CENG 3420 Ng Chi Hon 1155116317

Lab 1-1

Comment briefly explain my code

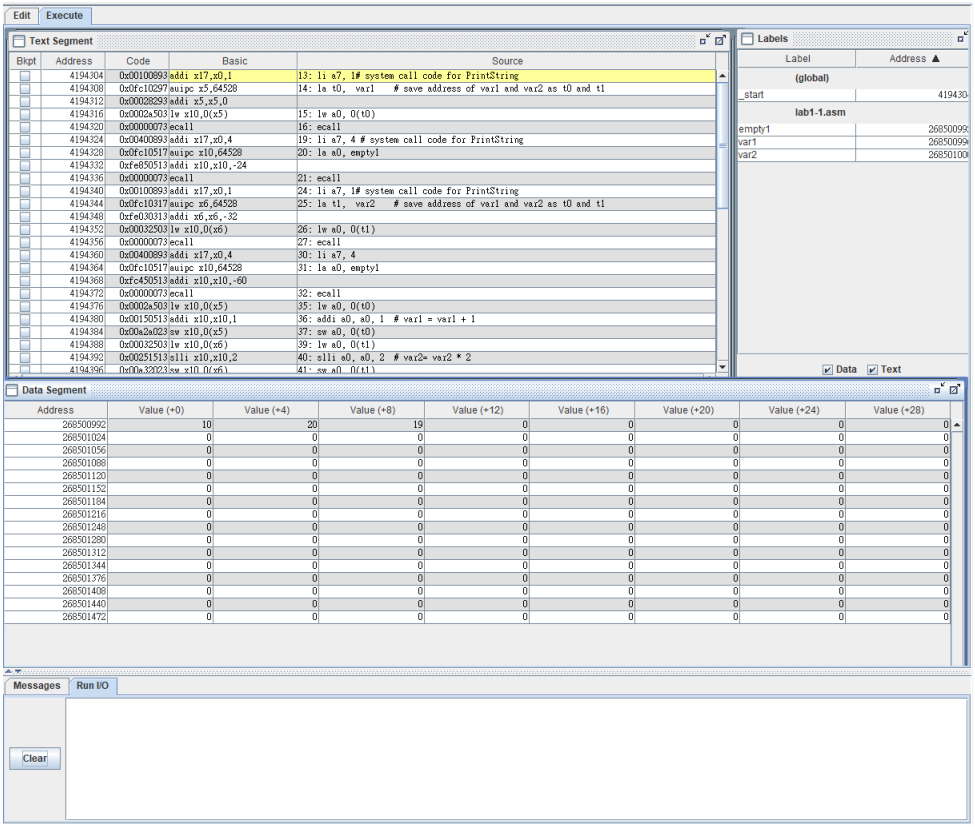
Edit Execute

lab1-1.asm

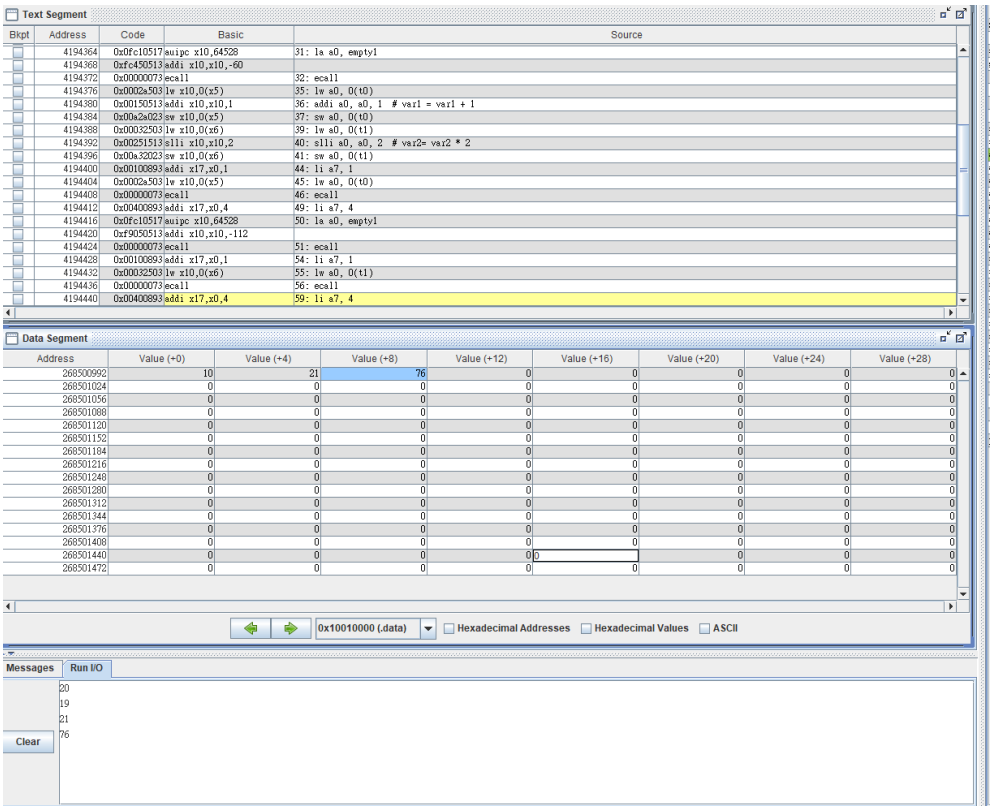
```
.globl _start

.data
empty1: .asciz "\n"
var1:   .word 20
var2: .word 19


.text
_start:

# print var1
li a7, 1# system call code for PrintString
la t0,  var1      # save address of var1 and var2 as t0 and t1
lw a0, 0(t0)
ecall

# print newline
li a7, 4 # system call code for PrintString
la a0, empty1
ecall

# print var2
li a7, 1# system call code for PrintString
la t1,  var2      # save address of var1 and var2 as t0 and t1
lw a0, 0(t1)
ecall

# print newline
li a7, 4
la a0, empty1
ecall

#modify var1,2 by load word and save word
lw a0, 0(t0)
addi a0, a0, 1   # var1 = var1 + 1
sw a0, 0(t0)

lw a0, 0(t1)
slli a0, a0, 2   # var2= var2 * 2
sw a0, 0(t1)

# print var1
li a7, 1
lw a0, 0(t0)
ecall

# print newline
li a7, 4
la a0, empty1
ecall

# print var2
li a7, 1
lw a0, 0(t1)
ecall

# print newline
li a7, 4
la a0, empty1
ecall

# swap var1 and var2 by load word and save word
lw t2, 0(t0)
lw t3, 0(t1)
sw t2, 0(t1)
sw t3, 0(t0)

# print var1
li a7, 1
lw a0, 0(t0)
ecall

# print newline
li a7, 4
la a0, empty1
ecall

# print var2
li a7, 1
lw a0, 0(t1)
ecall
```

