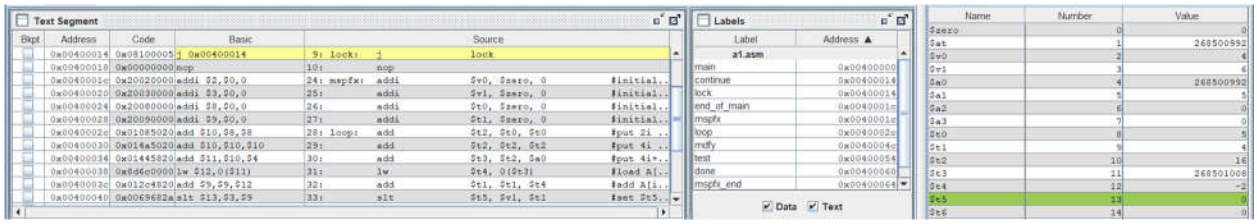
Lab Report 06

Assignment 1

Code:

Comments:

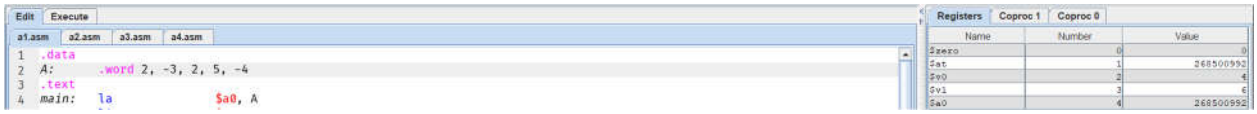
- Input 1: (-2, 6, -1, 3, -2)
  ⇨ Max sum = -2 + 6 – 1 + 3 = 6, w/ length = 4



$v0 = 4

$v1 = 6

- Input 2: (2, -3, 2, 5, -4)
  ⇨ Max sum = 2 + -3 + 2 + 5 = 5, w/ length = 4



$v0 = 4

$v1 = 6

Assignment 2

Code:

```
.data
A:          .word 7, -2, 5, 1, 5, 6, 7, 3, 6, 8, 8, 59, 5
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
```

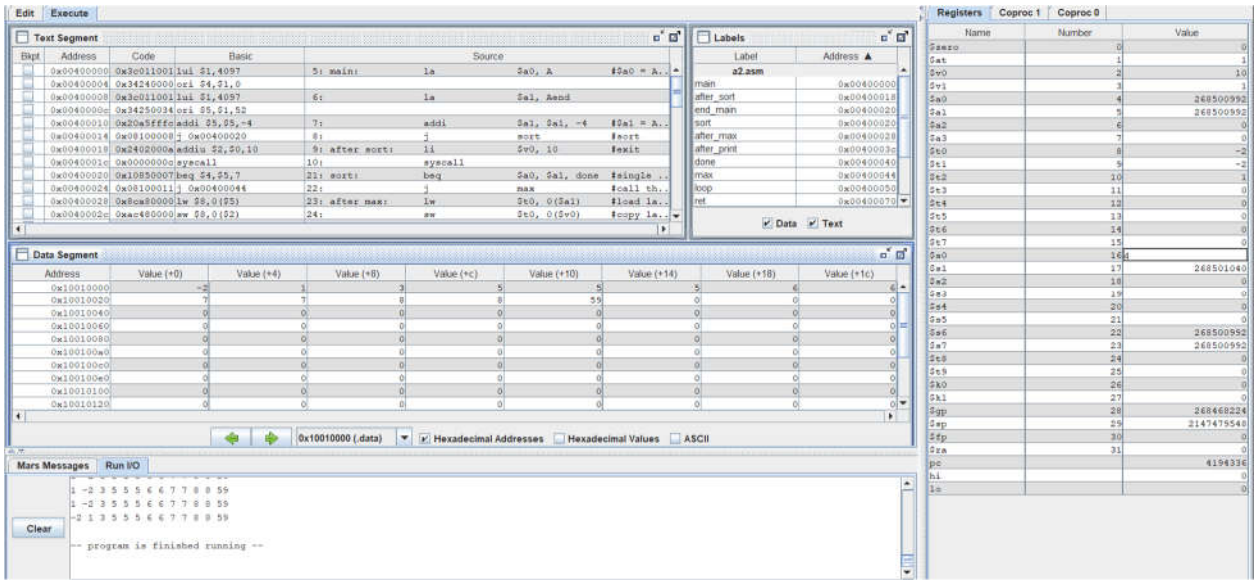
```
                syscall
end_main:
#-------------------------------------------------------------
#procedure sort (ascending selection sort using pointer)
#register usage in sort program
#$a0 pointer to the first element in unsorted part
#$a1 pointer to the last element in unsorted part
#$t0 temporary place for value of last element
#$v0 pointer to max element in unsorted part
#$v1 value of max element in unsorted part
#-------------------------------------------------------------
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, $a1, -4    #decrement pointer to last element
                j           print
after_print:    j           sort            #repeat sort for smaller list
done:           j           after_sort
#---------------------------------------------------------------------
#Procedure max
#function: fax the value and address of max element in the list
#$a0 pointer to first element
#$a1 pointer to last element
#---------------------------------------------------------------------
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
#------------------------------------------------------------------
#Procedure print
#------------------------------------------------------------------
print:          add         $s7, $v0, $zero         #Save
                add         $s6, $a0, $zero         #Save
                la          $s0, A                  #$s0 = Address(A[0])
                la          $s1, Aend
                addi        $s1, $s1, -4            #$s1 = Address(A[n-1])

loop2:          blt         $s1, $s0, endprint      #exit if Aned < A[i]

                li          $v0, 1                  #service 01: print integer
                lw          $a0, 0($s0)             #value of A[i]
                syscall
                li          $v0, 11
                li          $a0, ' '                #print space
                syscall
```

