

CENG 3420 Lab1 Report

Name: [REDACTED] SID: [REDACTED]

Lab1.1

Step by step algorithm:

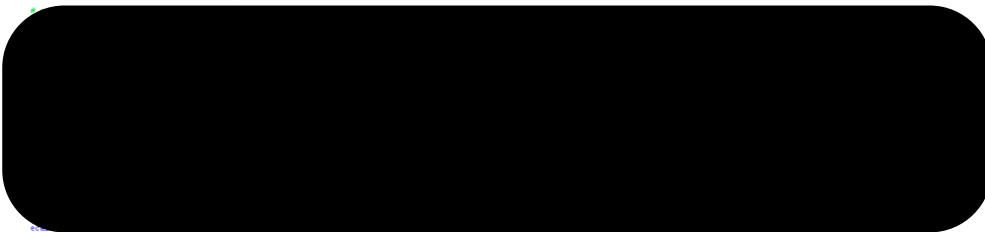
First of all, I need to define two variable one is var1 and the other one is var2 which is stored with 15 and 19 respectively. After that, the program will print the address of them which is using la a0, var1 and var2 to print the address with address 268501020, and 268501024. Then, I use addi to increase 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:



Console results:

The debugger window displays the following assembly code:

Inst	Address	Code	Basic	Source
	0x00400000	0x00400893	addi x17,x0,4	20: li a7, 4 # system call code for PrintString
	0x00400004	0xf0c10517	auipc x10,0x0000f0	21: la a9, avar # address of string to print
	0x00400008	0xf0c50513	addi x10,x10,0xffff...	
	0x0040000c	0x00000073	ecall	# Use the system call
	0x00400010	0x00400893	addi x17,x0,4	25: li a7, 4 # print "var1 address:"
	0x00400014	0xf0c10517	auipc x10,0x0000f0	26: la a9, var1_text_addr
	0x00400018	0x01050513	addi x10,x0,16	
	0x0040001c	0x00000073	ecall	
	0x00400020	0x0100893	addi x17,x0,1	29: li a7, 1 # print address of var1
	0x00400024	0xf0c10517	auipc x10,0x0000f0	30: la a9, var1
	0x00400028	0xf0c50513	addi x10,x0,0xffff...	
	0x0040002c	0x00000073	ecall	
	0x00400030	0x00400893	addi x17,x0,4	33: li a7, 4 # print "\n"
	0x00400034	0xf0c10517	auipc x10,0x0000f0	34: la a9, new_line
	0x00400038	0xf0c50513	addi x10,x0,29	
	0x0040003c	0x00000073	ecall	
	0x00400040	0x00400893	addi x17,x0,4	37: li a7, 4 # print "var2 address:"

The registers window shows the following values:

Registers	Name	Number	Value
	zero	0	0x00000000
	ra	1	0x00000000
	sp	2	0x7ffffefc
	gp	3	0x10000000
	tp	4	0x00000000
	t0	5	0x1001001c
	t1	6	0x00000000
	t2	7	0x00000000
	a0	8	0x00000000
	a1	9	0x00000000
	a2	10	0x00000000
	a3	11	0x0000004c
	a4	12	0x00000000
	a5	13	0x00000000
	a6	14	0x00000000
	a7	15	0x00000000
	a8	16	0x00000000
	a9	17	0x00000004
	a10	18	0x00000000
	a11	19	0x00000000
	a12	20	0x00000000
	a13	21	0x00000000
	a14	22	0x00000000
	a15	23	0x00000000
	a16	24	0x00000000
	a17	25	0x00000000
	a18	26	0x00000000
	a19	27	0x00000000
	a20	28	0x00000000
	a21	29	0x00000000
	a22	30	0x00000000
	a23	31	0x00000000
	pc		0x00400194