Console Result:

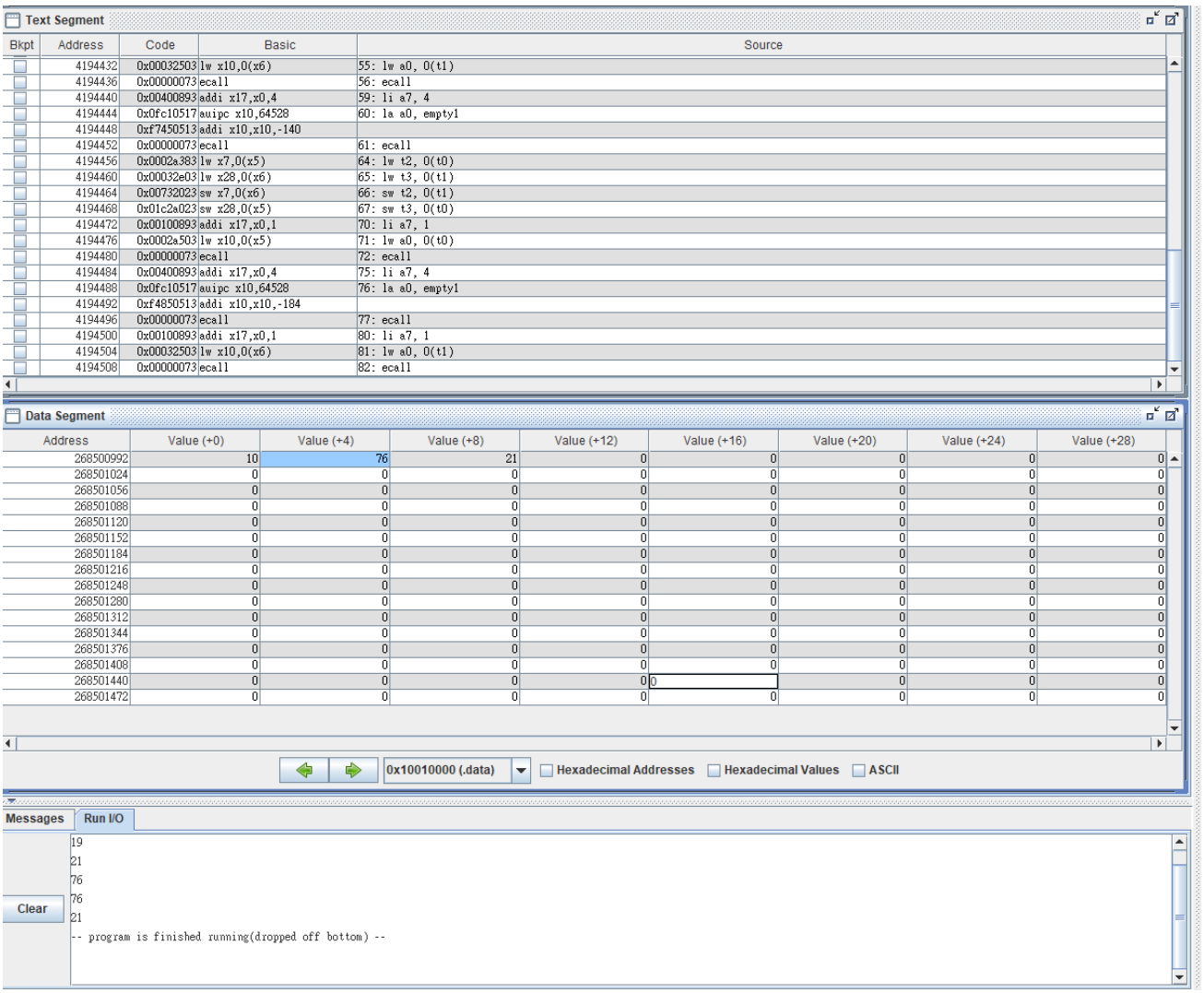
Original:



After add and multiply var1,var2:



After swapping:



Text Segment

Bkpt	Address	Code	Basic	Source
	4194432	0x00032503	lw x10,0(x6)	55: lw a0, 0(t1)
	4194436	0x00000073	ecall	56: ecall
	4194440	0x00400893	addi x17,x0,4	59: li a7, 4
	4194444	0x0fc10517	auipc x10,64528	60: la a0, empty1
	4194448	0xf7450513	addi x10,x10,-140	
	4194452	0x00000073	ecall	61: ecall
	4194456	0x0002a383	lw x7,0(x5)	64: lw t2, 0(t0)
	4194460	0x00032e03	lw x28,0(x6)	65: lw t3, 0(t1)
	4194464	0x00732023	sw x7,0(x6)	66: sw t2, 0(t1)
	4194468	0x01c2a023	sw x28,0(x5)	67: sw t3, 0(t0)
	4194472	0x00100893	addi x17,x0,1	70: li a7, 1
	4194476	0x0002a503	lw x10,0(x5)	71: lw a0, 0(t0)
	4194480	0x00000073	ecall	72: ecall
	4194484	0x00400893	addi x17,x0,4	75: li a7, 4
	4194488	0x0fc10517	auipc x10,64528	76: la a0, empty1
	4194492	0xf4850513	addi x10,x10,-184	
	4194496	0x00000073	ecall	77: ecall
	4194500	0x00100893	addi x17,x0,1	80: li a7, 1
	4194504	0x00032503	lw x10,0(x6)	81: lw a0, 0(t1)
	4194508	0x00000073	ecall	82: ecall

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	10	76	21	0	0	0	0	0
268501024	0	0	0	0	0	0	0	0
268501056	0	0	0	0	0	0	0	0
268501088	0	0	0	0	0	0	0	0
268501120	0	0	0	0	0	0	0	0
268501152	0	0	0	0	0	0	0	0
268501184	0	0	0	0	0	0	0	0
268501216	0	0	0	0	0	0	0	0
268501248	0	0	0	0	0	0	0	0
268501280	0	0	0	0	0	0	0	0
268501312	0	0	0	0	0	0	0	0
268501344	0	0	0	0	0	0	0	0
268501376	0	0	0	0	0	0	0	0
268501408	0	0	0	0	0	0	0	0
268501440	0	0	0	0	0	0	0	0
268501472	0	0	0	0	0	0	0	0

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Messages Run I/O

Clear

```
19
21
76
76
21
-- program is finished running(dropped off bottom) --
```

Lab1-2

In lab1-2, we have s1 to save array length, s2 to save i for for loop use and s3 for saving pivot current.

At start, swap 3rd element and last element for easy comparison

start for loop:
save t0 = array[0+i]

compare array[0+i] and s3 , if array[0+i] < s3, jump smallSwap

smallSwap :
load array[0+i] address and array[0+pivot] address and swap them after swap. Jump to swapExit

swapExit:
i = i +1
if i = array length then jump to exit
else back to loop

Exit:
After check array[0] to array[0+arraylength-1] we check pivot suppose position, and finally swap array[arraylength] (3rd element) with array[pivot] (3rd element suppose location)

