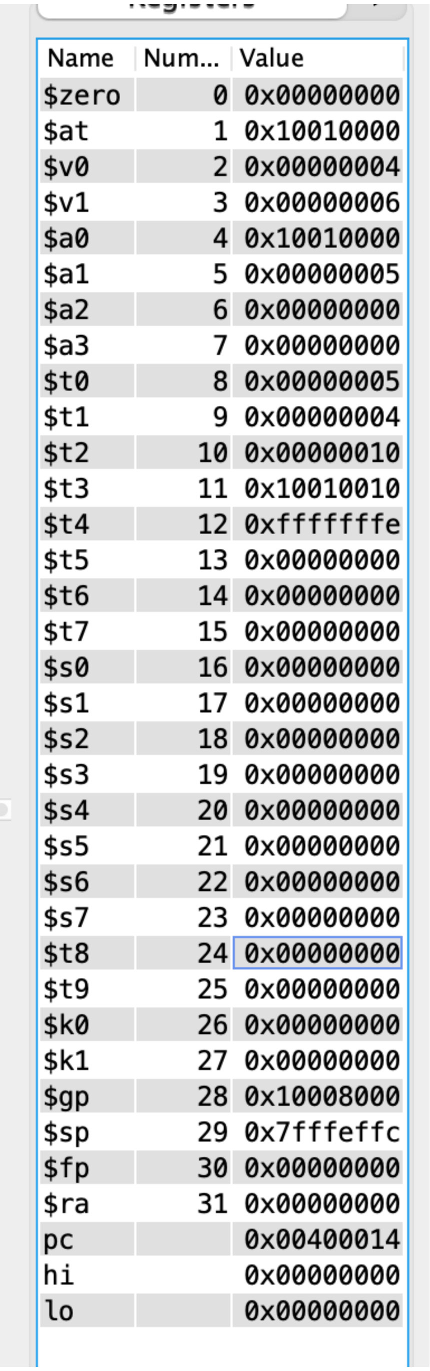
Week 6

Assignment 1

Source code:

```
#Laboratory Week 6, Assignment 1
.data
      A: .word -2, 6, -1, 3, -2    #Assign array value
.text
main: la $a0,A    #Load array address
      li $a1,5    #n = 5, Array length
      j mspfx
      nop
continue:
lock:       j lock
      nop
end_of_main:

#-------------------------------------------------------------------
#Procedure mspfx
# @brief find the maximum-sum prefix in a list of integers
# @param[in] a0 the base address of this list(A) need to be processed
# @param[in] a1 the number of elements in list(A)
# @param[out] v0 the length of sub-array of A in which max sum reachs.
# @param[out] v1 the max sum of a certain sub-array
#-------------------------------------------------------------------
#Procedure mspfx
#function: find the maximum-sum prefix in a list of integers
#the base address of this list(A) in $a0 and the number of
#elements is stored in a1
mspfx:      addi $v0,$zero,0 #initialize length in $v0 to 0
      addi $v1,$zero,0 #initialize max sum in $v1to 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:
```

Run results:



Name	Num...	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x00000004
$v1	3	0x00000006
$a0	4	0x10010000
$a1	5	0x00000005
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000005
$t1	9	0x00000004
$t2	10	0x00000010
$t3	11	0x10010010
$t4	12	0xfffffffe
$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
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400014
hi		0x00000000
lo		0x00000000

With the given array [-2, 6, -1, 3, -2], the max sum prefix is stored in $v1, which is 6. The sum is from the first 4 array element, which is -2, 6, -1 and 3.