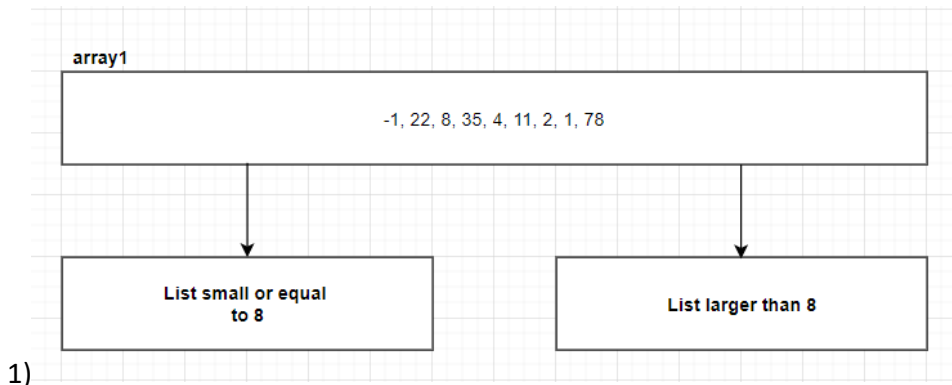
The console output shows the following results:

```
NS Hoi Lung 115510954
var1 address: 268501020
var2 address: 268501024
var1: 16
var2: 76
Swap var1: 76
Swap var2: 16
-- program is finished running (0) --
```

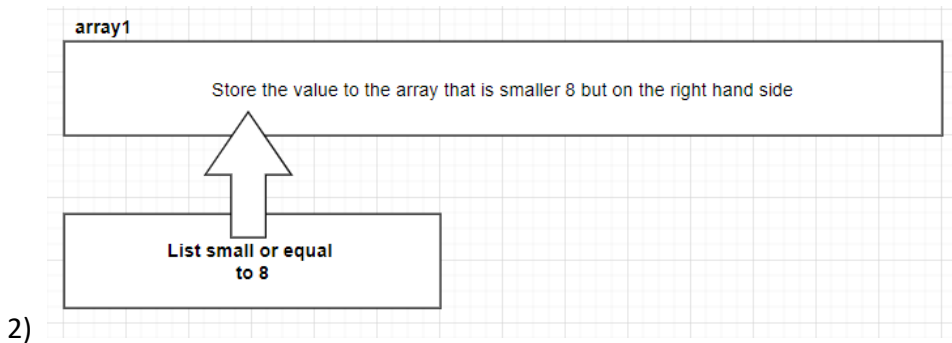
Lab1.2

Step by step algorithm:

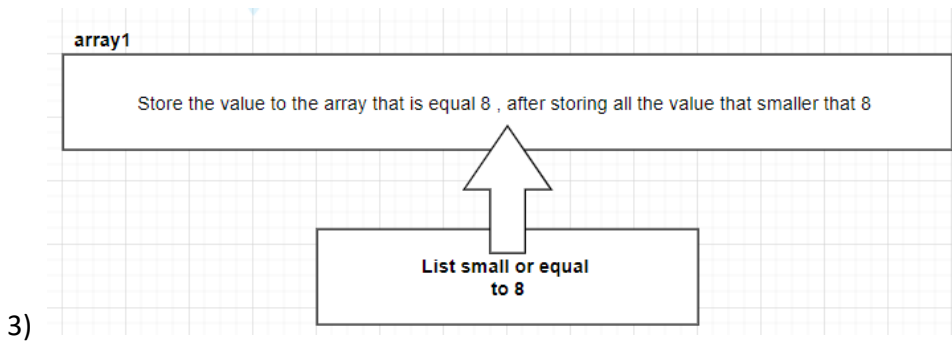
In this lab, 8 is the middle value the left-hand side will have -1, 5, 4, 2, 1 and the right-hand side will have 22, 35, 11, 78 which requirement the lab requirement. The method I am using will be shown in the graph below:



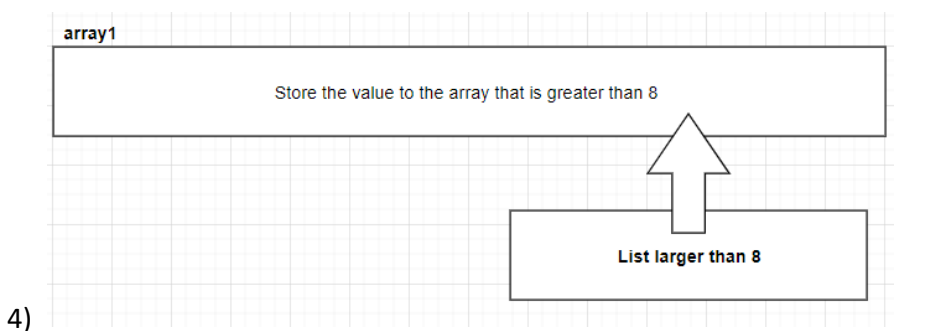
I am going to separate array1 which two list which is the list that small and equal to third element 8 and the list larger than 8.



