040007c] 3c011001 lui \$1, 4097 [printSortedArray] ; 38: la \$a0, printSortedArray
R10 [t2] = 0	[00400080] 34240035 ori \$4, \$1, 53 [printSortedArray]
R11 [t3] = 10010068	[00400084] 34020004 ori \$2, \$0, 4 ; 39: li \$v0, 4
R12 [t4] = 22	[00400088] 0000000c syscall ; 40: syscall
R13 [t5] = 2b	[0040008c] 3c011001 lui \$1, 4097 [arr] ; 42: la \$a1, arr # a1 = address of arr
R14 [t6] = 0	[00400090] 34250068 ori \$5, \$1, 104 [arr]
R15 [t7] = 0	[00400094] 0c10004f jal 0x0040013c [DISPLAY] ; 43: jal DISPLAY # Display the array
R16 [s0] = 4	[00400098] 3c011001 lui \$1, 4097 [strLine] ; 45: la \$a0, strLine
R17 [s1] = 0	[0040009c] 34240064 ori \$4, \$1, 100 [strLine]
R18 [s2] = 0	[004000a0] 34020004 ori \$2, \$0, 4 ; 46: li \$v0, 4
R19 [s3] = 0	[004000a4] 0000000c syscall ; 47: syscall
R20 [s4] = 0	[004000a8] 3c011001 lui \$1, 4097 [printSecondMax] ; 49: la \$a0, printSecondMax
R21 [s5] = 0	[004000ac] 34240048 ori \$4, \$1, 72 [printSecondMax]
R22 [s6] = 0	[004000b0] 34020004 ori \$2, \$0, 4 ; 50: li \$v0, 4
R23 [s7] = 0	[004000b4] 0000000c syscall ; 51: syscall
R24 [t8] = 0	[004000b8] 0c10005a jal 0x00400168 [SECOND_MAX]; 53: jal SECOND_MAX
R25 [t9] = 0	[004000bc] 3402000a ori \$2, \$0, 10 ; 55: li \$v0, 10
R26 [k0] = 0	[004000c0] 0000000c syscall ; 56: syscall
R27 [k1] = 0	[004000c4] 8caa0000 lw \$10, 0(\$5) ; 59: lw \$t2, 0(\$a1) # Load t2 = a[i]
R28 [gp] = 10008000	[004000c8] 11400032 beq \$10, \$0, 200 [END_FUNC-0x004000c8]
R29 [sp] = 7fffff1a4	[004000cc] 20a50004 addi \$5, \$5, 4 ; 62: addi \$a1, \$a1, 4 # Move the next element
R30 [s8] = 0	[004000d0] 22100001 addi \$16, \$16, 1 ; 63: addi \$s0, \$s0, 1 # Increase counter
R31 [ra] = 40000bc	[004000d4] 08100031 j 0x004000c4 [LEN] ; 64: j LEN
	[004000d8] 08100037 j 0x004000dc [OUTER_LOOP]; 67: j OUTER_LOOP # Jump to OUTER_LOOP
	... 70: bcc \$t0, \$a0, END_FUNC # Check if t0 = a0 - length of the array

