CENG 3420 Lab1 Report

Name: [redacted] SID: [redacted]

Lab1.1

Step by step algorithm:

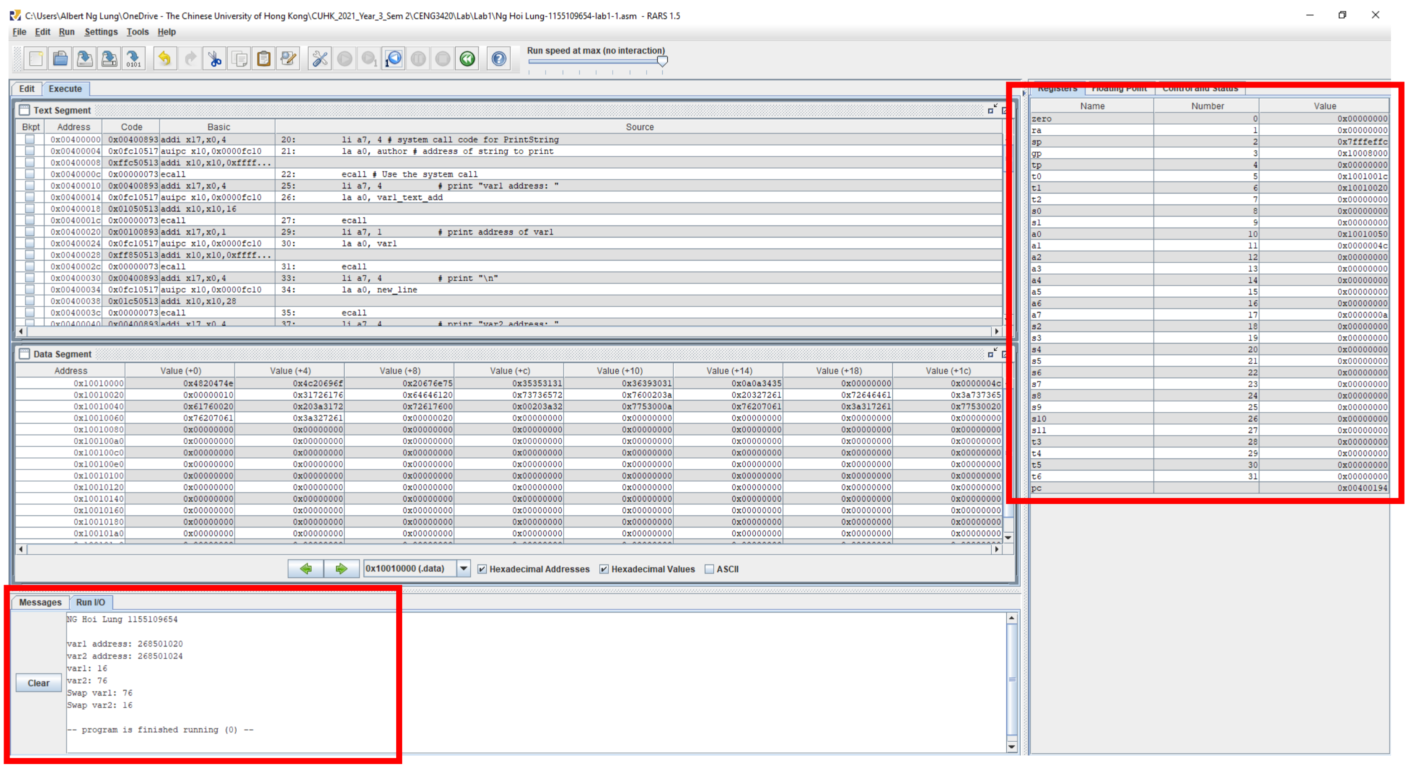
First of all, I need to define two variable one is var1 and the other one is var2 which is stored with 15 and 19 respectively. After that, the program will print the address of them which is using la a0, var1 and var2 to print the address with address 268501020, and 268501024. Then, I use addi to increase var1 by 1 and use li t0, 4 and mul a0, a0, t0 to load the imm 4 and multiply with the var2. After that we will get 16 and 76. Finally, we need to swap the two number which var1 is 16 and var2 is 76. I use lw to load the word to the address and la for remember the address. After that sw to store back the word to the remember address.