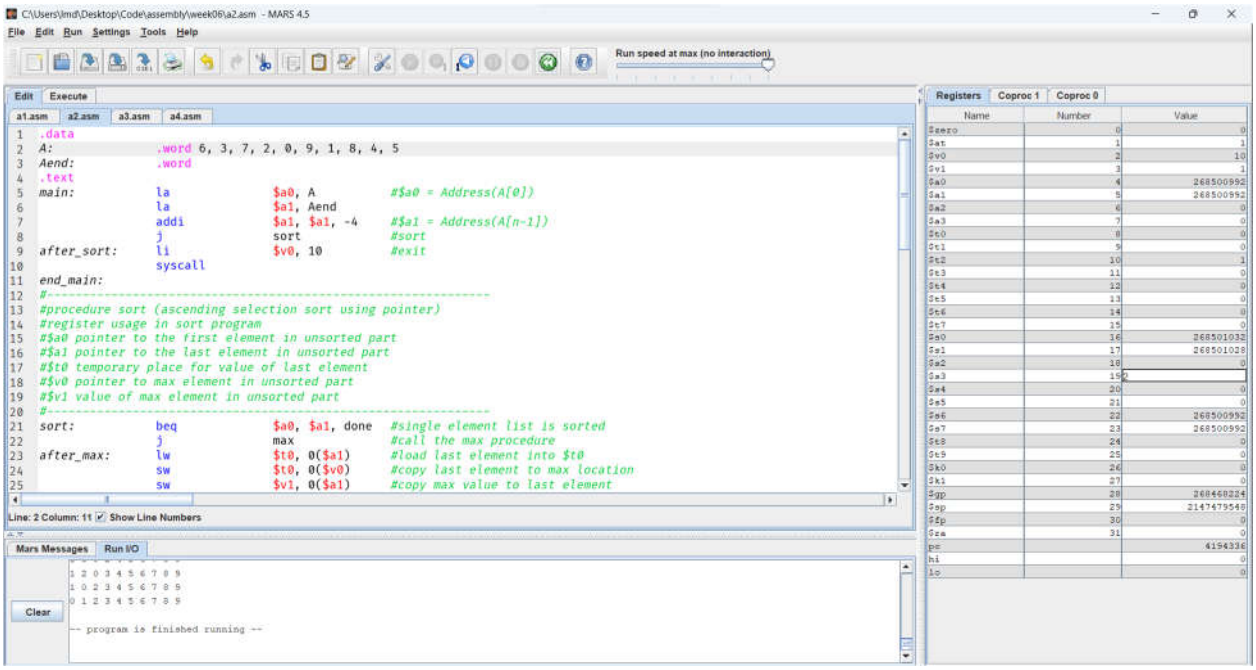
```
                addi        $s0, $s0, 4         #Move to A[i+1]
                j           loop2

endprint:       li          $v0, 11
                li          $a0, '\n'           #print enter
                syscall
                add         $v0, $s7, $zero     #Load
                add         $a0, $s6, $zero     #Load
                j           after_print
```

Comments:

- Input: 7, -2, 5, 1, 5, 6, 7, 3, 6, 8, 8, 59, 5



- Input: 6, 3, 7, 2, 0, 9, 1, 8, 4, 5



Assignment 3

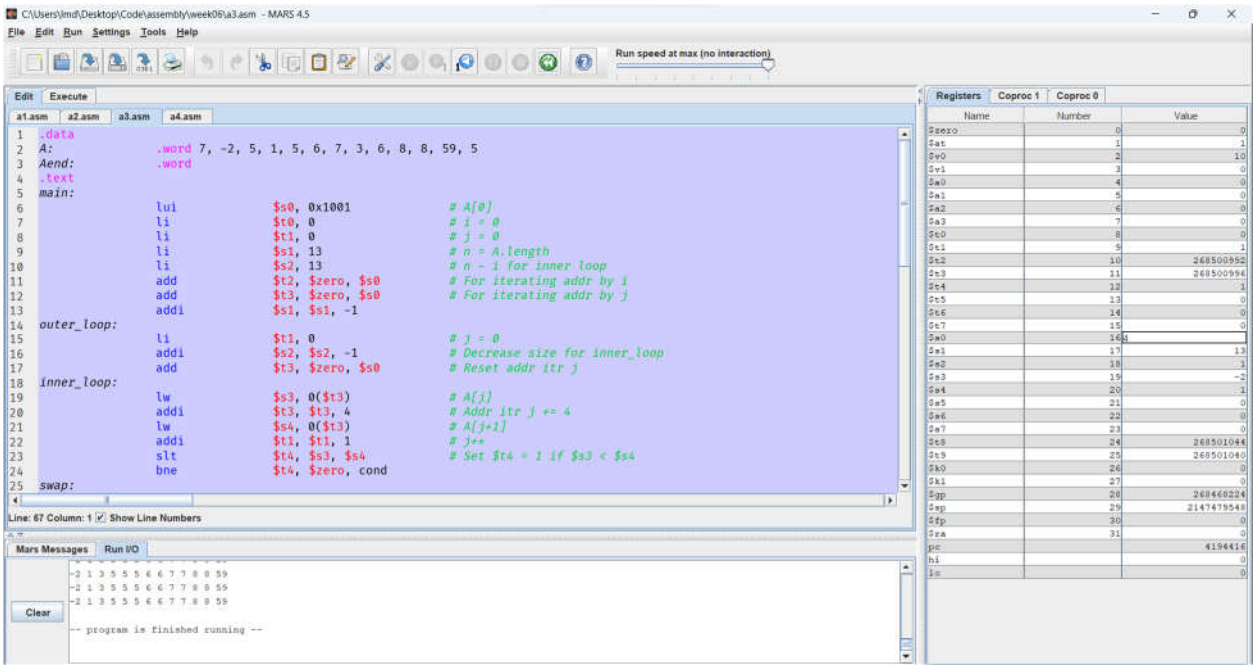
Code:

```
.data
A:          .word 7, -2, 5, 1, 5, 6, 7, 3, 6, 8, 8, 59, 5
Aend:       .word
.text
main:
            lui     $s0, 0x1001         # A[0]
            li      $t0, 0              # i = 0
            li      $t1, 0              # j = 0
            li      $s1, 13             # n = A.length
            li      $s2, 13             # n - i for inner loop
            add     $t2, $zero, $s0     # For iterating addr by i
            add     $t3, $zero, $s0     # For iterating addr by j
            addi    $s1, $s1, -1
outer_loop:
            li      $t1, 0              # j = 0
            addi    $s2, $s2, -1        # Decrease size for inner_loop
```

```
                add         $t3, $zero, $s0         # Reset addr itr j
inner_loop:
                lw          $s3, 0($t3)             # A[j]
                addi        $t3, $t3, 4             # Addr itr j += 4
                lw          $s4, 0($t3)             # A[j+1]
                addi        $t1, $t1, 1             # j++
                slt         $t4, $s3, $s4           # Set $t4 = 1 if $s3 < $s4
                bne         $t4, $zero, cond
swap:
                sw          $s3, 0($t3)
                sw          $s4, -4($t3)
                lw          $s4, 0($t3)
cond:
                bne         $t1, $s2, inner_loop    #j != n-i
                j           Print
EndPrt:
                addi        $t0, $t0, 1             #i++
                bne         $t0, $s1, outer_loop    #i != n
                li          $t0, 0
                addi        $s1, $s1, 1
exit:
                li          $v0, 10
                syscall


#---------------------------------------------------------------------
# Procedure Print
#---------------------------------------------------------------------
Print:          add         $s7, $v0, $zero         # Save
                add         $s6, $a0, $zero         # Save
```

```
                la      $t8, A              # $t8 = Address(A[0])
                la      $t9, Aend
                addi    $t9, $t9, -4        # $t9 = Address(A[n-1])

LoopPrt:        blt     $t9, $t8, EndLoopPrt # Exit if Aend < A[i]

                li      $v0, 1              # Service 01: print integer
                lw      $a0, 0($t8)         # Value of A[i]
                syscall
                li      $v0, 11
                li      $a0, ' '            # Print space
                syscall
                addi    $t8, $t8, 4         # Move to A[i+1]
                j       LoopPrt

EndLoopPrt:     li      $v0, 11
                li      $a0, '\n'           # Print enter
                syscall
                add     $v0, $s7, $zero     # Load
                add     $a0, $s6, $zero     # Load
                j       EndPrt
```