Int Regs [16]		Text
R2	[v0] = 4	[004000fc] 012a082a slt \$1, \$9, \$10 ; 79: bge \$t1, \$t2, CHECK_LOOP # Check if t1 >= t2, go to CHECK_LOOP
R3	[v1] = 0	[004000fc] 1020000d beq \$1, \$0, 52 [CHECK_LOOP-0x004000fc]
R4	[a0] = 10010048	[00400100] 340b0004 ori \$11, \$0, 4 ; 80: li \$t3, 4 # t3 = 4
R5	[a1] = 10010078	[00400104] 71695802 mul \$11, \$11, \$9 ; 81: mul \$t3, \$t3, \$t1 # t3 = t3*t1
R6	[a2] = 7ffff1b0	[00400108] 00ab5820 add \$11, \$5, \$11 ; 82: add \$t3, \$a1, \$t3 # t3 = a1 + t3 = a[i]
R7	[a3] = 0	[0040010c] 21290001 addi \$9, \$9, 1 ; 84: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R8	[t0] = 10010068	[00400110] 8d6c0000 lw \$12, 0(\$11) ; 86: lw \$t4, 0(\$t3)
R9	[t1] = 1	[00400114] 8d6d0004 lw \$13, 4(\$11) ; 87: lw \$t5, 4(\$t3)
R10	[t2] = 0	[00400118] 01ac082a slt \$1, \$13, \$12 ; 89: ble \$t4, \$t5, INNER_LOOP # if t4
R11	[t3] = 10010068	[0040011c] 1020ff55 beq \$1, \$0, -44 [INNER_LOOP-0x0040011c]
R12	[t4] = 22	[00400120] ad6d0000 sw \$13, 0(\$11) ; 91: sw \$t5, 0(\$t3)
R13	[t5] = 2b	[00400124] ad6c0004 sw \$12, 4(\$11) ; 92: sw \$t4, 4(\$t3)
R14	[t6] = 0	[00400128] 34190001 ori \$25, \$0, 1 ; 93: li \$t9, 1 # t9 = 1
R15	[t7] = 0	[0040012c] 0810003c j 0x004000f0 [INNER_LOOP]; 94: j INNER_LOOP
R16	[s0] = 4	[00400130] 13200018 beq \$25, \$0, 96 [END_FUNC-0x00400130]
R17	[s1] = 0	[00400134] 08100037 j 0x004000dc [OUTER_LOOP]; 99: j OUTER_LOOP
R18	[s2] = 0	[0040013c] 8caa0000 lw \$10, 0(\$5) ; 102: lw \$t2, 0(\$a1) # Load t2 = a[i]
R19	[s3] = 0	[00400140] 11400014 beq \$10, \$0, 80 [END_FUNC-0x00400140]
R20	[s4] = 0	[00400144] 00aa2021 addu \$4, \$0, \$10 ; 105: move \$a0, \$t2 # Print the a[i]
R21	[s5] = 0	[00400148] 34020001 ori \$2, \$0, 1 ; 106: li \$v0, 1
R22	[s6] = 0	[0040014c] 0000000c syscall ; 107: syscall
R23	[s7] = 0	[00400150] 3c011001 lui \$1, 4097 [strSpace]; 109: la \$a0, strSpace
R24	[t8] = 0	[00400154] 34240066 ori \$4, \$1, 102 [strspace]
R25	[t9] = 0	[00400158] 34240064 ori \$2, \$0, 4 ; 110: li \$v0, 4
R26	[x0] = 0	[0040015c] 0000000c syscall ; 111: syscall
R27	[x1] = 0	[00400160] 20a50004 addi \$5, \$5, 4 ; 113: addi \$a1, \$a1, 4 # Move to the next element
R28	[gp] = 10008000	[00400164] 0810004f j 0x0040013c [DISPLAY]; 114: j DISPLAY
R29	[sp] = 7ffff1a4	[00400168] 3c011001 lui \$1, 4097 [arr]; 117: la \$t0, arr # \$t0 = address of arr
R30	[s8] = 0	[0040016c] 34280068 ori \$8, \$1, 104 [arr]
R31	[ra] = 4000bc	[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]		Text
R2	[v0] = 4	[004000fc] 1020000d beq \$1, \$0, 52 [CHECK_LOOP-0x004000fc]
R3	[v1] = 0	[00400100] 340b0004 ori \$11, \$0, 4 ; 80: li \$t3, 4 # t3 = 4
R4	[a0] = 10010048	[00400104] 71695802 mul \$11, \$11, \$9 ; 81: mul \$t3, \$t3, \$t1 # t3 = t3*t1
R5	[a1] = 10010078	[00400108] 00ab5820 add \$11, \$5, \$11 ; 82: add \$t3, \$a1, \$t3 # t3 = a1 + t3 = a[i]
R6	[a2] = 7ffff1b0	[0040010c] 21290001 addi \$9, \$9, 1 ; 84: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R7	[a3] = 0	[00400110] 8d6c0000 lw \$12, 0(\$11) ; 86: lw \$t4, 0(\$t3)
R8	[t0] = 10010068	[00400114] 8d6d0004 lw \$13, 4(\$11) ; 87: lw \$t5, 4(\$t3)
R9	[t1] = 1	[00400118] 01ac082a slt \$1, \$13, \$12 ; 89: ble \$t4, \$t5, INNER_LOOP # if t4
R10	[t2] = 0	[0040011c] 1020ff55 beq \$1, \$0, -44 [INNER_LOOP-0x0040011c]
R11	[t3] = 10010068	[00400120] ad6d0000 sw \$13, 0(\$11) ; 91: sw \$t5, 0(\$t3)
R12	[t4] = 22	[00400124] ad6c0004 sw \$12, 4(\$11) ; 92: sw \$t4, 4(\$t3)
R13	[t5] = 2b	[00400128] 34190001 ori \$25, \$0, 1 ; 93: li \$t9, 1 # t9 = 1
R14	[t6] = 0	[0040012c] 0810003c j 0x004000f0 [INNER_LOOP]; 94: j INNER_LOOP
R15	[t7] = 0	[00400130] 13200018 beq \$25, \$0, 96 [END_FUNC-0x00400130]
R16	[s0] = 2	[00400134] 21080001 addi \$8, \$8, 1 ; 98: addi \$t0, \$t0, 1 # Increase the counter of OUTER_LOOP
R17	[s1] = 0	[00400138] 08100037 j 0x004000dc [OUTER_LOOP]; 99: j OUTER_LOOP
R18	[s2] = 0	[0040013c] 8caa0000 lw \$10, 0(\$5) ; 102: lw \$t2, 0(\$a1) # Load t2 = a[i]
R19	[s3] = 0	[00400140] 11400014 beq \$10, \$0, 80 [END_FUNC-0x00400140]
R20	[s4] = 0	[00400144] 00aa2021 addu \$4, \$0, \$10 ; 105: move \$a0, \$t2 # Print the a[i]
R21	[s5] = 0	[00400148] 34020001 ori \$2, \$0, 1 ; 106: li \$v0, 1
R22	[s6] = 0	[0040014c] 0000000c syscall ; 107: syscall
R23	[s7] = 0	[00400150] 3c011001 lui \$1, 4097 [strSpace]; 109: la \$a0, strSpace
R24	[t8] = 0	[00400154] 34240066 ori \$4, \$1, 102 [strspace]
R25	[t9] = 0	[00400158] 34240064 ori \$2, \$0, 4 ; 110: li \$v0, 4
R26	[x0] = 0	[0040015c] 0000000c syscall ; 111: syscall
R27	[x1] = 0	[00400160] 20a50004 addi \$5, \$5, 4 ; 113: addi \$a1, \$a1, 4 # Move to the next element
R28	[gp] = 10008000	[00400164] 0810004f j 0x0040013c [DISPLAY]; 114: j DISPLAY
R29	[sp] = 7ffff1a4	[00400168] 3c011001 lui \$1, 4097 [arr]; 117: la \$t0, arr # \$t0 = address of arr
R30	[s8] = 0	[0040016c] 34280068 ori \$8, \$1, 104 [arr]
R31	[ra] = 4000bc	[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]		Text
R2	[v0] = 4	[00400100] 340b0004 ori \$11, \$0, 4 ; 80: li \$t3, 4 # t3 = 4
R3	[v1] = 0	[00400104] 71695802 mul \$11, \$11, \$9 ; 81: mul \$t3, \$t3, \$t1 # t3 = t3*t1
R4	[a0] = 10010048	[00400108] 00ab5820 add \$11, \$5, \$11 ; 82: add \$t3, \$a1, \$t3 # t3 = a1 + t3 = a[i]
R5	[a1] = 10010078	[0040010c] 21290001 addi \$9, \$9, 1 ; 84: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R6	[a2] = 7ffff1b0	[00400110] 8d6c0000 lw \$12, 0(\$11) ; 86: lw \$t4, 0(\$t3)
R7	[a3] = 0	[00400114] 8d6d0004 lw \$13, 4(\$11) ; 87: lw \$t5, 4(\$t3)
R8	[t0] = 10010068	[00400118] 01ac082a slt \$1, \$13, \$12 ; 89: ble \$t4, \$t5, INNER_LOOP # if t4
R9	[t1] = 1	[0040011c] 1020ff55 beq \$1, \$0, -44 [INNER_LOOP-0x0040011c]
R10	[t2] = 0	[00400120] ad6d0000 sw \$13, 0(\$11) ; 91: sw \$t5, 0(\$t3)
R11	[t3] = 10010068	[00400124] ad6c0004 sw \$12, 4(\$11) ; 92: sw \$t4, 4(\$t3)
R12	[t4] = 22	[00400128] 34190001 ori \$25, \$0, 1 ; 93: li \$t9, 1 # t9 = 1
R13	[t5] = 2b	[0040012c] 0810003c j 0x004000f0 [INNER_LOOP]; 94: j INNER_LOOP
R14	[t6] = 0	[00400130] 13200018 beq \$25, \$0, 96 [END_FUNC-0x00400130]
R15	[t7] = 0	[00400134] 21080001 addi \$8, \$8, 1 ; 98: addi \$t0, \$t0, 1 # Increase the counter of OUTER_LOOP
R16	[s0] = 8	[00400138] 08100037 j 0x004000dc [OUTER_LOOP]; 99: j OUTER_LOOP
R17	[s1] = 0	[0040013c] 8caa0000 lw \$10, 0(\$5) ; 102: lw \$t2, 0(\$a1) # Load t2 = a[i]
R18	[s2] = 0	[00400140] 11400014 beq \$10, \$0, 80 [END_FUNC-0x00400140]
R19	[s3] = 0	[00400144] 00aa2021 addu \$4, \$0, \$10 ; 105: move \$a0, \$t2 # Print the a[i]
R20	[s4] = 0	[00400148] 34020001 ori \$2, \$0, 1 ; 106: li \$v0, 1
R21	[s5] = 0	[0040014c] 0000000c syscall ; 107: syscall
R22	[s6] = 0	[00400150] 3c011001 lui \$1, 4097 [strSpace]; 109: la \$a0, strSpace
R23	[s7] = 0	[00400154] 34240066 ori \$4, \$1, 102 [strspace]
R24	[t8] = 0	[00400158] 34240064 ori \$2, \$0, 4 ; 110: li \$v0, 4
R25	[t9] = 0	[0040015c] 0000000c syscall ; 111: syscall
R26	[x0] = 0	[00400160] 20a50004 addi \$5, \$5, 4 ; 113: addi \$a1, \$a1, 4 # Move to the next element
R27	[x1] = 0	[00400164] 0810004f j 0x0040013c [DISPLAY]; 114: j DISPLAY
R28	[gp] = 10008000	[00400168] 3c011001 lui \$1, 4097 [arr]; 117: la \$t0, arr # \$t0 = address of arr
R29	[sp] = 7ffff1a4	[0040016c] 34280068 ori \$8, \$1, 104 [arr]
R30	[s8] = 0	[00400170] 2210fffe addi \$16, \$16, -2 ; 118: sub \$s0, \$s0, 2 # \$s0 = \$s0 - 2
R31	[ra] = 4000bc	[00400174] 00108080 sll \$16, \$16, 2 ; 119: sll \$s0, \$s0, 2 # \$s0 = \$s0 * 4