Main Code:

Print Address: Add and Multiply: Swap:

[redacted]

Console results:


C:\Users\Albert Ng Lung\OneDrive - The Chinese University of Hong Kong\CUHK_2021_Year_3_Sem 2\CENG3420\Lab\Lab1\Ng Hoi Lung-1155109654-lab1-1.asm - RARS 1.5

File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment

```
Bkpt Address    Code       Basic                       Source
     0x00400000 0x00400893 addi x17,x0,4           20:     li a7, 4 # system call code for PrintString
     0x00400004 0x0fc10517 auipc x10,0x0000fc10    21:     la a0, author # address of string to print
     0x00400008 0xffc50513 addi x10,x10,0xffff...
     0x0040000c 0x00000073 ecall                   22:     ecall # Use the system call
     0x00400010 0x00400893 addi x17,x0,4           25:     li a7, 4          # print "var1 address: "
     0x00400014 0x0fc10517 auipc x10,0x0000fc10    26:     la a0, var1_text_add
     0x00400018 0x01050513 addi x10,x10,16
     0x0040001c 0x00000073 ecall                   27:     ecall
     0x00400020 0x00100893 addi x17,x0,1           29:     li a7, 1          # print address of var1
     0x00400024 0x0fc10517 auipc x10,0x0000fc10    30:     la a0, var1
     0x00400028 0xff850513 addi x10,x10,0xffff...
     0x0040002c 0x00000073 ecall                   31:     ecall
     0x00400030 0x00400893 addi x17,x0,4           33:     li a7, 4          # print "\n"
     0x00400034 0x0fc10517 auipc x10,0x0000fc10    34:     la a0, new_line
     0x00400038 0x01c50513 addi x10,x10,28
     0x0040003c 0x00000073 ecall                   35:     ecall
     0x00400040 0x00400893 addi x17,x0,4           37:     li a7, 4          # print "var2 address: "
```

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x4820474e	0x4c20696f	0x20676e75	0x35353131	0x36393031	0x0a0a3435	0x00000000	0x0000004c
0x10010020	0x00000010	0x31726176	0x64646120	0x73736572	0x7600203a	0x20327261	0x72646461	0x3a737365
0x10010040	0x61760020	0x203a3172	0x72617600	0x00203a32	0x7753000a	0x76207061	0x3a317261	0x77530020
0x10010060	0x76207061	0x3a327261	0x00000020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) [✓] Hexadecimal Addresses [✓] Hexadecimal Values [ ] ASCII

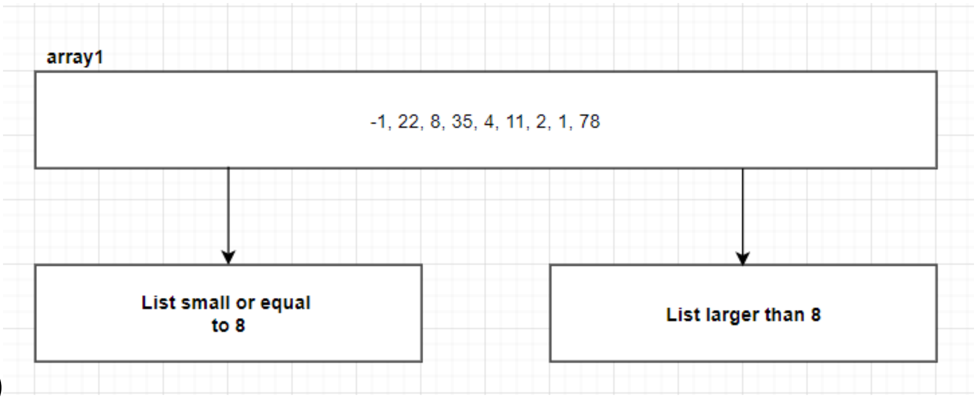
Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x1001001c
t1	6	0x10010020
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x10010050
a1	11	0x0000004c
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x0000000a
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400194

```
NG Hoi Lung 1155109654

var1 address: 268501020
var2 address: 268501024
var1: 16
var2: 76
Swap var1: 76
Swap var2: 16

-- program is finished running (0) --
```