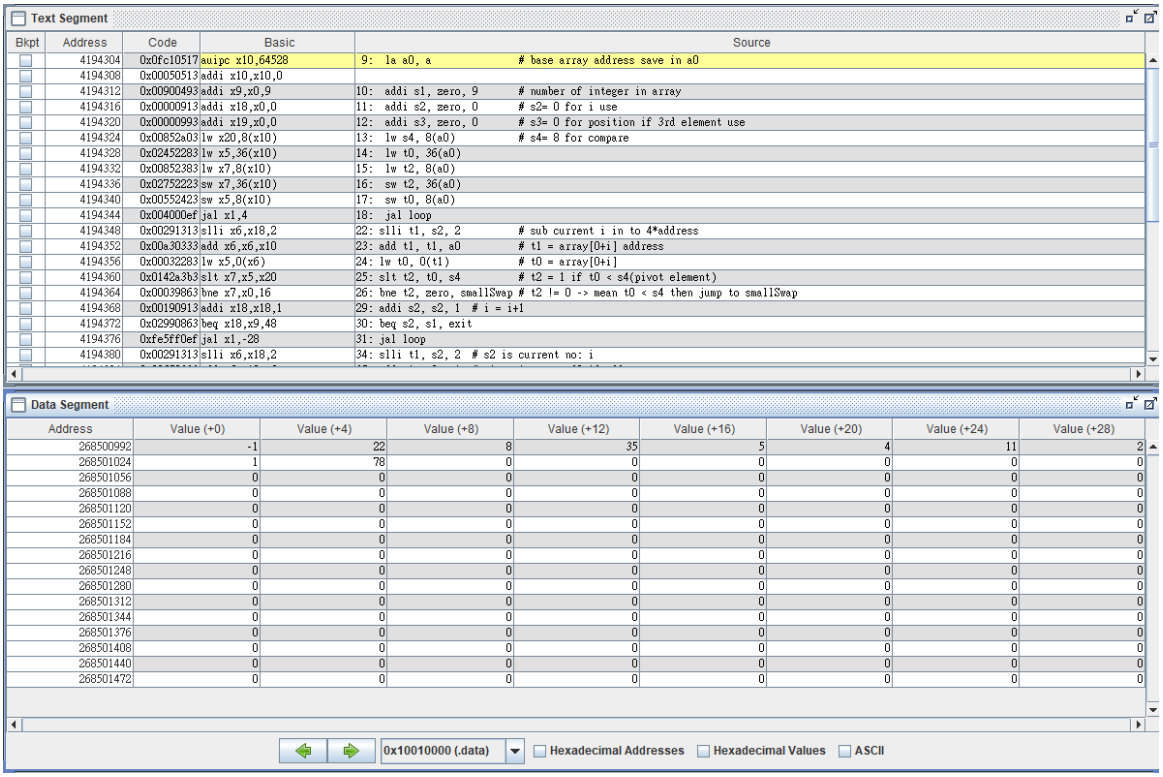
Edit | Execute

lab1-1.asm | lab1-2.asm* | lab1-3.asm

```
.globl _start


.data
a: .word -1 22 8 35 5 4 11 2 1 78

  .text
_start:
la a0, a                # base array address save in a0
addi s1, zero, 9        # number of integer in array
addi s2, zero, 0        # s2= 0 for i use
addi s3, zero, 0        # s3= 0 for position if 3rd element use
lw s4, 8(a0)            # s4= 8 for compare
lw t0, 36(a0)
lw t2, 8(a0)
sw t2, 36(a0)
sw t0, 8(a0)
jal loop

# LOOP (i = 0)-> SMALLSWAP -> SWAPEXIT -> LOOP (i = 1)-> .... -> LOOP (i = arrayLength)-> EXIT
loop:
slli t1, s2, 2          # sub current i in to 4*address
add t1, t1, a0          # t1 = array[0+i] address
lw t0, 0(t1)            # t0 = array[0+i]
slt t2, t0, s4          # t2 = 1 if t0 < s4(pivot element)
bne t2, zero, smallSwap # t2 != 0 -> mean t0 < s4 then jump to smallSwap

SwapExit:
addi s2, s2, 1  # i = i+1
beq s2, s1, exit
jal loop

smallSwap:
slli t1, s2, 2  # s2 is current no: i
add t1, a0, t1  # t1 = t1 = array[0+i] address
slli t3, s3, 2  # s3 is partictian no
add t3, a0, t3  # t3 = array[0+P]
lw t0, 0(t1)
lw t2, 0(t3)
sw t2, 0(t1)
sw t0, 0(t3)
addi s3, s3, 1
jal SwapExit

exit:
slli t1, s2, 2  # s2 is current no: i
add t1, a0, t1
slli t3, s3, 2  # s2 is partictian no
add t3, a0, t3
lw t0, 0(t1)
lw t2, 0(t3)
sw t2, 0(t1)
sw t0, 0(t3)
```

Console Result:

Original:



Text Segment

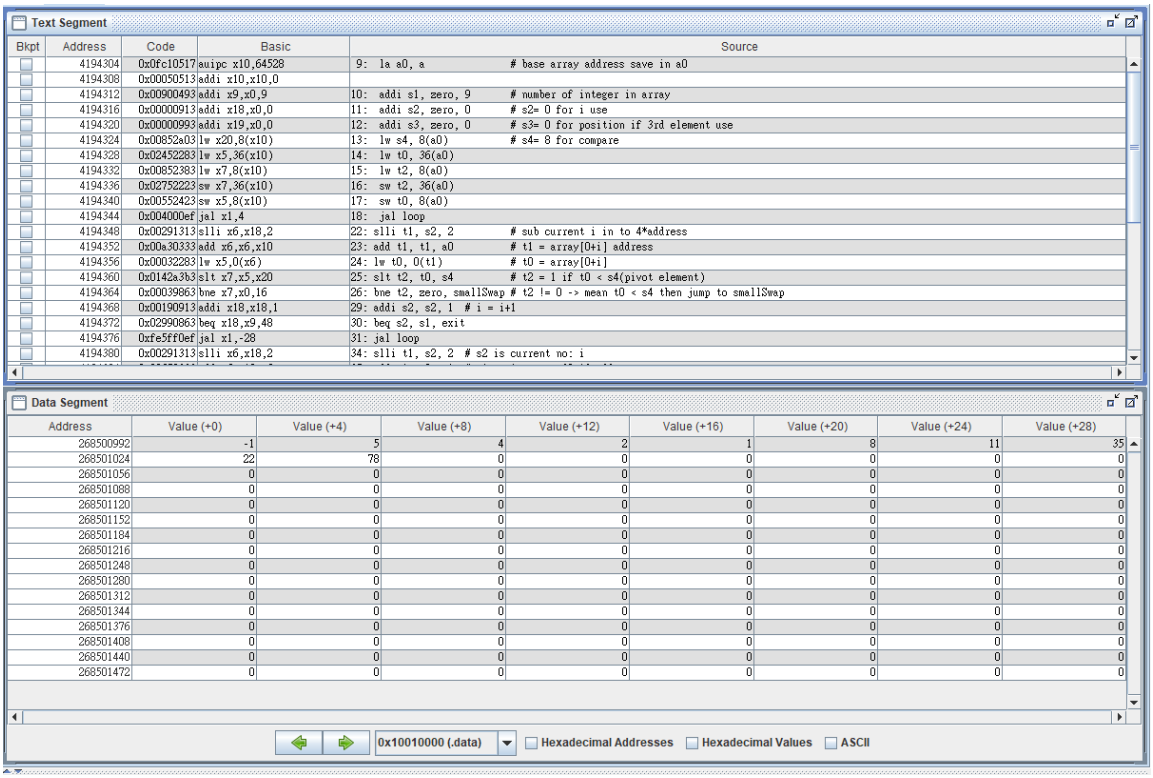
Bkpt	Address	Code	Basic	Source
	4194304	0x0fc10517	auipc x10,64528	9: la a0, a # base array address save in a0
	4194308	0x00050513	addi x10,x10,0	
	4194312	0x00900493	addi x9,x0,9	10: addi s1, zero, 9 # number of integer in array
	4194316	0x00000913	addi x18,x0,0	11: addi s2, zero, 0 # s2= 0 for i use
	4194320	0x00000993	addi x19,x0,0	12: addi s3, zero, 0 # s3= 0 for position if 3rd element use
	4194324	0x00852a03	lw x20,8(x10)	13: lw s4, 8(a0) # s4= 8 for compare
	4194328	0x02452283	lw x5,36(x10)	14: lw t0, 36(a0)
	4194332	0x00852383	lw x7,8(x10)	15: lw t2, 8(a0)
	4194336	0x02752223	sw x7,36(x10)	16: sw t2, 36(a0)
	4194340	0x00552423	sw x5,8(x10)	17: sw t0, 8(a0)
	4194344	0x004000ef	jal x1,4	18: jal loop
	4194348	0x00291313	slli x6,x18,2	22: slli t1, s2, 2 # sub current i in to 4*address
	4194352	0x00a30333	add x6,x6,x10	23: add t1, t1, a0 # t1 = array[0+i] address
	4194356	0x00032283	lw x5,0(x6)	24: lw t0, 0(t1) # t0 = array[0+i]
	4194360	0x0142a3b3	slt x7,x5,x20	25: slt t2, t0, s4 # t2 = 1 if t0 < s4(pivot element)
	4194364	0x00039863	bne x7,x0,16	26: bne t2, zero, smallSwap # t2 != 0 -> mean t0 < s4 then jump to smallSwap
	4194368	0x00190913	addi x18,x18,1	29: addi s2, s2, 1 # i = i+1
	4194372	0x02990863	beq x18,x9,48	30: beq s2, s1, exit
	4194376	0xfe5ff0ef	jal x1,-28	31: jal loop
	4194380	0x00291313	slli x6,x18,2	34: slli t1, s2, 2 # s2 is current no: i