Int Regs [16]		Text
R2	[v0] = 4	[00400104] 71695802 mul \$11, \$11, \$9 ; 81: mul \$t3, \$t3, \$t1 # t3 = t3*t1
R3	[v1] = 0	[00400108] 00ab5820 add \$11, \$5, \$11 ; 82: add \$t3, \$a1, \$t3 # t3 = a1 + t3 = a[i]
R4	[a0] = 10010048	[0040010c] 21290001 addi \$9, \$9, 1 ; 84: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R5	[a1] = 10010078	[00400110] 8d6c0000 lw \$12, 0(\$11) ; 86: lw \$t4, 0(\$t3)
R6	[a2] = 7ffff1b0	[00400114] 8d6d0004 lw \$13, 4(\$11) ; 87: lw \$t5, 4(\$t3)
R7	[a3] = 0	[00400118] 01ac082a slt \$1, \$13, \$12 ; 89: ble \$t4, \$t5, INNER_LOOP # if t4
<b>R8</b>	<b>[t0] = 10010070</b>	[0040011c] 1020ffff5 beq \$1, \$0, -44 [INNER_LOOP-0x0040011c] ; 91: mul \$t3, \$t3, \$t1 # t3 = t3*t1
R9	[t1] = 1	[00400120] ad6d0000 sw \$13, 0(\$11) ; 92: add \$t3, \$a1, \$t3 # t3 = a1 + t3 = a[i]
R10	[t2] = 0	[00400124] ad6c0004 sw \$12, 4(\$11) ; 93: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R11	[t3] = 10010068	[00400128] 34190001 or \$25, \$0, 1 ; 94: j INNER_LOOP
R12	[t4] = 22	[00400130] 13200018 beq \$25, \$0, 96 [END_FUNC-0x00400130] ; 95: li \$t9, 1 # t9 = 1
R13	[t5] = 2b	[00400134] 21080001 addi \$8, \$8, 1 ; 98: addi \$t0, \$t0, 1 # Increase the counter of OUTER_LOOP
R14	[t6] = 0	[00400138] 08100037 j 0x004000f0 [INNER_LOOP]; 94: j INNER_LOOP
R15	[t7] = 0	[0040013c] 34280068 addi \$8, \$8, 104 [arr] ; 99: j OUTER_LOOP
R16	[s0] = 8	[00400140] 01040200 add \$8, \$8, \$16 ; 100: add \$t0, \$t0, \$t0 # arr = arr + \$s0
R17	[s1] = 0	[00400144] 000a2021 addu \$4, \$0, \$10 ; 105: move \$a0, \$t2 # Print the a[i]
R18	[s2] = 0	[00400148] 34020001 ori \$2, \$0, 1 ; 106: li \$v0, 1
R19	[s3] = 0	[00400152] 0000000c syscall ; 107: syscall
R20	[s4] = 0	[00400156] 3c011001 lui \$1, 4097 [strSpace] ; 109: la \$a0, strSpace
R21	[s5] = 0	[0040015a] 34240066 ori \$4, \$1, 102 [strSpace] ; 110: li \$v0, 4
R22	[s6] = 0	[0040015e] 34020004 ori \$2, \$0, 4 ; 111: syscall
R23	[s7] = 0	[00400162] 0000000c syscall ; 112: add \$t0, \$t0, \$t0 # arr = arr + \$s0
R24	[t8] = 0	[00400166] 0810004f add \$5, \$5, 4 ; 113: addi \$a1, \$a1, 4 # Move to the next element
R25	[t9] = 0	[00400170] 2210ffffe addi \$16, \$16, -2 ; 114: j DISPLAY
R26	[k0] = 0	[00400174] 00108080 sll \$16, \$16, 2 ; 115: la \$t0, arr # \$t0 = address of arr
R27	[k1] = 0	[00400178] 01104020 add \$8, \$8, \$16 ; 116: sub \$s0, \$s0, 2 # \$s0 = \$s0 - 2
R28	[gp] = 10008000	[0040017c] 8d0b0000 lw \$11, 0(\$8) ; 117: sll \$s0, \$s0, 2 # \$s0 = \$s0 * 4
R29	[sp] = 7ffff1a4	[00400180] 00108080 sll \$16, \$16, 2 ; 118: add \$t0, \$t0, \$s0 # arr = arr + \$s0
R30	[s8] = 0	[00400184] 000b2021 addu \$4, \$0, \$11 ; 119: add \$t0, \$t0, \$s0 # arr = arr + \$s0
R31	[ra] = 4000bc	[00400188] 8d0b0000 lw \$t3, 0(\$t0) ; 120: add \$t0, \$t0, \$t3 # arr = arr[\$s0]
		[0040018c] 000b2021 addu \$4, \$0, \$11 ; 121: lw \$t3, 0(\$t0) # \$t3 = arr[\$s0]
		[00400190] 000b2021 addu \$4, \$0, \$11 ; 122: lw \$t3, 0(\$t0) # \$t3 = arr[\$s0]
		[00400194] 000b2021 addu \$4, \$0, \$11 ; 123: lw \$t3, 0(\$t0) # \$t3 = arr[\$s0]
		[00400198] 000b2021 addu \$4, \$0, \$11 ; 124: move \$a0, \$t3 # Print the \$t3

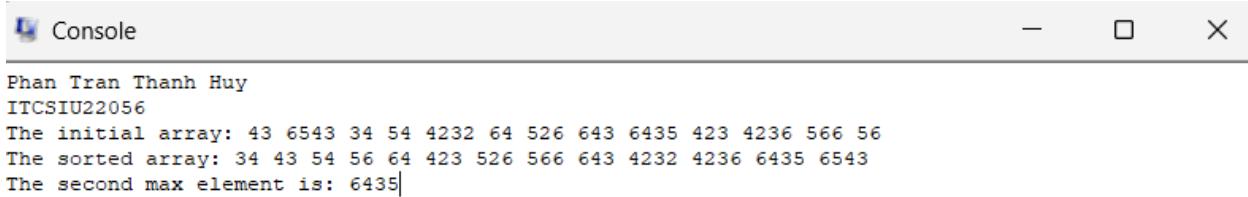
  

Int Regs [16]		Text
R2	[v0] = 4	[00400108] 00ab5820 add \$11, \$5, \$11 ; 82: add \$t3, \$a1, \$t3 # t3 = a1 + t3 = a[i]
R3	[v1] = 0	[0040010c] 21290001 addi \$9, \$9, 1 ; 84: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R4	[a0] = 10010048	[00400110] 8d6c0000 lw \$12, 0(\$11) ; 86: lw \$t4, 0(\$t3)
R5	[a1] = 10010078	[00400114] 8d6d0004 lw \$13, 4(\$11) ; 87: lw \$t5, 4(\$t3)
R6	[a2] = 7ffff1b0	[00400118] 01ac082a slt \$1, \$13, \$12 ; 89: ble \$t4, \$t5, INNER_LOOP # if t4
R7	[a3] = 0	[0040011c] 1020ffff5 beq \$1, \$0, -44 [INNER_LOOP-0x0040011c] ; 91: mul \$t3, \$t3, \$t1 # t3 = t3*t1
R8	[t0] = 10010070	[00400120] ad6d0000 sw \$13, 0(\$11) ; 92: add \$t3, \$a1, \$t3 # t3 = a1 + t3 = a[i]
R9	[t1] = 1	[00400124] ad6c0004 sw \$12, 4(\$11) ; 93: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R10	[t2] = 0	[00400128] 34190001 or \$25, \$0, 1 ; 94: j INNER_LOOP
<b>R11</b>	<b>[t3] = 36</b>	[00400130] 13200018 beq \$25, \$0, 96 [END_FUNC-0x00400130] ; 95: li \$t9, 1 # t9 = 1
R12	[t4] = 22	[00400134] 21080001 addi \$8, \$8, 1 ; 98: addi \$t0, \$t0, 1 # Increase the counter of OUTER_LOOP
R13	[t5] = 2b	[00400138] 08100037 j 0x004000f0 [INNER_LOOP]; 94: j INNER_LOOP
R14	[t6] = 0	[00400142] 34280068 addi \$8, \$8, 104 [arr] ; 99: j OUTER_LOOP
R15	[t7] = 0	[00400146] 0810004f add \$5, \$5, 4 ; 100: add \$t0, \$t0, \$t0 # arr = arr + \$s0
R16	[s0] = 8	[0040014a] 000a2021 addu \$4, \$0, \$10 ; 105: move \$a0, \$t2 # Print the a[i]
R17	[s1] = 0	[0040014e] 34020001 ori \$2, \$0, 1 ; 106: li \$v0, 1
R18	[s2] = 0	[00400152] 0000000c syscall ; 107: syscall
R19	[s3] = 0	[00400156] 3c011001 lui \$1, 4097 [strSpace] ; 109: la \$a0, strSpace
R20	[s4] = 0	[0040015a] 34240066 ori \$4, \$1, 102 [strSpace] ; 110: li \$v0, 4
R21	[s5] = 0	[0040015e] 34020004 ori \$2, \$0, 4 ; 111: syscall
R22	[s6] = 0	[00400162] 0000000c syscall ; 112: add \$t0, \$t0, \$t0 # arr = arr + \$s0
R23	[s7] = 0	[00400166] 0810004f add \$5, \$5, 4 ; 113: addi \$a1, \$a1, 4 # Move to the next element
R24	[t8] = 0	[00400170] 2210ffffe addi \$16, \$16, -2 ; 114: j DISPLAY
R25	[t9] = 0	[00400174] 00108080 sll \$16, \$16, 2 ; 115: la \$t0, arr # \$t0 = address of arr
R26	[k0] = 0	[00400178] 01104020 add \$8, \$8, \$16 ; 116: sub \$s0, \$s0, 2 # \$s0 = \$s0 - 2
R27	[k1] = 0	[0040017c] 8d0b0000 lw \$11, 0(\$8) ; 117: sll \$s0, \$s0, 2 # \$s0 = \$s0 * 4
R28	[gp] = 10008000	[00400180] 00108080 sll \$16, \$16, 2 ; 118: add \$t0, \$t0, \$s0 # arr = arr + \$s0
R29	[sp] = 7ffff1a4	[00400184] 000b2021 addu \$4, \$0, \$11 ; 119: add \$t0, \$t0, \$s0 # arr = arr + \$s0
R30	[s8] = 0	[00400188] 000b2021 addu \$4, \$0, \$11 ; 120: add \$t0, \$t0, \$t3 # arr = arr + \$s0
R31	[ra] = 4000bc	[00400192] 000b2021 addu \$4, \$0, \$11 ; 121: lw \$t3, 0(\$t0) # \$t3 = arr[\$s0]
		[00400196] 000b2021 addu \$4, \$0, \$11 ; 122: lw \$t3, 0(\$t0) # \$t3 = arr[\$s0]
		[004001a0] 000b2021 addu \$4, \$0, \$11 ; 123: lw \$t3, 0(\$t0) # \$t3 = arr[\$s0]
		[004001a4] 000b2021 addu \$4, \$0, \$11 ; 124: move \$a0, \$t3 # Print the \$t3

