

Phạm Duy Hưng – 20225850

Assignment 1:

```
.data
A: .word 29,11,-12,22,4
.text
main: la $a0,A
      li $a1,5
      j mspfx
      nop
continue:
lock:  j lock
      nop
end_of_main:
mspfx: addi $v0,$zero,0 #initialize length in $v0 to 0
      addi $v1,$zero,0 #initialize max sum in $v1 to 0
      addi $t0,$zero,0 #initialize index i in $t0 to 0
      addi $t1,$zero,0 #initialize running sum in $t1 to 0
loop:  add $t2,$t0,$t0 #put 2i in $t2
      add $t2,$t2,$t2 #put 4i in $t2
      add $t3,$t2,$a0 #put 4i+A (address of A[i]) in $t3
      lw $t4,0($t3) #load A[i] from mem(t3) into $t4
      add $t1,$t1,$t4 #add A[i] to running sum in $t1
      slt $t5,$v1,$t1 #set $t5 to 1 if max sum < new sum
      bne $t5,$zero,mdfy #if max sum is less, modify results
      j test #done?
mdfy:  addi $v0,$t0,1 #new max-sum prefix has length i+1
      addi $v1,$t1,0 #new max sum is the running sum
test:  addi $t0,$t0,1 #advance the index i
      slt $t5,$t0,$a1 #set $t5 to 1 if i<n
      bne $t5,$zero,loop #repeat if i<n
done:  j continue
mspfx_end:
```

The screenshot displays a MIPS assembler simulator interface with the following components:

- Text Segment:** A table showing assembly instructions with columns for Bkpt, Address, Code, Basic, and Source.

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c011001	lui \$t1,0x00001001	4: main: la \$a0,A
	0x00400004	0x34240000	ori \$t4,\$t1,0x00000000	
	0x00400008	0x24050005	addiu \$t5,\$t0,0x00000005	5: li \$a1,5
	0x0040000c	0x09100007	j 0x0040001c	6: j mspfx
	0x00400010	0x00000000	nop	7: nop
	0x00400014	0x09100005	j 0x00400014	9: lock: j lock
	0x00400018	0x00000000	nop	10: nop
	0x0040001c	0x20020000	addi \$t2,\$t0,0x00000000	12: mspfx: addi \$v0,\$zero,0 #initialize...
	0x00400020	0x20030000	addi \$t3,\$t0,0x00000000	13: addi \$v1,\$zero,0 #initialize...
- Data Segment:** A table showing memory addresses and their corresponding values in various formats.

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x0000001d	0x0000000b	0xffffffff4	0x0000001e	0x00000004	0x00000000	0x00000000	0x00000000
0x10010004	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010008	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001000c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010010	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010014	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010018	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001001c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
- Labels:** A table mapping labels to memory addresses.

Label	Address
main	0x00400000
continue	0x00400014
lock	0x00400014
end_of_main	0x0040001c
mspfx	0x0040001c
loop	0x0040002c
mdfy	0x0040004c
- Registers:** A table showing the state of MIPS registers.

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10010000
\$v0	2	0x00000005
\$v1	3	0x0000003e
\$a0	4	0x10010000
\$a1	5	0x00000005
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000005
\$t1	9	0x0000003e
\$t2	10	0x00000010
\$t3	11	0x10010010
\$t4	12	0x00000004
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$s8	24	0x00000000
\$s9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7fffffc
\$fp	30	0x00000000
\$ra	31	0x00000000
- Mars Messages:** A section showing system messages, including "Reset: reset completed."

- Dãy số đã cho có tổng chuỗi con lớn nhất là 54, chuyển sang hệ hex là 36 => Kết quả đúng.

Assignment 2:

```
.data
A: .word 29,11,-12,22,4,5,-8,-9,10,-15
Aend: .word

.text
main:      la $a0,A #$a0 = Address(A[0])
           la $a1,Aend
           addi $a1,$a1,-4 #$a1 = Address(A[n-1])
           j sort #sort
after_sort: li $v0, 10 #exit
           syscall
end_main:
sort:      beq $a0,$a1,done #single element list is sorted
           j max #call the max procedure
after_max: lw $t0,0($a1) #load last element into $t0
           sw $t0,0($v0) #copy last element to max location
           sw $v1,0($a1) #copy max value to last element
           addi $a1,$a1,-4 #decrement pointer to last element
           j sort #repeat sort for smaller list
done:      j after_sort
max:
           addi $v0,$a0,0 #init max pointer to first element
           lw $v1,0($v0) #init max value to first value
           addi $t0,$a0,0 #init next pointer to first
loop:
           beq $t0,$a1,ret #if next=last, return
           addi $t0,$t0,4 #advance to next element
           lw $t1,0($t0) #load next element into $t1
           slt $t2,$t1,$v1 #(next)<(max) ?
           bne $t2,$zero,loop #if (next)<(max), repeat
           addi $v0,$t0,0 #next element is new max element
           addi $v1,$t1,0 #next value is new max value
           j loop #change completed; now repeat
ret:
           j after_max
```