Lab1.2

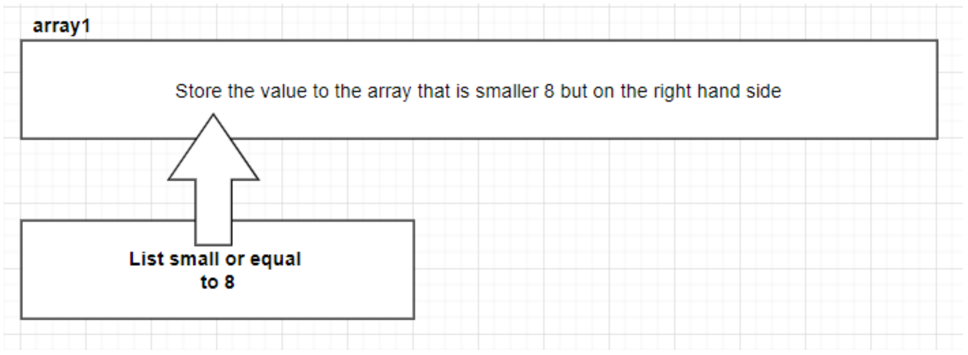
Step by step algorithm:

In this lab, 8 is the middle value the left-hand side will have -1, 5, 4, 2, 1 and the right-hand side will have 22, 35,11, 78 which requirement the lab requirement. The method I am using will be shown in the graph below:



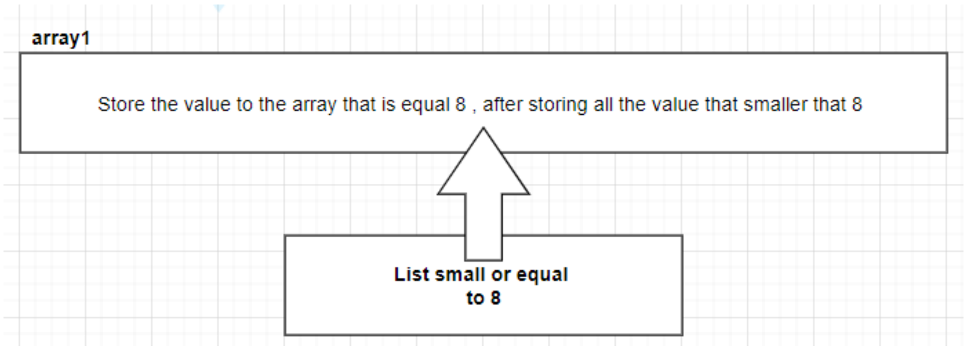
1)

I am going to separate array1 which two list which is the list that small and equal to third element 8 and the list larger than 8.


array1
Store the value to the array that is smaller 8 but on the right hand side
List small or equal to 8

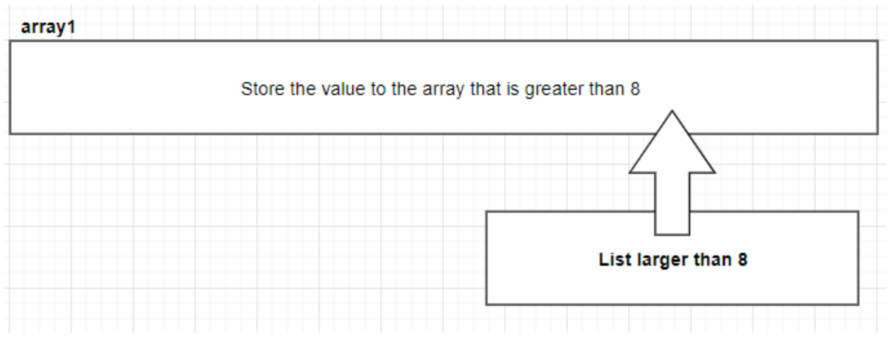
2)

First, I store the value that smaller than the third element 8.


array1
Store the value to the array that is equal 8 , after storing all the value that smaller that 8
List small or equal to 8

3)

After storing all the smaller value, than we can store the third element of 8 to the array1.