Int Regs [16]		Text
R2	[v0] = 4	[0040010c] 21290001 addi \$9, \$9, 1 ; 84: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R3	[v1] = 0	[00400110] 8d6c0000 lw \$12, 0(\$11) ; 86: lw \$t4, 0(\$t3)
R4	[a0] = 10010078	[00400114] 8d6d0004 lw \$13, 4(\$11) ; 87: lw \$t5, 4(\$t3)
R5	[a1] = 10010078	[00400118] 01ac082a slt \$1, \$13, \$12 ; 89: ble \$t4, \$t5, INNER_LOOP # if t4
R6	[a2] = 7ffff1b0	[0040011c] 1020ffff5 beq \$1, \$0, -44 [INNER_LOOP-0x0040011c] ; 91: mul \$t3, \$t3, \$t1 # t3 = t3*t1
R7	[a3] = 0	[00400120] ad6d0000 sw \$13, 0(\$11) ; 92: add \$t3, \$a1, \$t3 # t3 = a1 + t3 = a[i]
R8	[t0] = 10010070	[00400124] ad6c0004 sw \$12, 4(\$11) ; 93: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R9	[t1] = 1	[00400128] 34190001 or \$25, \$0, 1 ; 94: j INNER_LOOP
R10	[t2] = 0	[00400130] 13200018 beq \$25, \$0, 96 [END_FUNC-0x00400130] ; 95: li \$t9, 1 # t9 = 1
<b>R11</b>	<b>[t3] = 36</b>	[00400134] 21080001 addi \$8, \$8, 1 ; 98: addi \$t0, \$t0, 1 # Increase the counter of OUTER_LOOP
R12	[t4] = 22	[00400138] 08100037 j 0x004000f0 [INNER_LOOP]; 94: j INNER_LOOP
R13	[t5] = 2b	[00400142] 34280068 addi \$8, \$8, 104 [arr] ; 99: j OUTER_LOOP
R14	[t6] = 0	[00400146] 0810004f add \$5, \$5, 4 ; 100: add \$t0, \$t0, \$t0 # arr = arr + \$s0
R15	[t7] = 0	[00400150] 0810004f add \$5, \$5, 4 ; 101: j OUTER_LOOP
R16	[s0] = 8	[00400154] 000a2021 addu \$4, \$0, \$10 ; 105: move \$a0, \$t2 # Print the a[i]
R17	[s1] = 0	[00400158] 34020001 ori \$2, \$0, 1 ; 106: li \$v0, 1
R18	[s2] = 0	[00400162] 0000000c syscall ; 107: syscall
R19	[s3] = 0	[00400166] 3c011001 lui \$1, 4097 [strSpace] ; 109: la \$a0, strSpace
R20	[s4] = 0	[00400170] 34240066 ori \$4, \$1, 102 [strSpace] ; 110: li \$v0, 4
R21	[s5] = 0	[00400174] 34020004 ori \$2, \$0, 4 ; 111: syscall
R22	[s6] = 0	[00400178] 0000000c syscall ; 112: add \$t0, \$t0, \$t0 # arr = arr + \$s0
R23	[s7] = 0	[00400182] 0000000c syscall ; 113: addi \$a1, \$a1, 4 # Move to the next element
R24	[t8] = 0	[00400186] 0810004f add \$5, \$5, 4 ; 114: j DISPLAY
R25	[t9] = 0	[00400190] 0810004f add \$5, \$5, 4 ; 115: la \$t0, arr # \$t0 = address of arr
R26	[k0] = 0	[00400194] 0810004f add \$5, \$5, 4 ; 116: sub \$s0, \$s0, 2 # \$s0 = \$s0 - 2
R27	[k1] = 0	[00400198] 0810004f add \$5, \$5, 4 ; 117: sll \$s0, \$s0, 2 # \$s0 = \$s0 * 4
R28	[gp] = 10008000	[00400202] 0810004f add \$5, \$5, 4 ; 118: add \$t0, \$t0, \$s0 # arr = arr + \$s0
R29	[sp] = 7ffff1a4	[00400206] 0810004f add \$5, \$5, 4 ; 119: add \$t0, \$t0, \$s0 # arr = arr + \$s0
R30	[s8] = 0	[00400210] 0810004f add \$5, \$5, 4 ; 120: add \$t0, \$t0, \$t3 # arr = arr + \$s0
R31	[ra] = 4000bc	[00400214] 0810004f add \$5, \$5, 4 ; 121: lw \$t3, 0(\$t0) # \$t3 = arr[\$s0]
		[00400218] 0810004f add \$5, \$5, 4 ; 122: lw \$t3, 0(\$t0) # \$t3 = arr[\$s0]
		[00400222] 0810004f add \$5, \$5, 4 ; 123: lw \$t3, 0(\$t0) # \$t3 = arr[\$s0]
		[00400226] 0810004f add \$5, \$5, 4 ; 124: move \$a0, \$t3 # Print the \$t3

Print the result:



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

## Exercise 4:

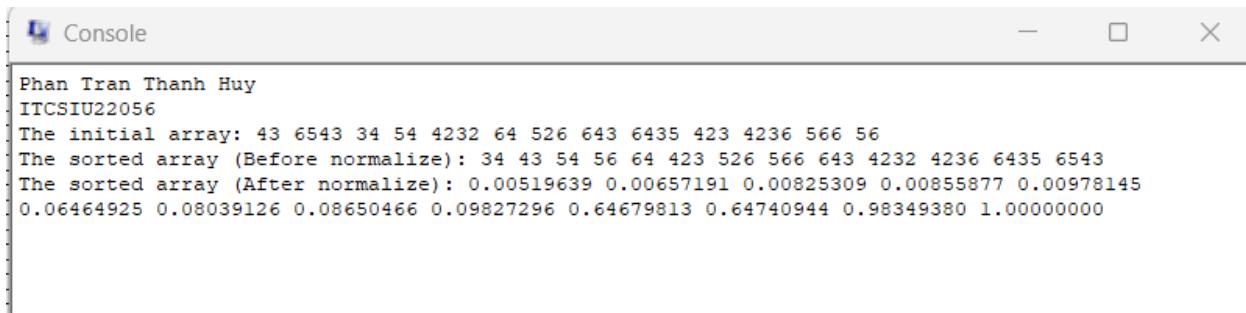
## Code:

```

1  .data
2  strHello: .ascii "Hello, Intel Mac! (CPU: %04X@%04X)\n"
3  arrInitialArray: .ascii "The initial array:\n"
4  arrNormalized: .ascii "The sorted array (Before normalizing):\n"
5  arrNormalizedArray: .ascii "The sorted array (After normalizing):\n"
6  arrSpace: .ascii "\n"
7  strSpace: .ascii " "
8  arr: .word $10, $0A, $0, $0, $0B, $0C, $0D, $0E, $0F, $0G, $0H, $0I, $0J, $0K, $0L, $0M, $0N, $0O, $0P, $0Q, $0R, $0S, $0T, $0U, $0V, $0W, $0X, $0Y, $0Z
9  sortIndex: .space 1000
10 .text
11 .globl main
12
13 main:
14     la $a0, arr        # $a0 = address of arr
15     li $a1, 0           # $a1 = 0
16     jal L0K            # Compute the length of the array
17
18     la $a0, arr        # $a0 = address of arr
19     li $a1, 4           # $a1 = 4
20     syscall
21
22     la $a0, arr        # $a0 = address of arr
23     li $a1, 0           # $a1 = 0
24     jal L0M            # Display the array
25
26     la $a0, arr        # $a0 = address of arr
27     li $a1, 1           # $a1 = 1
28     jal L0N            # Sort the array
29
30     la $a0, arr        # $a0 = address of arr
31     li $a1, 4           # $a1 = 4
32     syscall
33
34     la $a0, arr        # $a0 = address of arr
35     li $a1, 0           # $a1 = 0
36     jal L0O            # Sort the array
37
38     la $a0, arr        # $a0 = address of arr
39     li $a1, 4           # $a1 = 4
40     syscall
41
42     la $a0, arr        # $a0 = address of arr
43     li $a1, 0           # $a1 = 0
44     jal L0P            # Display the array
45
46     la $a0, arr        # $a0 = address of arr
47     li $a1, 4           # $a1 = 4
48     syscall
49
50     la $a0, arr        # $a0 = address of arr
51     li $a1, 0           # $a1 = 0
52     jal L0Q            # Normalize the array
53
54     la $a0, arr        # $a0 = address of arr
55     li $a1, 4           # $a1 = 4
56     syscall
57
58     ntbl $t0, $t1, $t0  # Convert the maximum element $t0 from int to float $t1
59     cvtfs $t0, $t1       # $t0 = $t0
60     bne $t0, $0, END_FUNC # $t0 = 0, stop the END_FUNC
61     la $a0, arr        # $a0 = address of arr
62     jal NUMBER_TFF    # Display the normalized array
63
64     li $v0, 10
65     syscall
66
67 LDN:
68     la $t1, $(arr)      # Load $t1 = arr[1]
69     bne $t1, $0, END_FUNC # $t1 = 0, stop the LDN
70     beq $t1, $0, END_FUNC # $t1 = 0, stop the LDN
71     addi $t1, $t1, 1      # Move the next element
72     addi $t0, $t0, 1      # Increase the counter
73     j LDN
74
75 SORT:
76     OUTTER_LOOP:
77     QUIT_LOOP:
78     la $t0, arr, END_FUNC # Check if t0 == -8 - length of the array, stop the OUTTER_LOOP
79     li $t1, 0
80     addi $t1, $t1, -8      # t1 = -8
81     addi $t1, $t1, -1      # t1 = -9
82     bne $t1, $t0, QUIT_LOOP # Check if t1 <= t0, go to QUIT_LOOP
83     li $t1, 4
84     mul $t1, $t1, $t1      # t1 = t1*t1
85     add $t1, $t1, $t0      # t1 = 21*t1 - t0
86
87     addi $t1, $t1, 1      # Increase the counter of INNTER_LOOP
88
89     Iw $t1, $(arr)
90     Iw $t1, $(arr)
91     Iw $t1, $(arr)
92
93     BNE $t1, $t0, TIMER_LOOP # If t1 < t5, skip the swap
94     d Swap Variable
95     sw $t1, $(arr)
96     sw $t0, $(arr)
97     li $t1, 4
98     add $t1, $t1, $t0      # t1 = t1 + t0
99     j INNTER_LOOP
100
101 INNTER_LOOP:
102     bge $t1, $t0, END_FUNC # Check if t1 >= 0, stop the sort
103     addi $t1, $t1, 1      # Increase the counter of INNTER_LOOP
104     j OUTTER_LOOP
105
106 DSWAP:
107     Iw $t1, $(arr)        # Load t1 = arr[1]
108     bne $t1, $0, END_FUNC # Check if t1 == 0, go to END_FUNC
109
110     move $t0, $t1          # Print the arr[1]
111     li $v0, 1
112     syscall
113
114     la $t0, $t0
115     la $t0, $t0
116     li $v0, 1
117     syscall
118
119     addi $t1, $t1, 4      # Move to the next element
120     j INNTER_LOOP
121
122 MATHFUNC:
123     la $t0, arr        # $t0 = address of arr
124     sub $t0, $t0, 1      # $t0 = $t0 - 1
125     sll $t0, $t0, 1      # $t0 = $t0 * 2
126     add $t0, $t0, $t0      # $t0 = arr[0] + $t0
127
128     Iw $t1, $(arr)        # Load t1 = arr[50]
129
130     j END_FUNC
131
132 NORMALIZE:
133     la $t1, $(arr)        # $t1 = arr[1]
134     bne $t1, $0, END_FUNC # Check if $t1 <= 0, go to END_FUNC
135
136     ntbl $t1, $t2, $t0    # Convert $t1 from int to float $t2
137     cvtfs $t2, $t2, $t0
138
139     divs $t1, $t2, $t0    # Divide $t1 / $t2/80 = arr[1]/maximum element
140     move $t1, $t2          # Print the $t1
141     li $v0, 1
142     syscall
143
144     addi $t1, $t1, 4      # Move to the next element
145     j NORMALIZE
146
147 L0K:
148     la $t0, strSpace
149     li $v0, 4
150     syscall
151
152 L0M:
153     la $t0, strSpace
154     li $v0, 4
155     syscall
156
157 L0N:
158     addi $t1, $t1, 4      # Move to the next element
159     j NORMALIZE
160
161 L0O:
162     li $v0, 4