Assignment 3:

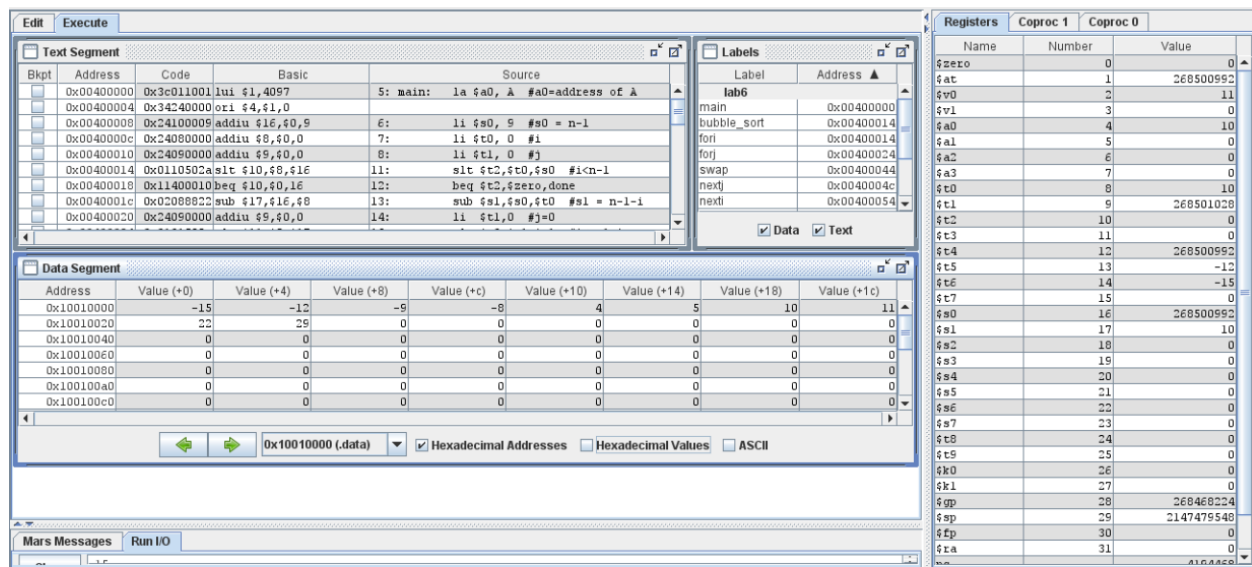
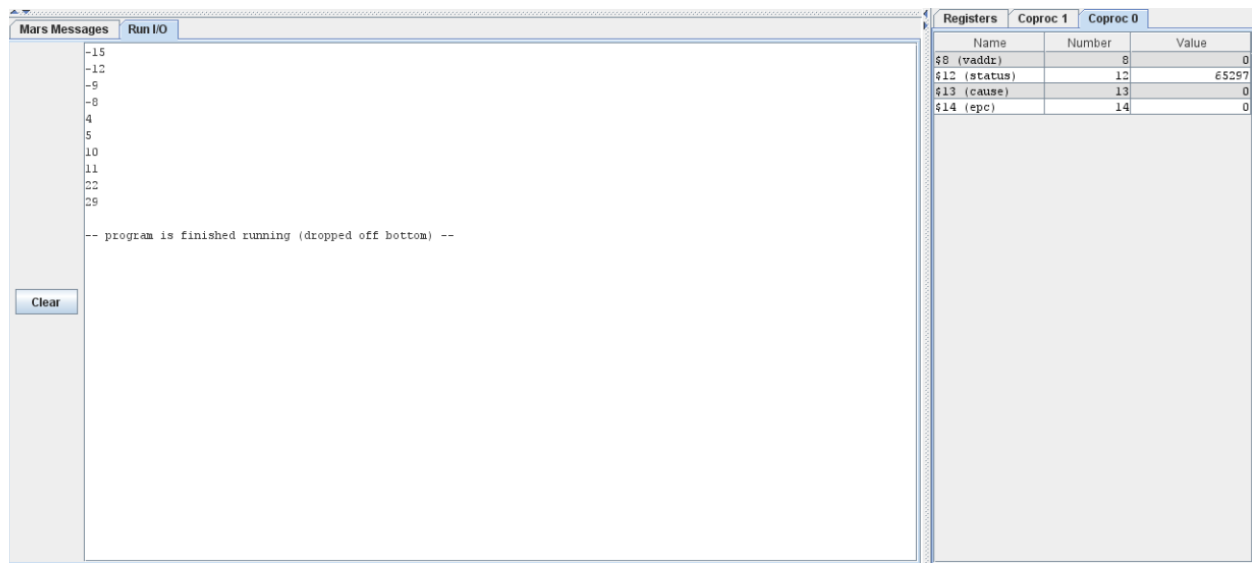
```
.data
A: .word 29,11,-12,22,4,5,-8,-9,10,-15

.text
main:  la $a0, A  #a0=address of A
      li $s0, 9   #s0 = n-1
      li $t0, 0   #i
      li $t1, 0   #j
bubble_sort:
fori:
      slt $t2,$t0,$s0  #i<n-1
      beq $t2,$zero,done
      sub $s1,$s0,$t0  #s1 = n-1-i
      li $t1,0        #j=0
forj:
      slt $t3,$t1,$s1  #j<n-1-i ?
      beq $t3,$zero,nexti
      sll $t4,$t1,2    #t4 = 4j
      add $t4,$t4,$a0  #t4 = address of A[j]
      lw $t5, 0($t4)   #t5=a[j]
      lw $t6, 4($t4)   #t6=a[j+1]
      slt $t7,$t5,$t6  #A[j] < A[j+1]?
      bne $t7,$zero,nextj
swap:
      sw $t5, 4($t4)
      sw $t6, 0($t4)
nextj:
      addi $t1,$t1,1   #j++
      j forj
nexti:
      addi $t0,$t0,1   #i++
      j fori
done:
      nop
print_array:
      la $s0,A
      li $t0,0         #i
      li $s1,10        #n
loop:
      slt $t2,$t0,$s1
      beq $t2,$zero,stop
      sll $t1,$t0,2
      add $t1,$t1,$s0  #t1=&A[i]
```

```

lw $a0,0($t1)  #a0=A[i]
li $v0,1      #print integer
syscall
li $a0, '\n'
li $v0,11
syscall
addi $t0,$t0,1 #i++
j loop
stop:
nop