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	-1	22	8	35	5	4	11	2
268501024	1	78	0	0	0	0	0	0
268501056	0	0	0	0	0	0	0	0
268501088	0	0	0	0	0	0	0	0
268501120	0	0	0	0	0	0	0	0
268501152	0	0	0	0	0	0	0	0
268501184	0	0	0	0	0	0	0	0
268501216	0	0	0	0	0	0	0	0
268501248	0	0	0	0	0	0	0	0
268501280	0	0	0	0	0	0	0	0
268501312	0	0	0	0	0	0	0	0
268501344	0	0	0	0	0	0	0	0
268501376	0	0	0	0	0	0	0	0
268501408	0	0	0	0	0	0	0	0
268501440	0	0	0	0	0	0	0	0
268501472	0	0	0	0	0	0	0	0

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

After partition:



Text Segment

Bkpt	Address	Code	Basic	Source
	4194304	0x0fc10517	auipc x10,64528	9: la a0, a # base array address save in a0
	4194308	0x00050513	addi x10,x10,0	
	4194312	0x00900493	addi x9,x0,9	10: addi s1, zero, 9 # number of integer in array
	4194316	0x00000913	addi x18,x0,0	11: addi s2, zero, 0 # s2= 0 for i use
	4194320	0x00000993	addi x19,x0,0	12: addi s3, zero, 0 # s3= 0 for position if 3rd element use
	4194324	0x00852a03	lw x20,8(x10)	13: lw s4, 8(a0) # s4= 8 for compare
	4194328	0x02452283	lw x5,36(x10)	14: lw t0, 36(a0)
	4194332	0x00852383	lw x7,8(x10)	15: lw t2, 8(a0)
	4194336	0x02752223	sw x7,36(x10)	16: sw t2, 36(a0)
	4194340	0x00552423	sw x5,8(x10)	17: sw t0, 8(a0)
	4194344	0x004000ef	jal x1,4	18: jal loop
	4194348	0x00291313	slli x6,x18,2	22: slli t1, s2, 2 # sub current i in to 4*address
	4194352	0x00a30333	add x6,x6,x10	23: add t1, t1, a0 # t1 = array[0+i] address
	4194356	0x00032283	lw x5,0(x6)	24: lw t0, 0(t1) # t0 = array[0+i]
	4194360	0x0142a3b3	slt x7,x5,x20	25: slt t2, t0, s4 # t2 = 1 if t0 < s4(pivot element)
	4194364	0x00039863	bne x7,x0,16	26: bne t2, zero, smallSwap # t2 != 0 -> mean t0 < s4 then jump to smallSwap
	4194368	0x00190913	addi x18,x18,1	29: addi s2, s2, 1 # i = i+1
	4194372	0x02990863	beq x18,x9,48	30: beq s2, s1, exit
	4194376	0xfe5ff0ef	jal x1,-28	31: jal loop
	4194380	0x00291313	slli x6,x18,2	34: slli t1, s2, 2 # s2 is current no: i

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	-1	5	4	2	1	8	11	35
268501024	22	78	0	0	0	0	0	0
268501056	0	0	0	0	0	0	0	0
268501088	0	0	0	0	0	0	0	0
268501120	0	0	0	0	0	0	0	0
268501152	0	0	0	0	0	0	0	0
268501184	0	0	0	0	0	0	0	0
268501216	0	0	0	0	0	0	0	0
268501248	0	0	0	0	0	0	0	0
268501280	0	0	0	0	0	0	0	0
268501312	0	0	0	0	0	0	0	0
268501344	0	0	0	0	0	0	0	0
268501376	0	0	0	0	0	0	0	0
268501408	0	0	0	0	0	0	0	0
268501440	0	0	0	0	0	0	0	0
268501472	0	0	0	0	0	0	0	0