```

**Test case:**



```
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]	Text
R2 [v0] = 4	[00400114] 340b0004 ori \$11, \$0, 4 ; 85: li \$t3, 4 # t3 = 4
R3 [v1] = 0	[00400118] 71695802 mul \$11, \$11, \$9 ; 86: mul \$t3, \$t3, \$t1 # t3 = t3*t1
R4 [a0] = 10010080	[0040011c] 00ab5820 add \$11, \$5, \$11 ; 87: add \$t3, \$a1, \$t3 # t3 = a1 + t3 = a[i]
R5 [a1] = 10010094	[00400120] 21290001 addi \$9, \$9, 1 ; 89: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R6 [a2] = 7ffff1b0	[00400124] 8d6c0000 lw \$12, 0(\$11) ; 91: lw \$t4, 0(\$t3)
R7 [a3] = 0	[00400128] 8d6d0004 lw \$13, 4(\$11) ; 92: lw \$t5, 4(\$t3)
R8 [t0] = 10010084	[0040012c] 01ac082a slt \$1, \$13, \$12 ; 94: ble \$t4, \$t5, INNER_LOOP # if t4
R9 [t1] = 1	[00400130] 1020ffff5 beg \$1, \$0, -44 [INNER_LOOP-0x00400130]
R10 [t2] = 0	[00400134] ad6d0000 sw \$13, 0(\$11) ; 96: sw \$t5, 0(\$t3)
R11 [t3] = 10010084	[00400138] ad6c0004 sw \$12, 4(\$11) ; 97: sw \$t4, 4(\$t3)
R12 [t4] = 22	[0040013c] 34190001 ori \$25, \$0, 1 ; 98: li \$t9, 1 # t9 = 1
R13 [t5] = 2b	[00400140] 08100041 j 0x00400104 [INNER_LOOP]; 99: j INNER_LOOP
R14 [t6] = 0	[00400144] 13200023 beg \$25, \$0, 140 [END_FUNC-0x00400144]
R15 [t7] = 0	[00400148] 21080001 addi \$8, \$8, 1 ; 103: addi \$t0, \$t0, 1 # Increase the counter of OUTER_LOOP
R16 [s0] = 4	[0040014c] 0810003c j 0x004000f0 [OUTER_LOOP]; 104: j OUTER_LOOP
R17 [s1] = 0	[00400150] 8caa0000 lw \$10, 0(\$5) ; 107: lw \$t2, 0(\$a1) # Load t2 = a[i]
R18 [s2] = 0	[00400154] 1140001f beg \$10, \$0, 124 [END_FUNC-0x00400154]
R19 [s3] = 0	[00400158] 000a2021 addu \$4, \$0, \$10 ; 110: move \$a0, \$t2 # Print the a[i]
R20 [s4] = 0	[0040015c] 34020001 ori \$2, \$0, 1 ; 111: li \$v0, 1
R21 [s5] = 0	[00400160] 0000000c syscall ; 112: syscall
R22 [s6] = 0	[00400164] 3c011001 lui \$1, 4097 [strSpace] ; 114: la \$a0, strSpace
R23 [s7] = 0	[00400168] 34240082 ori \$4, \$1, 130 [strSpace]
R24 [t8] = 0	[0040016c] 34020004 ori \$2, \$0, 4 ; 115: li \$v0, 4
R25 [t9] = 0	[00400170] 0000000c syscall ; 116: syscall
R26 [k0] = 0	[00400174] 20a50004 addi \$5, \$5, 4 ; 118: addi \$a1, \$a1, 4 # Move to the next element
R27 [k1] = 0	[00400178] 08100054 j 0x00400150 [DISPLAY]; 119: j DISPLAY
R28 [gp] = 10008000	[0040017c] 3c011001 lui \$1, 4097 [arr] ; 122: la \$t0, arr # \$t0 = address of arr
R29 [sp] = 7ffff1a4	[00400180] 34280084 ori \$8, \$1, 132 [arr]
R30 [s8] = 0	[00400184] 2210ffff beg \$1, \$0, -44 [INNER_LOOP-0x00400184]
R31 [ra] = 4000ac	[00400188] 00108080 addi \$16, \$16, -1 ; 123: sub \$s0, \$s0, 1 # \$s0 = \$s0 - 1
	[0040018c] 01104020 sll \$16, \$16, 2 ; 124: sll \$s0, \$s0, 2 # \$s0 = \$s0 * 4
	[00400190] 01104020 add \$8, \$8, \$16 ; 125: add \$t0, \$t0, \$s0 # arr = arr + \$s0

Int Regs [16]	Text
R2 [v0] = 4	[00400114] 340b0004 ori \$11, \$0, 4 ; 85: li \$t3, 4 # t3 = 4
R3 [v1] = 0	[00400118] 71695802 mul \$11, \$11, \$9 ; 86: mul \$t3, \$t3, \$t1 # t3 = t3*t1
R4 [a0] = 10010080	[0040011c] 00ab5820 add \$11, \$5, \$11 ; 87: add \$t3, \$a1, \$t3 # t3 = a1 + t3 = a[i]
R5 [a1] = 10010094	[00400120] 21290001 addi \$9, \$9, 1 ; 89: addi \$t1, \$t1, 1 # Increase the counter of INNER_LOOP
R6 [a2] = 7ffff1b0	[00400124] 8d6c0000 lw \$12, 0(\$11) ; 91: lw \$t4, 0(\$t3)
R7 [a3] = 0	[00400128] 8d6d0004 lw \$13, 4(\$11) ; 92: lw \$t5, 4(\$t3)
R8 [t0] = 10010084	[0040012c] 01ac082a slt \$1, \$13, \$12 ; 94: ble \$t4, \$t5, INNER_LOOP # if t4
R9 [t1] = 1	[00400130] 1020ffff5 beg \$1, \$0, -44 [INNER_LOOP-0x00400130]
R10 [t2] = 0	[00400134] ad6d0000 sw \$13, 0(\$11) ; 96: sw \$t5, 0(\$t3)
R11 [t3] = 10010084	[00400138] ad6c0004 sw \$12, 4(\$11) ; 97: sw \$t4, 4(\$t3)
R12 [t4] = 22	[0040013c] 34190001 ori \$25, \$0, 1 ; 98: li \$t9, 1 # t9 = 1
R13 [t5] = 2b	[00400140] 08100041 j 0x00400104 [INNER_LOOP]; 99: j INNER_LOOP
R14 [t6] = 0	[00400144] 13200023 beg \$25, \$0, 140 [END_FUNC-0x00400144]
R15 [t7] = 0	[00400148] 21080001 addi \$8, \$8, 1 ; 103: addi \$t0, \$t0, 1 # Increase the counter of OUTER_LOOP
R16 [s0] = 3	[0040014c] 0810003c j 0x004000f0 [OUTER_LOOP]; 104: j OUTER_LOOP
R17 [s1] = 0	[00400150] 8caa0000 lw \$10, 0(\$5) ; 107: lw \$t2, 0(\$a1) # Load t2 = a[i]
R18 [s2] = 0	[00400154] 1140001f beg \$10, \$0, 124 [END_FUNC-0x00400154]
R19 [s3] = 0	[00400158] 000a2021 addu \$4, \$0, \$10 ; 110: move \$a0, \$t2 # Print the a[i]
R20 [s4] = 0	[0040015c] 34020001 ori \$2, \$0, 1 ; 111: li \$v0, 1
R21 [s5] = 0	[00400160] 0000000c syscall ; 112: syscall
R22 [s6] = 0	[00400164] 3c011001 lui \$1, 4097 [strSpace] ; 114: la \$a0, strSpace
R23 [s7] = 0	[00400168] 34240082 ori \$4, \$1, 130 [strSpace]
R24 [t8] = 0	[0040016c] 34020004 ori \$2, \$0, 4 ; 115: li \$v0, 4
R25 [t9] = 0	[00400170] 0000000c syscall ; 116: syscall
R26 [k0] = 0	[00400174] 20a50004 addi \$5, \$5, 4 ; 118: addi \$a1, \$a1, 4 # Move to the next element
R27 [k1] = 0	[00400178] 08100054 j 0x00400150 [DISPLAY]; 119: j DISPLAY
R28 [gp] = 10008000	[0040017c] 3c011001 lui \$1, 4097 [arr] ; 122: la \$t0, arr # \$t0 = address of arr
R29 [sp] = 7ffff1a4	[00400180] 34280084 ori \$8, \$1, 132 [arr]
R30 [s8] = 0	[00400184] 2210ffff beg \$1, \$0, -44 [INNER_LOOP-0x00400184]
R31 [ra] = 4000ac	[00400188] 00108080 addi \$16, \$16, -1 ; 123: sub \$s0, \$s0, 1 # \$s0 = \$s0 - 1
	[0040018c] 01104020 sll \$16, \$16, 2 ; 124: sll \$s0, \$s0, 2 # \$s0 = \$s0 * 4
	[00400190] 01104020 add \$8, \$8, \$16 ; 125: add \$t0, \$t0, \$s0 # arr = arr + \$s0

Int Regs [16] Text

```

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

[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, $9, $11 ; 87: add $t3, $a1, $t1 # 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] 1020ffff beg $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] 01040200 add $25, $0, 140 [END_FUNC-0x00400144]
[0040014c] 0810003c ori $0, $0, 1 ; 103: addi $t0, $t0, 1 # Increase the counter of 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 [arr] ; 119: j DISPLAY
[00400168] 34280084 ori $8, $1, 132 [arr] ; 122: la $t0, arr # $t0 = address of arr
[00400172] 2210ffff addi $16, $16, -1 ; 123: sub $s0, $s0, 1 # $s0 = $s0 - 1
[00400174] 2210ffff addi $16, $16, -1 ; 124: sll $s0, $s0, 2 # $s0 = $s0 * 4
[00400178] 08100054 add $8, $8, $16 ; 125: add $t0, $t0, $s0 # arr = arr + $s0
[00400180] 00108080 sll $16, $16, 2 ; 127: lw $t3, 0($s0) # $t3 = arr[$s0]
[00400184] 01104020 add $8, $8, $16 ; 128: add $t0, $t0, $s0 # arr = arr + $s0

```

Int Regs [16] Text