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



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



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:

The screenshot shows a debugger window with the following components:

- Text Segment:** A table of assembly instructions with columns for BitPt, Address, Code, Basic, and Source. The code includes instructions like `la s0, array1`, `lw s1, len`, `li t4, 4`, `mul t4, s1, t4`, `add t0, s0, t4`, `add t1, zero, zero`, `slti t4, s4, 1`, `add t3, s0, t4`, `add t3, zero, zero`, `lw s0, (t0)`, `jal check`, `addi s0, s0, 4`, `addi s1, s1, -1`, `bgt s1, zero, loop`, and `la s0, array1`.
- Registers:** A table with columns for Name, Number, and Value. Registers shown include `zero` through `pc`.
- Data Segment:** A table showing memory addresses and their values in various bases (hex, decimal, octal, binary).
- Message/Run IO:** A window showing the program's output: `-1`, `22`, `8`, `35`, `5`, `4`, `11`, `2`, `1`, `78`, followed by `-- program is finished running (0) --`.

The screenshot shows a console window with the following output:

```
-1
22
8
35
5
4
11
2
1
78
-- program is finished running (0) --
```

Lab1.3

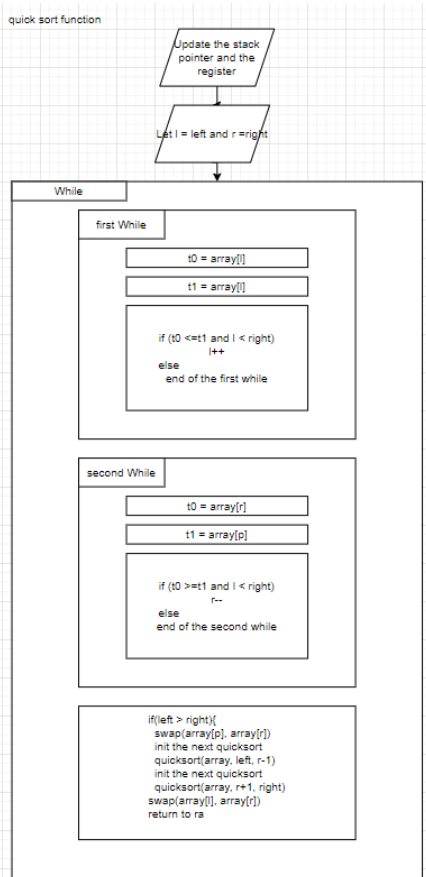
Step by step algorithm:

Assembly implementation function:

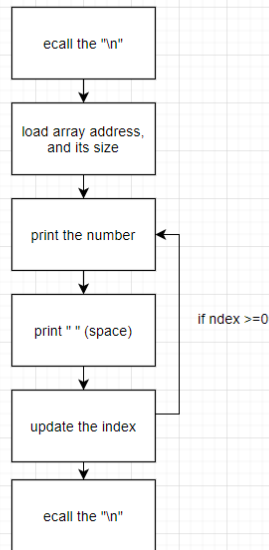
(quick sort function)

(print function)

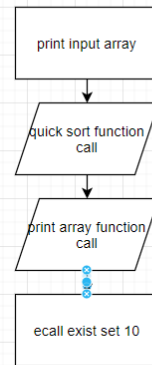
(_start)



print array function



_start



Assembly key code:

```

149 #
150 #
151 #
152 #
153 #
154 #
155 #
156 #
157 #
158 #
159 #
160 #
161 #
162 #
163 #
164 #
165 #
166 #
167 #
168 #
169 #
170 #
171 #
172 #
173 #
174 #
175 #
27 #
28 #
29 #
30 #
31 #
32 #
33 #
34 #
35 #
36 #
37 #
38 #
39 #
40 #
41 #
42 #
43 #
44 #
45 #
46 #
47 #
48 #
49 #
50 #
51 #
52 #
53 #
54 #
55 #
  
```



```
es Run I/O
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]