4)

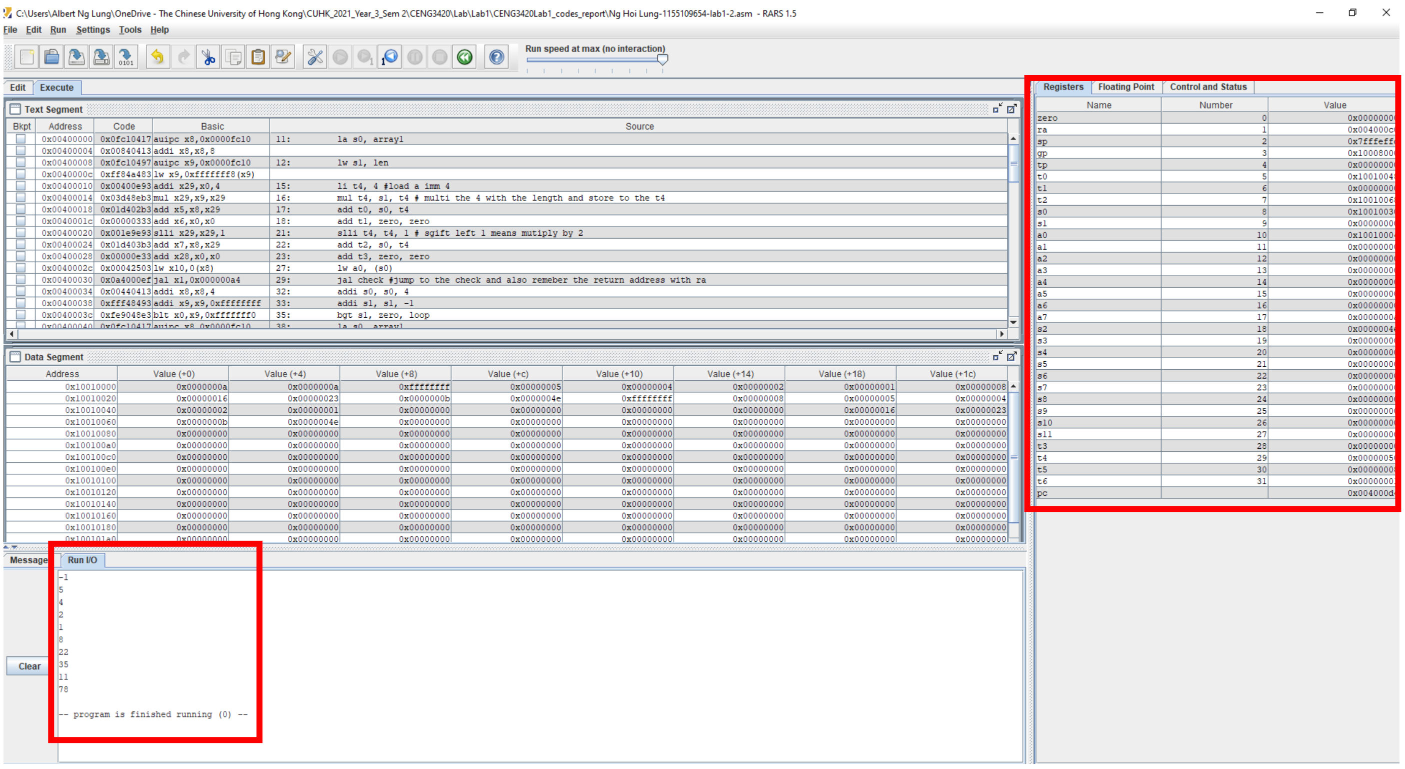
Finally, we store the remain element that is smaller than the third element of 8 to the array1.

At last the array1 will be replaced by the new arrangement to fit the requirement.

Console results:

Input:
```
array1: .word -1 22 8 35 5 4 11 2 1 78
```

Output:



C:\Users\Albert Ng Lung\OneDrive - The Chinese University of Hong Kong\CUHK_2021_Year_3_Sem 2\CENG3420\Lab\Lab1\CENG3420Lab1_codes_report\Ng Hoi Lung-1155109654-lab1-2.asm - RARS 1.5