```

R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 10010080
R5 [a1] = 10010094
R6 [a2] = 7ffff1b0
R7 [a3] = 0
R8 [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

[00400118] 71695802 mul $11, $11, $9 ; 86: mul $t3, $t3, $t1 # t3 = t3*t1
[0040011c] 00ab5820 add $11, $9, $11 ; 87: add $t3, $a1, $t1 # 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] 1020ffff beg $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 add $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 [arrSpace] ; 114: la $a0, strSpace
[00400168] 34240082 ori $8, $1, 130 [strSpace] ; 115: li $v0, 4
[00400172] 34020004 ori $2, $0, 4 ; 116: syscall
[00400176] 0000000c syscall ; 117: syscall
[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] ; 123: sub $s0, $s0, 1 # $s0 = $s0 - 1
[00400184] 2210ffff addi $16, $16, -1 ; 124: sll $s0, $s0, 2 # $s0 = $s0 * 4
[00400188] 00108080 sll $16, $16, 2 ; 125: add $t0, $t0, $s0 # arr = arr + $s0
[0040018c] 01104020 add $8, $8, $16 ; 127: lw $t3, 0($s0) # $t3 = arr[$s0]
[00400190] 8d0b0000 lw $11, 0($8) ; 128: add $t0, $t0, $s0 # arr = arr + $s0

```

Int Regs [16] Text

```

R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 10010080
R5 [a1] = 10010094
R6 [a2] = 7ffff1b0
R7 [a3] = 0
R8 [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

[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] 1020ffff beg $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 add $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 [arrSpace] ; 114: la $a0, strSpace
[00400168] 34240082 ori $8, $1, 130 [strSpace] ; 115: li $v0, 4
[00400172] 34020004 ori $2, $0, 4 ; 116: syscall
[00400176] 0000000c syscall ; 117: syscall
[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] ; 123: sub $s0, $s0, 1 # $s0 = $s0 - 1
[00400184] 2210ffff addi $16, $16, -1 ; 124: sll $s0, $s0, 2 # $s0 = $s0 * 4
[00400188] 00108080 sll $16, $16, 2 ; 125: add $t0, $t0, $s0 # arr = arr + $s0
[0040018c] 01104020 add $8, $8, $16 ; 127: lw $t3, 0($s0) # $t3 = arr[$s0]
[00400190] 8d0b0000 lw $11, 0($8) ; 128: add $t0, $t0, $s0 # arr = arr + $s0

```

## Normalize display steps:

Int Regs [16]

R2	[v0] = 4
R3	[v1] = 0
R4	[a0] = 1001005b
R5	[a1] = 10010084
R6	[a2] = 7ffff1b0
R7	[a3] = 0
R8	[t0] = 10010090
R9	[t1] = 0
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	[t10] = 0
R27	[t11] = 0
R28	[gp] = 10000000
R29	[sp] = 7ffff1a4
R30	[s8] = 0
R31	[ra] = 40000d0

Text

```
[00400094] 0c100054 jal 0x00400150 [DISPLAY] ; 43: jal DISPLAY # Display the array
[00400098] 3c011001 lui $1, 4097 [strLine] ; 45: la $a0, strLine
[0040009c] 34240080 ori $4, $1, 128 [strLine]
[004000a0] 34020004 ori $2, $0, 4 ; 46: li $v0, 4
[004000a4] 0000000c syscall ; 47: syscall
[004000a8] 0c10005f jal 0x0040017c [MAXIMUM] ; 48: jal MAXIMUM
[004000ac] 3c011001 lui $1, 4097 [printNormalizeArray]
[004000b0] 3424005b ori $4, $1, 91 [printNormalizeArray]
[004000b4] 00020004 ori $2, $0, 4 ; 49: li $v0, 4
[004000b8] 0000000c syscall ; 50: syscall
[004000bc] 448b0000 mtc1 $11, $f0 ; 55: mtc1 $t3, $f0 # Convert the maximum element $t3 from int to float $f0
[004000c0] 46580020 cvt.s.w $f0, $f0 ; 56: cvt.s.w $f0, $f0
[004000c4] 3c011001 lui $1, 4097 [arr] ; 57: la $a1, arr # $a1 = address of arr
[004000c8] 34250084 ori $5, $1, 132 [arr]
[004000cc] 0c100066 jal 0x00400198 [NORMALIZE] ; 58: jal NORMALIZE # Display the normalized array
[004000d0] 34020004 ori $2, $0, 10 ; 60: li $v0, 10
[004000d4] 0000000c syscall ; 61: syscall
[004000d8] 8caa0000 lw $10, 0($5) ; 64: lw $t2, 0($a1) # Load t2 = a[i]
[004000dc] 1140003d beq $10, $0, 244 [END_FUNC-0x004000dc]
[004000e0] 20a50004 addi $10, $5, 4 ; 67: addi $a1, $a1, 4 # Move the next element
[004000e4] 22100001 addi $16, $16, 1 ; 68: addi $s0, $s0, 1 # Increase counter
[004000e8] 08100036 j 0x004000d8 [LEN] ; 69: j LEN
[004000ec] 0810003c j 0x004000f0 [OUTER_LOOP] ; 72: j OUTER_LOOP # Jump to OUTER_LOOP
[004000f0] 0110082a sll $1, $1, 128 [OUTER_LOOP]
[004000f4] 1020ffff beq $1, $0, 220 [END_FUNC-0x004000f4]
[004000f8] 34150000 ori $25, $0, 0 ; 75: li $t9, 0 # t9 = 0
[004000fc] 34050000 ori $9, $0, 0 ; 77: li $t1, 0 # t1 = 0
[00400104] 08100041 j 0x00400104 [INNER_LOOP] ; 78: j INNER_LOOP # Jump to INNER_LOOP
[00400108] 02085022 sub $10, $16, $8 ; 81: sub $t2, $s0, $t0 # t2 = a0 - t0
[0040010c] 214affff addi $10, $10, -1 ; 82: addi $t2, $t2, -1 # t2 = t2 - 1 = the current index
[0040010e] 012a082a slt $1, $9, $10 ; 84: bge $t1, $t2, CHECK_LOOP # Check if t1 >= t2, go to CHECK_LOOP
```

Int Regs [16]

R2	[v0] = 4
R3	[v1] = 0
R4	[a0] = 1001005b
R5	[a1] = 10010084
R6	[a2] = 7ffff1b0
R7	[a3] = 0
R8	[t0] = 10010090
R9	[t1] = 1
R10	[t2] = 22
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] = 10000000
R29	[sp] = 7ffff1a4
R30	[s8] = 0
R31	[ra] = 40000d0

Text

```
[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, $t4, $t5, INNER_LOOP # if t4
[00400130] 1020ffff 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] 000a0201 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 $a1, arr # $t0 = address of 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]
```

IP Regs | Int Regs [16]

FIR	= 9800
FCCSR	= 0
Single Precision	
F00	= 45cc7800
FG1	= 0
FG2	= 42000000
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
...	= 0

Data | Text

```
[00400130] 1020ffff 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] 000a0201 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] 3c0101001 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 $a1, arr # $t0 = address of 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]
[0040019f] 448a1000 mtc1 $10, $f2 ; 135: mtc1 $t2, $f2 # Convert $t2 from int to float $f2
[004001a0] 448a1000 cvt.s.w $f2, $f2 ; 136: cvt.s.w $f2, $f2
[004001a4] 468010a0 div.s $f2, $f2, $f0 ; 138: div.s $f2, $f2, $f0 # Divide $f2 = $f2/$f0 = arr[i]/Maximum_element
```

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  
... = 0

Text