Result:



Assignment 4

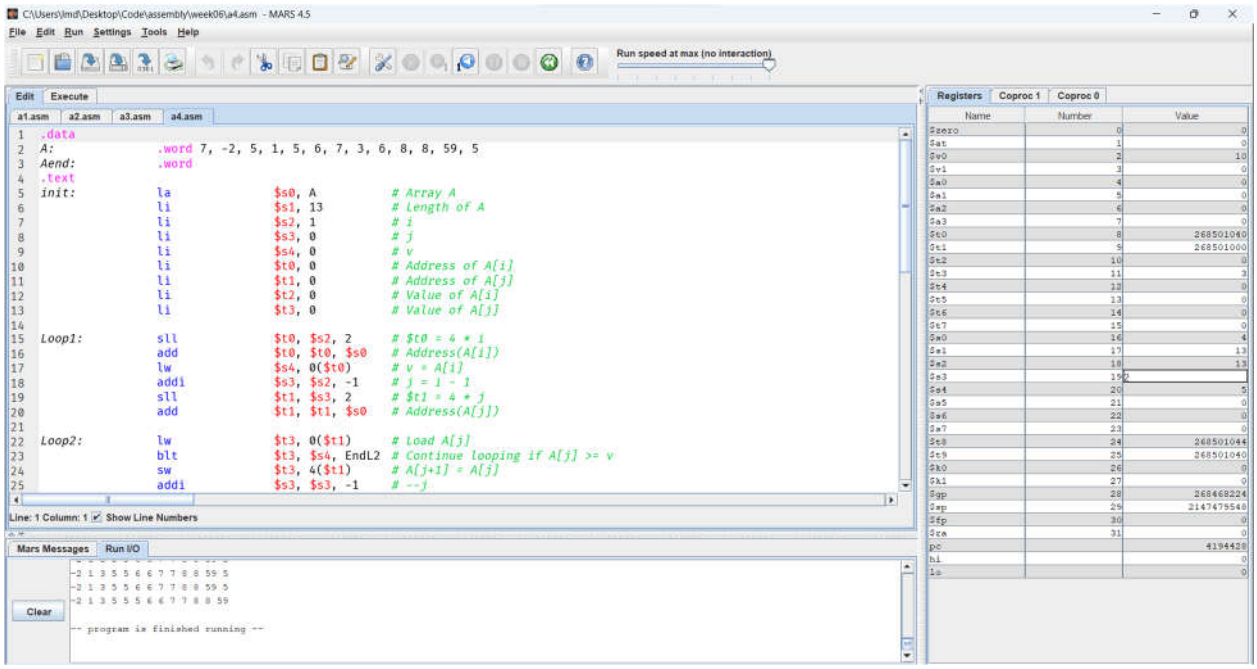
Code:

```
.data
A:          .word 7, -2, 5, 1, 5, 6, 7, 3, 6, 8, 8, 59, 5
Aend:       .word
.text
init:       la      $s0, A          # Array A
            li      $s1, 13         # Length of A
            li      $s2, 1          # i
            li      $s3, 0          # j
            li      $s4, 0          # v
            li      $t0, 0          # Address of A[i]
            li      $t1, 0          # Address of A[j]
            li      $t2, 0          # Value of A[i]
            li      $t3, 0          # Value of A[j]

Loop1:      sll     $t0, $s2, 2     # $t0 = 4 * i
            add     $t0, $t0, $s0   # Address(A[i])
```

```
                lw      $s4, 0($t0)     # v = A[i]
                addi    $s3, $s2, -1    # j = i - 1
                sll     $t1, $s3, 2     # $t1 = 4 * j
                add     $t1, $t1, $s0   # Address(A[j])

Loop2:          lw      $t3, 0($t1)     # Load A[j]
                blt     $t3, $s4, EndL2 # Continue looping if A[j] >= v
                sw      $t3, 4($t1)     # A[j+1] = A[j]
                addi    $s3, $s3, -1    # --j
                addi    $t1, $t1, -4    # Keep memory access consistent with j
                bge     $s3, $0, Loop2  # Loop if j >= 0

EndL2:          sw      $s4, 4($t1)     # A[j+1] = v
                addi    $s2, $s2, 1     # ++i
                j       Print

EndPrt:         blt     $s2, $s1, Loop1 # Loop while i < A.length

Exit:           li      $v0, 10 # Load exit operation
                syscall

#---------------------------------------------------------------------
# Procedure Print
#---------------------------------------------------------------------
Print:          add     $s7, $v0, $zero         # Save
                add     $s6, $a0, $zero         # Save
                la      $t8, A                  # $t8 = Address(A[0])
                la      $t9, Aend
                addi    $t9, $t9, -4            # $t9 = Address(A[n-1])
```

```
LoopPrt:      blt     $t9, $t8, EndLoopPrt    # Exit if Aend < A[i]

              li      $v0, 1                  # Service 01: print integer
              lw      $a0, 0($t8)             # Value of A[i]
              syscall
              li      $v0, 11
              li      $a0, ' '                # Print space
              syscall
              addi    $t8, $t8, 4             # Move to A[i+1]
              j       LoopPrt

EndLoopPrt:   li      $v0, 11
              li      $a0, '\n'               # Print enter
              syscall
              add     $v0, $s7, $zero         # Load
              add     $a0, $s6, $zero         # Load
              j       EndPrt
```