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Lab1-3

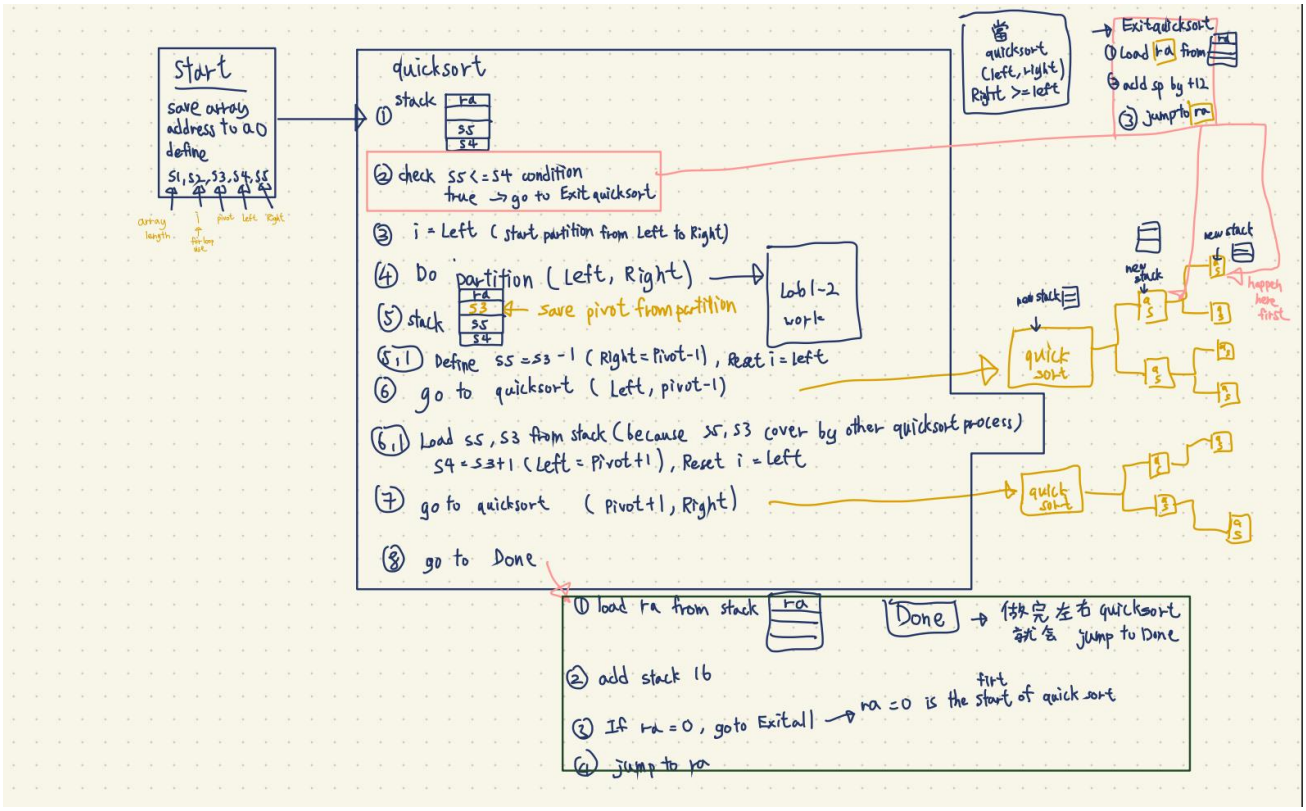
QuickSort

lab1-1.asm | lab1-2.asm | lab1-3.asm

```
.globl _start

.data
a: .word -1 22 8 35 5 4 11 2 1 78

# s1 = array length s2 = i s3 = pivot s4 = left s5 = right
.text
_start:
la a0, a
addi s1, zero, 9        # number of integer in array
addi s2, zero, 0        # s2= 0 for i use foploop

addi s3, zero, 0        # s3= 0 for pivot
addi s4, zero, 0        # s4= 0 for left
add s5, zero, s1        # s5= 0 for Right (begining default as Rightest 1 = s1)

# quicksort _1-> partition -> quicksort _1 -> quicksort _1_L -> partition -> quicksort _1_L -> quicksort _1_L_L ->.....
quickSort:
addi sp, sp, -16
sw ra, 12(sp)
sw s5, 4(sp)
sw s4, 0(sp)
ble s5,s4,exitquickSort
add s2, zero, s4
jal  partition
# addi s7, zero, 1000 # check point 1 #ok
sw s3, 8(sp)            # save pivot point after partition

addi s5, s3, -1         # Right = Pivot -1
add s3, zero, s4        # reset pivot point for partition
jal quickSort           # quickSort( Left, Pivot -1)


lw s5, 4(sp)            # load back Righr and pivot Value from stack
lw s3, 8(sp)
addi s4, s3, 1          # Left = pivot + 1
add s3, zero, s4        # reset pivot point for partition
jal quickSort           # quickSort(pivot + 1, Right)
jal Done
```

```
#parttitopn
partition:
addi sp, sp, -4
sw ra, 0(sp)              # save return address to quicksort function
partitionLoop:
slli t1, s2, 2            # sub current i in to 4*address
add t1, t1, a0            # t1 = array[0+i] address
lw t0, 0(t1)              # t0 = array[0+i]
slli t3, s5, 2
add t3, t3, a0
lw t5, 0(t3)              # t5 = pivot element
slt t2, t0, t5            # t2 = 1 if t0 < t5(pivot element)
bne t2, zero, smallSwap # t2 != 0 -> mean t0 < pivot value then jump to smallSwap

SwapExit:
addi s2, s2, 1  # i = i+1
beq s2, s5, exit
jal partitionLoop
exit:
slli t1, s2, 2  # s2 is current no: i
add t1, a0, t1
slli t3, s3, 2  # s2 is partictian no
add t3, a0, t3
lw t0, 0(t1)
lw t2, 0(t3)
sw t2, 0(t1)
sw t0, 0(t3)
lw ra, 0(sp)
addi sp, sp, 4
jr ra

smallSwap:
slli t1, s2, 2  # s2 is current no: i
add t1, a0, t1  # t1 = t1 = array[0+i] address
slli t3, s3, 2  # s3 is partictian no
add t3, a0, t3  # t3 = array[0+P]
lw t0, 0(t1)
lw t2, 0(t3)
sw t2, 0(t1)
sw t0, 0(t3)
addi s3, s3, 1
jal SwapExit

#parttitopn Done
```

```
exitquickSort:
lw ra, 12(sp)
addi sp, sp, 16
jr ra

Done:
lw ra, 12(sp)
addi sp, sp, 16
beq ra, zero, Exitall              # prevent ra == 0 when start of quick sort
jr ra

Exitall:
```