```
[00400138] ad6d0000 sw $13, 0($11) ; 96: sw $t5, 0($t3)
[00400138] ad6d0004 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
[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
[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 mtcl $10, $f2 ; 135: mtcl $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
[004001a8] 46001083 mov.s $f12, $f2 ; 140: mov.s $f12, $f2 # Print the $f2
[004001ac] 08100108 beq $10, 0($5) ; 141: li $v0, 2
```

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  
... = 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
[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
[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 mtcl $10, $f2 ; 135: mtcl $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
[004001a8] 46001083 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]

Int Regs [16]

R2 [v0] = 4  
R3 [v1] = 0  
R4 [a0] = 10010082  
R5 [a1] = 10010088  
R6 [a2] = 7fffffb0  
R7 [a3] = 0  
R8 [t0] = 10010090  
R9 [t1] = 1  
R10 [t2] = 2b  
R11 [t3] = 19ff  
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] = 10000800  
R29 [sp] = 7fffff1a4  
R30 [s8] = 0  
R31 [ra] = 40000d0

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 mtcl $10, $f2 ; 135: mtcl $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
[004001a8] 46001083 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
FIR = 9800 FCSR = 0	Single Precision FG0 = 45cc7800 FG1 = 0 FG2 = 3bd7592b	Text	<pre>[00400154] 1140001f beg \$10, \$0, 124 [END_FUNC-0x00400154] [00400158] 000a2021 add \$4, \$0, \$10 ; 110: move \$a0, \$t2 # Print the a[i] [0040015c] 34020001 ori \$2, \$0, 1 ; 111: li \$v0, 1 [00400160] 00000000 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] 00000004 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] 34240084 ori \$8, \$1, 132 [arr] [00400184] 2210ffff addi \$16, \$16, -1 ; 123: sub \$s0, \$s0, 1 # \$s0 = \$s0 - 1 [00400188] 00100800 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(\$s0) ; 127: lw \$t3, 0(\$t0) # \$t3 = arr[\$s0] [00400194] 08100074 j 0x004001d0 [END_FUNC] ; 129: j END_FUNC [00400198] 8caa0000 lw \$10, 0(\$s5) ; 132: lw \$t2, 0(\$a1) # \$t2 = arr[i] [004001a0] 11400000 beg \$10, \$0, 52 [END_FUNC-0x0040019c] [004001a1] 448a1000 mtc1 \$10, \$f2 ; 135: mtc1 \$t2, \$f2 # Convert \$t2 from int to float \$f2 [004001a4] 46001083 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] 46001303 mov.s \$f12, \$f2 ; 140: mov.s \$f12, \$f2 # Print the \$f2 [004001b0] 34020002 ori \$2, \$0, 2 ; 141: li \$v0, 2 [004001b4] 34240082 ori \$8, \$1, 130 [strSpace] [004001b8] 3c011001 lui \$1, 4097 [strSpace] ; 144: la \$a0, strSpace [004001bc] 34240082 ori \$2, \$0, 4 ; 145: li \$v0, 4 [004001c4] 00000000 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 </pre>
FIR = 9800 FCSR = 0	Single Precision FG0 = 45cc7800 FG1 = 0 FG2 = 3bd7592b	Text	<pre>[00400154] 1140001f beg \$10, \$0, 124 [END_FUNC-0x00400154] [00400158] 000a2021 add \$4, \$0, \$10 ; 110: move \$a0, \$t2 # Print the a[i] [0040015c] 34020001 ori \$2, \$0, 1 ; 111: li \$v0, 1 [00400160] 00000000 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] 00000004 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] 34240084 ori \$8, \$1, 132 [arr] [00400184] 2210ffff addi \$16, \$16, -1 ; 123: sub \$s0, \$s0, 1 # \$s0 = \$s0 - 1 [00400188] 00100800 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(\$s0) ; 127: lw \$t3, 0(\$t0) # \$t3 = arr[\$s0] [00400194] 08100074 j 0x004001d0 [END_FUNC] ; 129: j END_FUNC [00400198] 8caa0000 lw \$10, 0(\$s5) ; 132: lw \$t2, 0(\$a1) # \$t2 = arr[i] [004001a0] 11400000 beg \$10, \$0, 52 [END_FUNC-0x0040019c] [004001a1] 448a1000 mtc1 \$10, \$f2 ; 135: mtc1 \$t2, \$f2 # Convert \$t2 from int to float \$f2 [004001a4] 46001083 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] 46001303 mov.s \$f12, \$f2 ; 140: mov.s \$f12, \$f2 # Print the \$f2 [004001b0] 34020002 ori \$2, \$0, 2 ; 141: li \$v0, 2 [004001b4] 34240082 ori \$8, \$1, 130 [strSpace] [004001b8] 3c011001 lui \$1, 4097 [strSpace] ; 144: la \$a0, strSpace [004001bc] 34240082 ori \$2, \$0, 4 ; 145: li \$v0, 4 [004001c4] 00000000 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 </pre>
FIR = 9800 FCSR = 0	Single Precision FG0 = 45cc7800 FG1 = 0 FG2 = 3bd7592b	Text	<pre>[00400154] 1140001f beg \$10, \$0, 124 [END_FUNC-0x00400154] [00400158] 000a2021 add \$4, \$0, \$10 ; 110: move \$a0, \$t2 # Print the a[i] [0040015c] 34020001 ori \$2, \$0, 1 ; 111: li \$v0, 1 [00400160] 00000000 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] 00000004 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] 34240084 ori \$8, \$1, 132 [arr] [00400184] 2210ffff addi \$16, \$16, -1 ; 123: sub \$s0, \$s0, 1 # \$s0 = \$s0 - 1 [00400188] 00100800 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(\$s0) ; 127: lw \$t3, 0(\$t0) # \$t3 = arr[\$s0] [00400194] 08100074 j 0x004001d0 [END_FUNC] ; 129: j END_FUNC [00400198] 8caa0000 lw \$10, 0(\$s5) ; 132: lw \$t2, 0(\$a1) # \$t2 = arr[i] [004001a0] 11400000 beg \$10, \$0, 52 [END_FUNC-0x0040019c] [004001a1] 448a1000 mtc1 \$10, \$f2 ; 135: mtc1 \$t2, \$f2 # Convert \$t2 from int to float \$f2 [004001a4] 46001083 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] 46001303 mov.s \$f12, \$f2 ; 140: mov.s \$f12, \$f2 # Print the \$f2 [004001b0] 34020002 ori \$2, \$0, 2 ; 141: li \$v0, 2 [004001b4] 34240082 ori \$8, \$1, 130 [strSpace] [004001b8] 3c011001 lui \$1, 4097 [strSpace] ; 144: la \$a0, strSpace [004001bc] 34240082 ori \$2, \$0, 4 ; 145: li \$v0, 4 [004001c4] 00000000 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 </pre>
FIR = 9800 FCSR = 0	Single Precision FG0 = 45cc7800 FG1 = 0 FG2 = 3bd7592b	Text	<pre>[00400154] 1140001f beg \$10, \$0, 124 [END_FUNC-0x00400154] [00400158] 000a2021 add \$4, \$0, \$10 ; 110: move \$a0, \$t2 # Print the a[i] [0040015c] 34020001 ori \$2, \$0, 1 ; 111: li \$v0, 1 [00400160] 00000000 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] 00000004 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] 34240084 ori \$8, \$1, 132 [arr] [00400184] 2210ffff addi \$16, \$16, -1 ; 123: sub \$s0, \$s0, 1 # \$s0 = \$s0 - 1 [00400188] 00100800 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(\$s0) ; 127: lw \$t3, 0(\$t0) # \$t3 = arr[\$s0] [00400194] 08100074 j 0x004001d0 [END_FUNC] ; 129: j END_FUNC [00400198] 8caa0000 lw \$10, 0(\$s5) ; 132: lw \$t2, 0(\$a1) # \$t2 = arr[i] [004001a0] 11400000 beg \$10, \$0, 52 [END_FUNC-0x0040019c] [004001a1] 448a1000 mtc1 \$10, \$f2 ; 135: mtc1 \$t2, \$f2 # Convert \$t2 from int to float \$f2 [004001a4] 46001083 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] 46001303 mov.s \$f12, \$f2 ; 140: mov.s \$f12, \$f2 # Print the \$f2 [004001b0] 34020002 ori \$2, \$0, 2 ; 141: li \$v0, 2 [004001b4] 34240082 ori \$8, \$1, 130 [strSpace] [004001b8] 3c011001 lui \$1, 4097 [strSpace] ; 144: la \$a0, strSpace [004001bc] 34240082 ori \$2, \$0, 4 ; 145: li \$v0, 4 [004001c4] 00000000 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 </pre>

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