```



⇒ Dãy 29,11,-12,22,4,5,-8,-9,10,-15 đã được sắp xếp theo thứ tự tăng dần.

Assignment 4:

```
.data
A: .word 29,11,-12,22,4,5,-8,-9,10,-15
.text
    la $s0,A    #s0=address of A
    li $s1,10   #s1 = n
    sll $s1,$s1,2
    addi $s2,$s0,4
    li $s3,4
insertion_sort:
    slt $v1,$s3,$s1    #if i>9 go to end
    beq $v1,$0,end
    add $t0,$0,$s2      #else continue
    add $t1,$0,$s3
loop:
    lw $t2,-4($t0)
    lw $t3,0($t0)
    slt $v1,$t2,$t3    #if a[j]<a[j+1] go to end_of_loop
    bne $v1,$0,end_of_loop
swap:                    #else go to swap
    sw $t2,0($t0)
    sw $t3,-4($t0)
continue:
    subi $t0,$t0,4      #j=j-1
    subi $t1,$t1,4
    slti $v1,$t1,4      #if j>0 go to loop
    beq $v1,$0,loop
end_of_loop:
    addi $s2,$s2,4      #i=i+1
    addi $s3,$s3,4
j insertion_sort        #go to insertion_sort
end:
    nop
print_array:
    la $s0,A
    li $t0,0    #i
    li $s1,10   #n
loopp:
    slt $t2,$t0,$s1
    beq $t2,$zero,stop
    sll $t1,$t0,2
    add $t1,$t1,$s0 #t1=&A[i]
    lw $a0,0($t1)  #a0=A[i]
    li $v0,1       #print integer
```

```
syscall
li $a0, '\n'
li $v0, 11
syscall
addi $t0, $t0, 1 #i++
j loopp
stop:
nop
```

[illegible]

Mars Messages

Run IO

```

Reset: reset completed.

-15
-12
-9
-8
4
9
10
11
22
29

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

```

Clear

Registers

Coproc 1

Coproc 0

Name	Number	Value
sp0r0	0	0
ia5	1	269500992
z00	2	31
rv1	3	0
ea0	4	10
ta1	5	0
ra2	6	0
ta3	7	0
ea3	8	10
tl1	9	269501023
ta2	10	0
tl3	11	-15
ra4	12	0
tl5	13	0
ra5	14	0
tl7	15	0
ea0	16	269500992
ra1	17	10
ra2	18	269501032
ta3	19	40
ra4	20	0
ra5	21	0
ra6	22	0
ra7	23	0
tl8	24	0
tl9	25	0
ra0	26	0
tl1	27	0
gdp	28	269468224
lpy	29	2147479641
zfp	30	0
raa	31	0
gpc		4194476
tl		
lo		0

⇒ Dãy 29, 11, -12, 22, 4, 5, -8, -9, 10, -15 đã được sắp xếp theo thứ tự tăng dần.