Console Result:

Original:

Text Segment

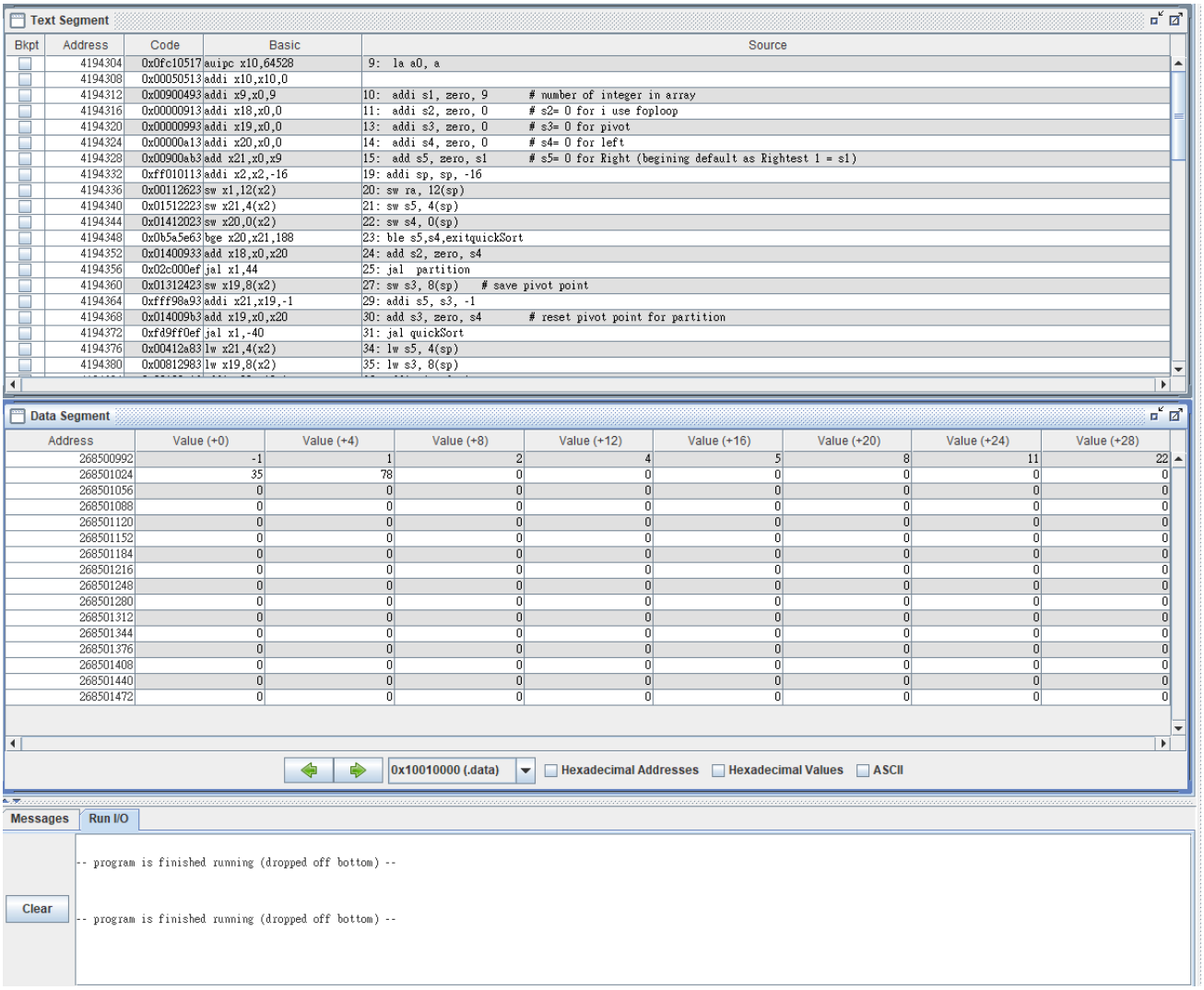
Bkpt	Address	Code	Basic	Source
	4194304	0x0fc10517	auipc x10,64528	9: la a0, a
	4194308	0x00050513	addi x10,x10,0	
	4194312	0x00900493	addi x9,x0,9	10: addi s1, zero, 9 # number of integer in array
	4194316	0x00000913	addi x18,x0,0	11: addi s2, zero, 0 # s2= 0 for i use foploop
	4194320	0x00000993	addi x19,x0,0	13: addi s3, zero, 0 # s3= 0 for pivot
	4194324	0x00000a13	addi x20,x0,0	14: addi s4, zero, 0 # s4= 0 for left
	4194328	0x00900ab3	add x21,x0,x9	15: add s5, zero, s1 # s5= 0 for Right (begining default as Rightest 1 = s1)
	4194332	0xff010113	addi x2,x2,-16	19: addi sp, sp, -16
	4194336	0x00112623	sw x1,12(x2)	20: sw ra, 12(sp)
	4194340	0x01512223	sw x21,4(x2)	21: sw s5, 4(sp)
	4194344	0x01412023	sw x20,0(x2)	22: sw s4, 0(sp)
	4194348	0x0b5a5e63	bge x20,x21,188	23: ble s5,s4,exitquickSort
	4194352	0x01400933	add x18,x0,x20	24: add s2, zero, s4
	4194356	0x02c000ef	jal x1,44	25: jal partition
	4194360	0x01312423	sw x19,8(x2)	27: sw s3, 8(sp) # save pivot point
	4194364	0xfff98a93	addi x21,x19,-1	29: addi s5, s3, -1
	4194368	0x014009b3	add x19,x0,x20	30: add s3, zero, s4 # reset pivot point for partition
	4194372	0xfd9ff0ef	jal x1,-40	31: jal quickSort
	4194376	0x00412a83	lw x21,4(x2)	34: lw s5, 4(sp)
	4194380	0x00812983	lw x19,8(x2)	35: lw s3, 8(sp)