File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Message Run I/O
Clear

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x0fc10417	auipc x8,0x0000fc10	11: la s0, array1
	0x00400004	0x00840413	addi x8,x8,8	
	0x00400008	0x0fc10497	auipc x9,0x0000fc10	12: lw s1, len
	0x0040000c	0xff84a483	lw x9,0xfffffff8(x9)	
	0x00400010	0x00400e93	addi x29,x0,4	15: li t4, 4 #load a imm 4
	0x00400014	0x03d48eb3	mul x29,x9,x29	16: mul t4, s1, t4 # multi the 4 with the length and store to the t4
	0x00400018	0x01d402b3	add x5,x8,x29	17: add t0, s0, t4
	0x0040001c	0x00000333	add x6,x0,x0	18: add t1, zero, zero
	0x00400020	0x001e9e93	slli x29,x29,1	21: slli t4, t4, 1 # sgift left 1 means mutiply by 2
	0x00400024	0x01d403b3	add x7,x8,x29	22: add t2, s0, t4
	0x00400028	0x00000e33	add x28,x0,x0	23: add t3, zero, zero
	0x0040002c	0x00042503	lw x10,0(x8)	27: lw a0, (s0)
	0x00400030	0x0a4000ef	jal x1,0x000000a4	29: jal check #jump to the check and also remeber the return address with ra
	0x00400034	0x00440413	addi x8,x8,4	32: addi s0, s0, 4
	0x00400038	0xfff48493	addi x9,x9,0xffffffff	33: addi s1, s1, -1
	0x0040003c	0xfe9048e3	blt x0,x9,0xfffffff0	35: bgt s1, zero, loop
	0x00400040	0x0fc10417	auipc x8,0x0000fc10	38: la s0, array1

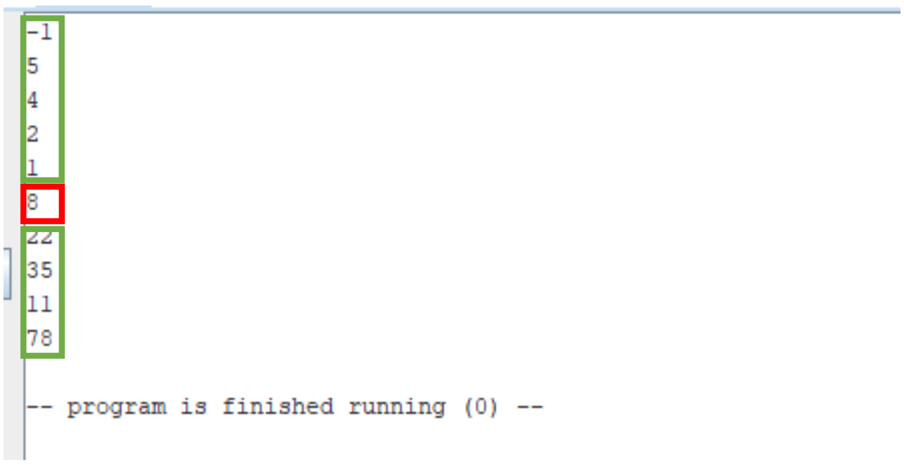
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x0000000a	0x0000000a	0xffffffff	0x00000005	0x00000004	0x00000002	0x00000001	0x00000008
0x10010020	0x00000016	0x00000023	0x0000000b	0x0000004e	0xffffffff	0x00000008	0x00000005	0x00000004
0x10010040	0x00000002	0x00000001	0x00000000	0x00000000	0x00000000	0x00000000	0x00000016	0x00000023
0x10010060	0x0000000b	0x0000004e	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Registers / Floating Point / Control and Status

Name	Number	Value
zero	0	0x0000000
ra	1	0x004000c
sp	2	0x7fffeff
gp	3	0x1000800
tp	4	0x0000000
t0	5	0x1001004
t1	6	0x0000000
t2	7	0x1001006
s0	8	0x1001003
s1	9	0x0000000
a0	10	0x1001000
a1	11	0x0000000
a2	12	0x0000000
a3	13	0x0000000
a4	14	0x0000000
a5	15	0x0000000
a6	16	0x0000000
a7	17	0x0000000
s2	18	0x0000004
s3	19	0x0000000
s4	20	0x0000000
s5	21	0x0000000
s6	22	0x0000000
s7	23	0x0000000
s8	24	0x0000000
s9	25	0x0000000
s10	26	0x0000000
s11	27	0x0000000
t3	28	0x0000000
t4	29	0x0000005
t5	30	0x0000000
t6	31	0x0000000
pc		0x004000d

```
-1
5
4
2
1
8
22
35
11
78

-- program is finished running (0) --
```