Result:


C:\Users\lmd\Desktop\Code\assembly\week06\a4.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
a1.asm a2.asm a3.asm a4.asm
1 .data
2 A: .word 7, -2, 5, 1, 5, 6, 7, 3, 6, 8, 8, 59, 5
3 Aend: .word
4 .text
5 init: la $s0, A # Array A
6 li $s1, 13 # Length of A
7 li $s2, 1 # i
8 li $s3, 0 # j
9 li $s4, 0 # v
10 li $t0, 0 # Address of A[i]
11 li $t1, 0 # Address of A[j]
12 li $t2, 0 # Value of A[i]
13 li $t3, 0 # Value of A[j]
14
15 Loop1: sll $t0, $s2, 2 # $t0 = 4 * i
16 add $t0, $t0, $s0 # Address(A[i])
17 lw $s4, 0($t0) # v = A[i]
18 addi $s3, $s2, -1 # j = i - 1
19 sll $t1, $s3, 2 # $t1 = 4 * j
20 add $t1, $t1, $s0 # Address(A[j])
21
22 Loop2: lw $t3, 0($t1) # load A[j]
23 blt $t3, $s4, EndL2 # Continue looping if A[j] >= v
24 sw $t3, 4($t1) # A[j+1] = A[j]
25 addi $s3, $s3, -1 # --j
Line: 1 Column: 1 Show Line Numbers
Mars Messages Run I/O
-2 1 3 5 5 6 6 7 7 8 8 59 5
-2 1 3 5 5 6 6 7 7 8 8 59 5
-2 1 3 5 5 5 6 6 7 7 8 8 59
Clear
-- program is finished running --
Registers Coproc 1 Coproc 0
Name Number Value
$zero 0 0
$at 1 0
$v0 2 10
$v1 3 0
$a0 4 0
$a1 5 0
$a2 6 0
$a3 7 0
$t0 8 268501040
$t1 9 268501000
$t2 10 0
$t3 11 3
$t4 12 0
$t5 13 0
$t6 14 0
$t7 15 0
$s0 16 4
$s1 17 13
$s2 18 13
$s3 19 2
$s4 20 5
$s5 21 0
$s6 22 0
$s7 23 0
$t8 24 268501044
$t9 25 268501040
$k0 26 0
$k1 27 0
$gp 28 268468224
$sp 29 2147479548
$fp 30 0
$ra 31 0
pc 4194428
hi 0
lo 0