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	-1	22	8	35	5	4	11	2
268501024	1	78	0	0	0	0	0	0
268501056	0	0	0	0	0	0	0	0
268501088	0	0	0	0	0	0	0	0
268501120	0	0	0	0	0	0	0	0
268501152	0	0	0	0	0	0	0	0
268501184	0	0	0	0	0	0	0	0
268501216	0	0	0	0	0	0	0	0
268501248	0	0	0	0	0	0	0	0
268501280	0	0	0	0	0	0	0	0
268501312	0	0	0	0	0	0	0	0
268501344	0	0	0	0	0	0	0	0
268501376	0	0	0	0	0	0	0	0
268501408	0	0	0	0	0	0	0	0
268501440	0	0	0	0	0	0	0	0
268501472	0	0	0	0	0	0	0	0

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

After QuickSort:



Text Segment

Bkpt	Address	Code	Basic	Source
	4194304	0x0fc10517	auipc x10,64528	9: la a0, a
	4194308	0x00050513	addi x10,x10,0	
	4194312	0x00900493	addi x9,x0,9	10: addi s1, zero, 9 # number of integer in array
	4194316	0x00000913	addi x18,x0,0	11: addi s2, zero, 0 # s2= 0 for i use foploop
	4194320	0x00000993	addi x19,x0,0	13: addi s3, zero, 0 # s3= 0 for pivot
	4194324	0x00000a13	addi x20,x0,0	14: addi s4, zero, 0 # s4= 0 for left
	4194328	0x00900ab3	add x21,x0,x9	15: add s5, zero, s1 # s5= 0 for Right (begining default as Rightest 1 = s1)
	4194332	0xff010113	addi x2,x2,-16	19: addi sp, sp, -16
	4194336	0x00112623	sw x1,12(x2)	20: sw ra, 12(sp)
	4194340	0x01512223	sw x21,4(x2)	21: sw s5, 4(sp)
	4194344	0x01412023	sw x20,0(x2)	22: sw s4, 0(sp)
	4194348	0x0b5a5e63	bge x20,x21,188	23: ble s5,s4,exitquickSort
	4194352	0x01400933	add x18,x0,x20	24: add s2, zero, s4
	4194356	0x02c000ef	jal x1,44	25: jal partition
	4194360	0x01312423	sw x19,8(x2)	27: sw s3, 8(sp) # save pivot point
	4194364	0xfff98a93	addi x21,x19,-1	29: addi s5, s3, -1
	4194368	0x014009b3	add x19,x0,x20	30: add s3, zero, s4 # reset pivot point for partition
	4194372	0xfd9ff0ef	jal x1,-40	31: jal quickSort
	4194376	0x00412a83	lw x21,4(x2)	34: lw s5, 4(sp)
	4194380	0x00812983	lw x19,8(x2)	35: lw s3, 8(sp)

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	-1	1	2	4	5	8	11	22
268501024	35	78	0	0	0	0	0	0
268501056	0	0	0	0	0	0	0	0
268501088	0	0	0	0	0	0	0	0
268501120	0	0	0	0	0	0	0	0
268501152	0	0	0	0	0	0	0	0
268501184	0	0	0	0	0	0	0	0
268501216	0	0	0	0	0	0	0	0
268501248	0	0	0	0	0	0	0	0
268501280	0	0	0	0	0	0	0	0
268501312	0	0	0	0	0	0	0	0
268501344	0	0	0	0	0	0	0	0
268501376	0	0	0	0	0	0	0	0
268501408	0	0	0	0	0	0	0	0
268501440	0	0	0	0	0	0	0	0
268501472	0	0	0	0	0	0	0	0

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Messages Run I/O

Clear

-- program is finished running (dropped off bottom) --

-- program is finished running (dropped off bottom) --