CENG 3420 Lab1 Report
Name: Wong Wai Chun SID:1155173231

Lab1.1

I define var1 and var2 with 15 and 19 respectively, and I print the MEMORY addresses of var1 and var2 using ecall by loading them in a0, value 1 for int, value 0 for char. Using addi to increase var1 by 1 and slli to multiply var2 by 4.i use sb for exchanging var1 and var2.

```
.data
var1: .byte 15
var2: .byte 19
msg: .asciz "\n"
```

print the MEMORY addresses:

```
li a7,1
la a0, var1                    #address var1
ecall

li a7,4
la a0, msg
ecall

li a7,1
la a0, var2                    #address var2
ecall
```

addi and slli(and print):

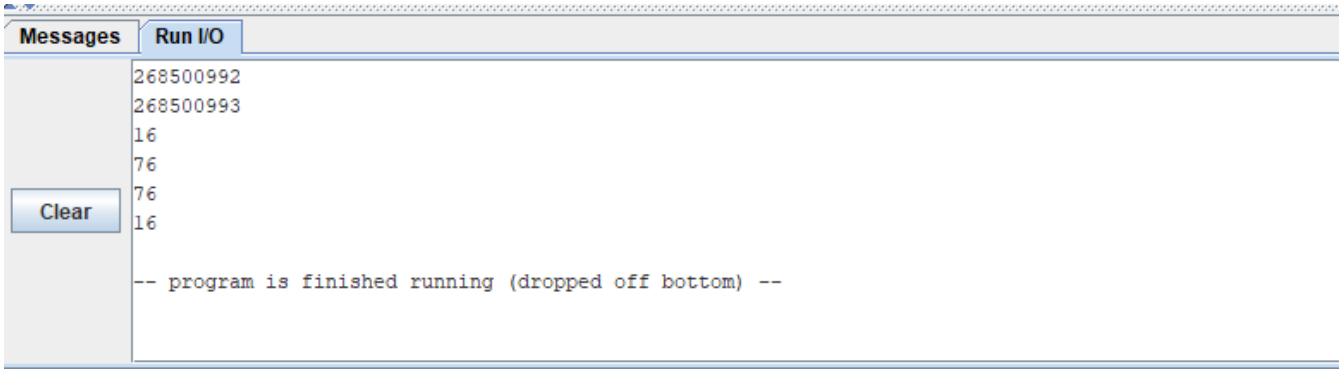
```
la a1,var1
la a2,var2
lb t0,var1
lb t1,var2

addi t0,t0,1
slli t1,t1,2

sb t0, (a1)
sb t1, (a2)
```

```
li a7,1
lb a0, var1                    #var1
ecall

li a7,4
la a0, msg
ecall

li a7,1
lb a0, var2                    #var2
ecall
```

Swap:

```
sb t0, (a2)
sb t1, (a1)

li a7,1
lb a0, var1
ecall

li a7,4
la a0, msg
ecall

li a7,1
lb a0, var2
ecall

li a7,4
la a0, msg
ecall
```

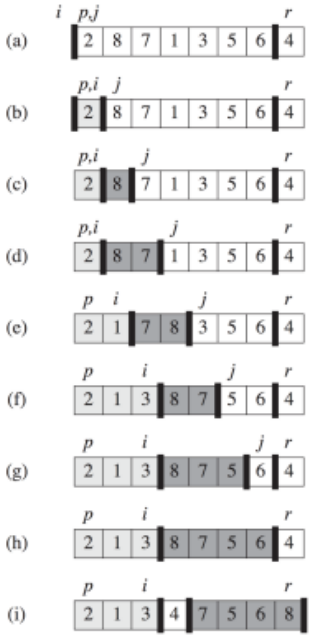
Result:



Lab1.2

Algo:

```
1:  function PARTITION(A, lo, hi)
2:      pivot ← A[hi]
3:      i ← lo-1;
4:      for j = lo; j ≤ hi-1; j ← j+1 do
5:          if A[j] ≤ pivot then
6:              i ← i+1;
7:              swap A[i] with A[j];
8:          end if
9:      end for
10:     swap A[i+1] with A[hi];
11:     return i+1;
12: end function
```



To fit in our case, we choose 8 to be the r, so I have to swap 8 and 78 first. Then it follow the algo.

```
.text
_start:

    la a0, array1
    li a1, 0            #start  lo/j
    li a2, 9            #end    hi


    #swap 8 and 78 > set 8 to be pivot
    lw t0, 8(a0)
    lw t1, 36(a0)
    sw t0, 36(a0)       #t0 = 8
    sw t1, 8(a0)

    addi t1,a1,-1           #i
    li t2,8             #hi-1

    j loop
```

Before:

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	-1	22	8	35	5	4	11	2
0x10010020	1	78	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0

After:

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	-1	5	4	2	1	8	11	35
0x10010020	22	78	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0

Code(exclude start):

```
loop:
    bgt a1, t2, end      #end conitdtion


    slli t6,a1,2
    add  t6,a0,t6
    lw t3, 0(t6)           #get -1    t6= adress of j

    jal ra, check
    addi a1,a1,1          #j++
    j loop

check:
    ble t3,t0,lessthaneq
    jalr zero, 0(ra)

lessthaneq:
    addi t1,t1,1         #i++
    slli t4,t1,2
    add t4, t4, a0

    lw t5, 0(t4)         #a[i]
    sw t3, 0(t4)         #swap value
    sw t5, 0(t6)

    jalr zero, 0(ra)

end:
   addi t1,t1,1         #i++
   slli t4,t1,2
   add t4, t4, a0
   lw t5, 0(t4)         #a[i+1]
   sw t0, 0(t4)
   sw t5, 36(a0)
```

Lab1.3

Algo:

```
1: function QUICKSORT(A, lo, hi)
2:     if lo < hi then
3:         p ← partition(A, lo, hi);
4:         quicksort(A, lo, p - 1);
5:         quicksort(A, p + 1, hi);
6:     end if
7: end function
```

This is the extension of Lab1.2, it choose 1 point as pivot and separate two part each time, from i=0 to i=p-1 and i=p+1 to i=hi. I changed to use stack pointer to store ra and more element, because i don't have enough temporary registers.

```
.text
_start:

        #input args
        la a0, array1
        li a1, 0        #lo
        li a2, 9        #hi
        jal ra, quicksort
        j realend

quicksort:
        addi sp, sp , -4
        sw ra, 0(sp)

        jal ra, checkIsLessThan

        lw ra, 0(sp)    #target jump location
        addi sp, sp, 4
        jalr zero, 0(ra)

checkIsLessThan:
        bge a1, a2, Back        # hi greater or equal to lo

        #sore original ra
        addi sp, sp, -4
        sw ra, 0(sp)
```

```
Back:
        jalr zero, 0(ra)

partition:
        addi sp, sp, -4
        sw ra, 0(sp)

        addi sp, sp, -16
        add a3, a1, zero        #lo / j
        add a4, a2, zero        #hi

        # privot information
        slli t0, a4, 2
        add t0, t0, a0
        sw t0, 0(sp)    #address of privot
        lw t0, 0(t0)
        sw t0, 4(sp)    #value of privot

        addi t0, a3, -1 #t0 = i
        sw t0, 8(sp)    #value of i

        #jump to loop
        j loop
```

These two part is newly added to fit into the quicksort,and the other part is remain the same.

Result:

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	-1	1	2	4	5	8	11	22
0x10010020	35	78	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0