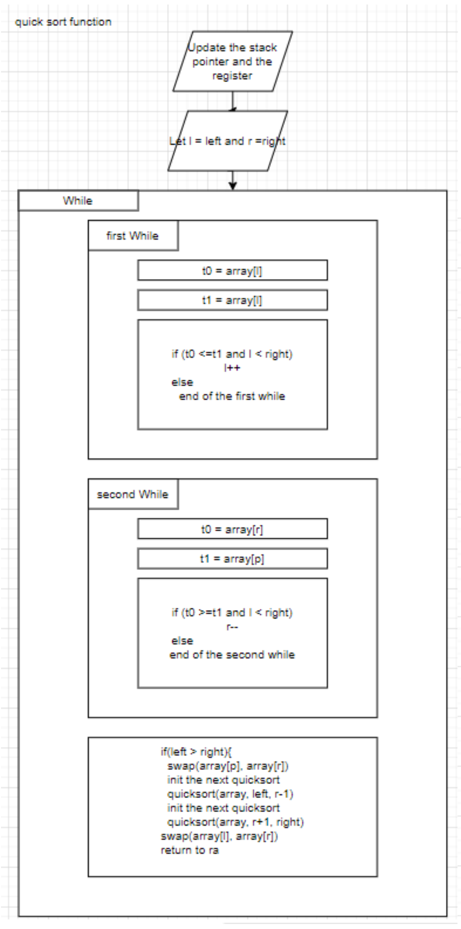
```
-1
5
4
2
1
8
22
35
11
78

-- program is finished running (0) --
```

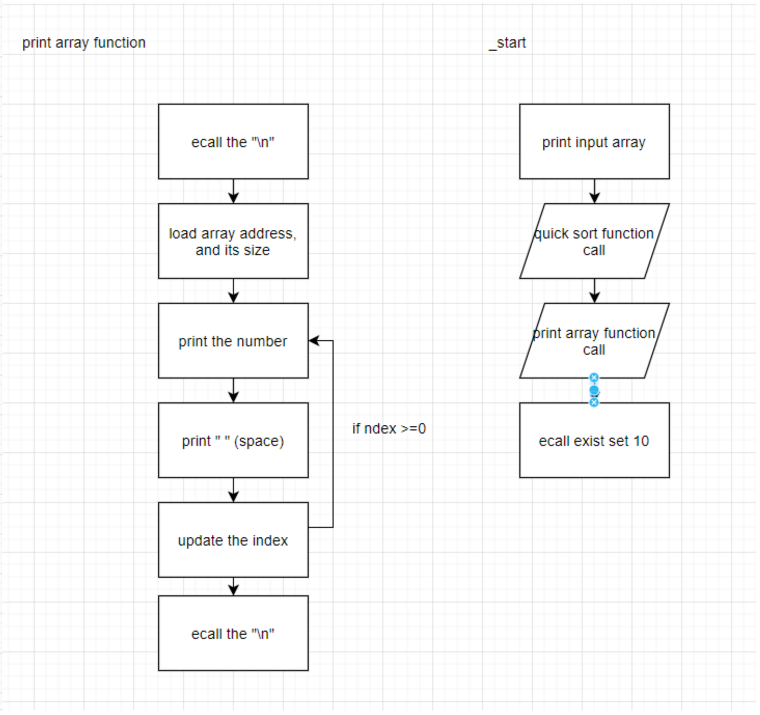
Lab1.3

Step by step algorithm:

Assembly implementation fuction:

(quick sort function) (print function) (_start)



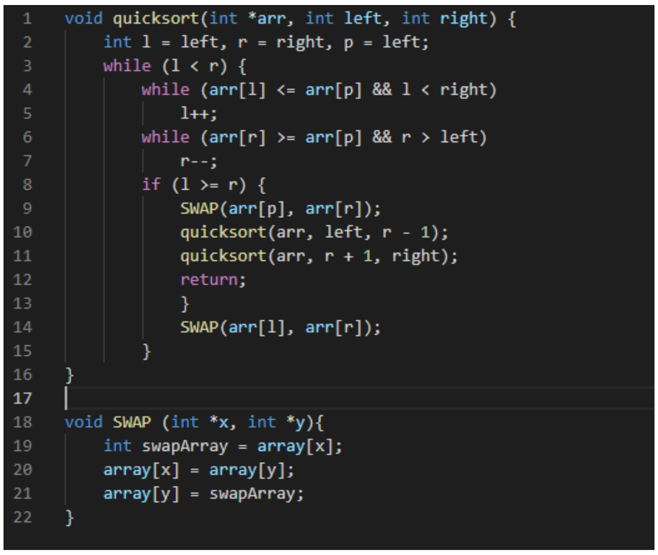


Assembly key code:

[redacted]

```
If_quick1_jump:

[redacted]

j Loop_
```

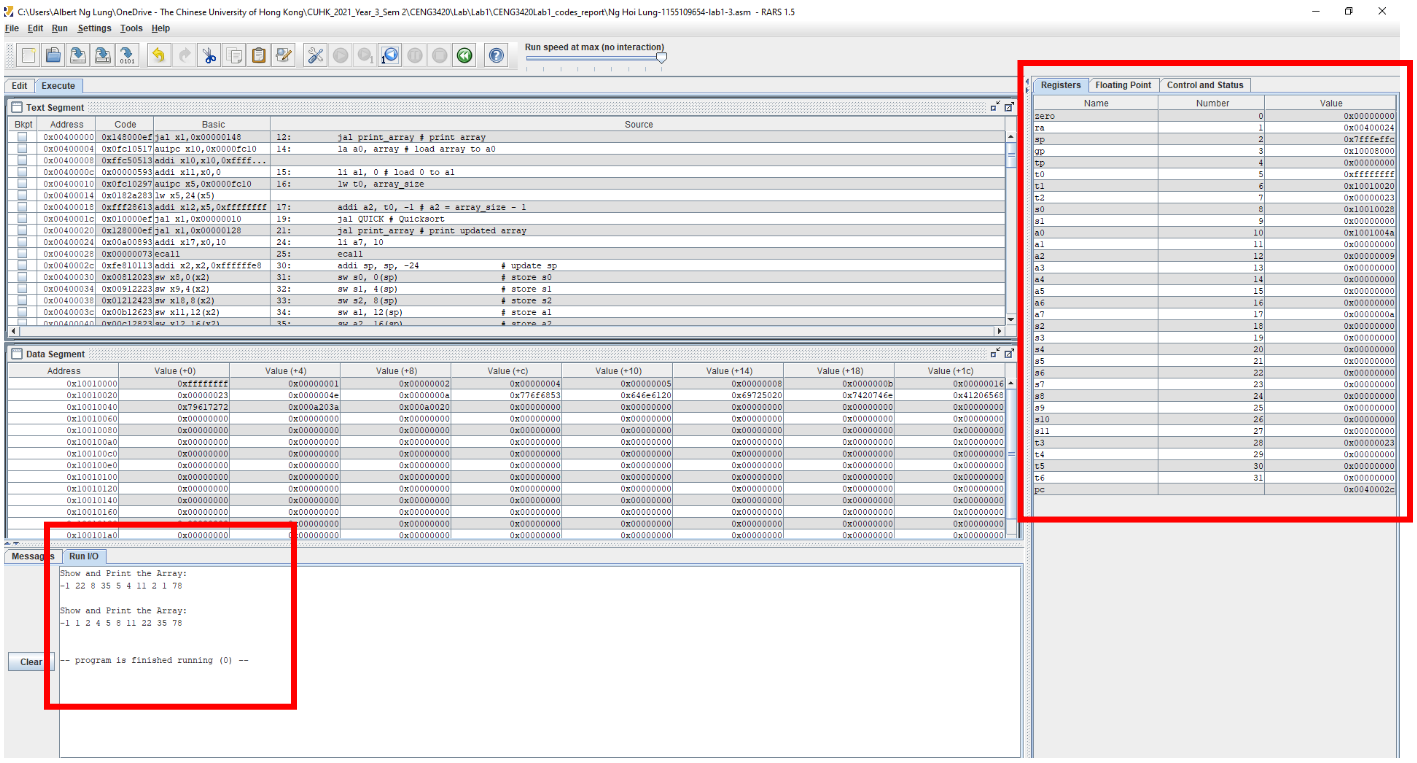
C Code:



```c
void quicksort(int *arr, int left, int right) {
    int l = left, r = right, p = left;
    while (l < r) {
        while (arr[l] <= arr[p] && l < right)
            l++;
        while (arr[r] >= arr[p] && r > left)
            r--;
        if (l >= r) {
            SWAP(arr[p], arr[r]);
            quicksort(arr, left, r - 1);
            quicksort(arr, r + 1, right);
            return;
            }
            SWAP(arr[l], arr[r]);
        }
}

void SWAP (int *x, int *y){
    int swapArray = array[x];
    array[x] = array[y];
    array[y] = swapArray;
}
```

Console results:

In ascending order



C:\Users\Albert Ng Lung\OneDrive - The Chinese University of Hong Kong\CUHK_2021_Year_3_Sem 2\CENG3420\Lab\Lab1\CENG3420Lab1_codes_report\Ng Hoi Lung-1155109654-lab1-3.asm - RARS 1.5

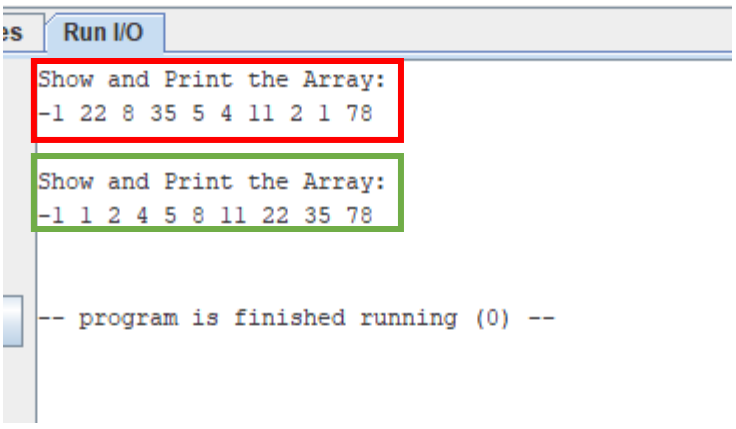
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Data Segment
Registers Floating Point Control and Status
Messages Run I/O
Clear

Bkpt	Address	Code	Basic	Source
	0x00400000	0x148000ef	jal x1,0x00000148	12: jal print_array # print array
	0x00400004	0x0fc10517	auipc x10,0x0000fc10	14: la a0, array # load array to a0
	0x00400008	0xffc50513	addi x10,x10,0xffff...	
	0x0040000c	0x00000593	addi x11,x0,0	15: li a1, 0 # load 0 to a1
	0x00400010	0x0fc10297	auipc x5,0x0000fc10	16: lw t0, array_size
	0x00400014	0x0182a283	lw x5,24(x5)	
	0x00400018	0xfff28613	addi x12,x5,0xffffffff	17: addi a2, t0, -1 # a2 = array_size - 1
	0x0040001c	0x010000ef	jal x1,0x00000010	19: jal QUICK # Quicksort
	0x00400020	0x128000ef	jal x1,0x00000128	21: jal print_array # print updated array
	0x00400024	0x00a00893	addi x17,x0,10	24: li a7, 10
	0x00400028	0x00000073	ecall	25: ecall
	0x0040002c	0xfe810113	addi x2,x2,0xffffffe8	30: addi sp, sp, -24 # update sp
	0x00400030	0x00812023	sw x8,0(x2)	31: sw s0, 0(sp) # store s0
	0x00400034	0x00912223	sw x9,4(x2)	32: sw s1, 4(sp) # store s1
	0x00400038	0x01212423	sw x18,8(x2)	33: sw s2, 8(sp) # store s2
	0x0040003c	0x00b12623	sw x11,12(x2)	34: sw a1, 12(sp) # store a1
	0x00400040	0x00c12823	sw x12,16(x2)	35: sw a2, 16(sp) # store a2

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0xffffffff	0x00000001	0x00000002	0x00000004	0x00000005	0x00000008	0x0000000b	0x00000016
0x10010020	0x00000023	0x0000004e	0x0000000a	0x776f6853	0x646e6120	0x69725020	0x7420746e	0x41206568
0x10010040	0x79617272	0x000a203a	0x000a0020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
[illegible]	[illegible]	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Name	Number	Value
zero	0	0x00000000
ra	1	0x00400024
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0xffffffff
t1	6	0x10010020
t2	7	0x00000023
s0	8	0x10010028
s1	9	0x00000000
a0	10	0x1001004a
a1	11	0x00000000
a2	12	0x00000009
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x0000000a
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000023
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040002c

```
Show and Print the Array:
-1 22 8 35 5 4 11 2 1 78

Show and Print the Array:
-1 1 2 4 5 8 11 22 35 78


-- program is finished running (0) --
```



```
Show and Print the Array:
-1 22 8 35 5 4 11 2 1 78

Show and Print the Array:
-1 1 2 4 5 8 11 22 35 78



-- program is finished running (0) --
```

Reference: TextBook -Computer Organization and Design_ The Hardware Software Interface [RISC-V Edition]