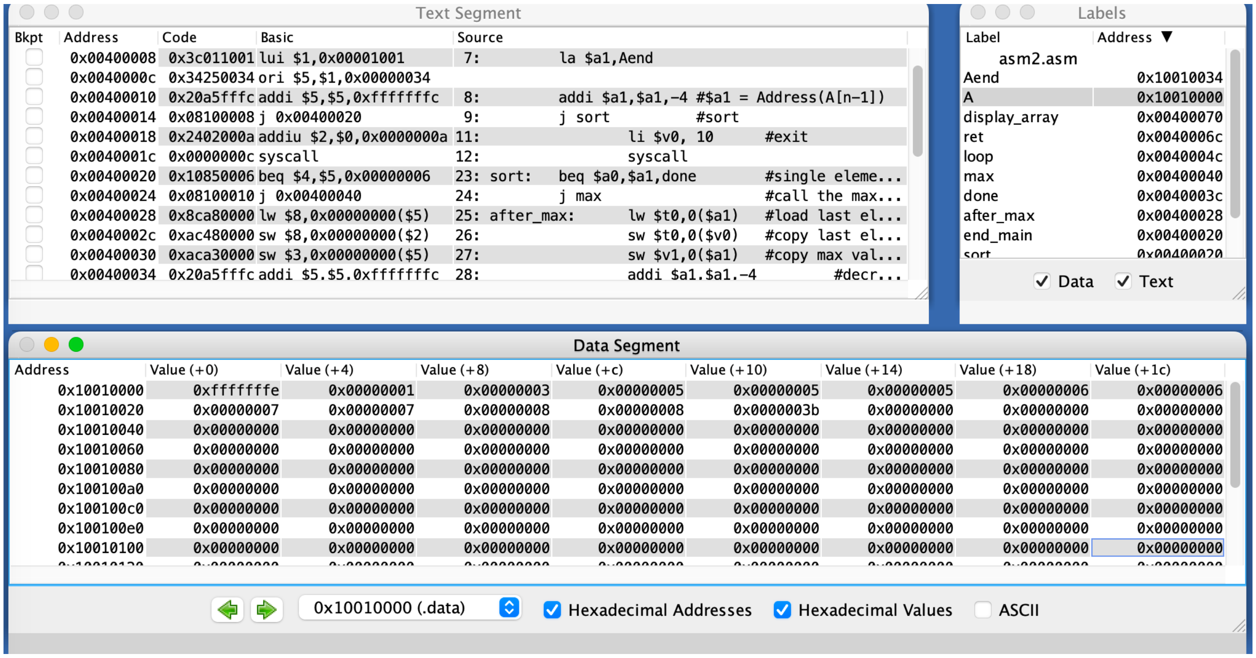
Assignment 2

Source code

```
#Laboratory Week 6, Assignment 2
.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 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

display_array:
    li $t0, 0  # Initialize loop counter to 0
```

Run results:



Text Segment

Bkpt	Address	Code	Basic	Source
□	0x00400008	0x3c011001	lui $1,0x00001001	7: la $a1,Aend
□	0x0040000c	0x34250034	ori $5,$1,0x00000034	
□	0x00400010	0x20a5fffc	addi $5,$5,0xfffffffc	8: addi $a1,$a1,-4 #$a1 = Address(A[n-1])
□	0x00400014	0x08100008	j 0x00400020	9: j sort #sort
□	0x00400018	0x2402000a	addiu $2,$0,0x0000000a	11: li $v0, 10 #exit
□	0x0040001c	0x0000000c	syscall	12: syscall
□	0x00400020	0x10850006	beq $4,$5,0x00000006	23: sort: beq $a0,$a1,done #single eleme...
□	0x00400024	0x08100010	j 0x00400040	24: j max #call the max...
□	0x00400028	0x8ca80000	lw $8,0x00000000($5)	25: after_max: lw $t0,0($a1) #load last el...
□	0x0040002c	0xac480000	sw $8,0x00000000($2)	26: sw $t0,0($v0) #copy last el...
□	0x00400030	0xaca30000	sw $3,0x00000000($5)	27: sw $v1,0($a1) #copy max val...
□	0x00400034	0x20a5fffc	addi $5.$5.0xfffffffc	28: addi $a1.$a1.-4 #decr...

Labels

Label	Address ▼
asm2.asm	
Aend	0x10010034
A	0x10010000
display_array	0x00400070
ret	0x0040006c
loop	0x0040004c
max	0x00400040
done	0x0040003c
after_max	0x00400028
end_main	0x00400020
sort	0x00400020

☑ Data ☑ Text

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0xfffffffe	0x00000001	0x00000003	0x00000005	0x00000005	0x00000005	0x00000006	0x00000006
0x10010020	0x00000007	0x00000007	0x00000008	0x00000008	0x0000003b	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	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

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

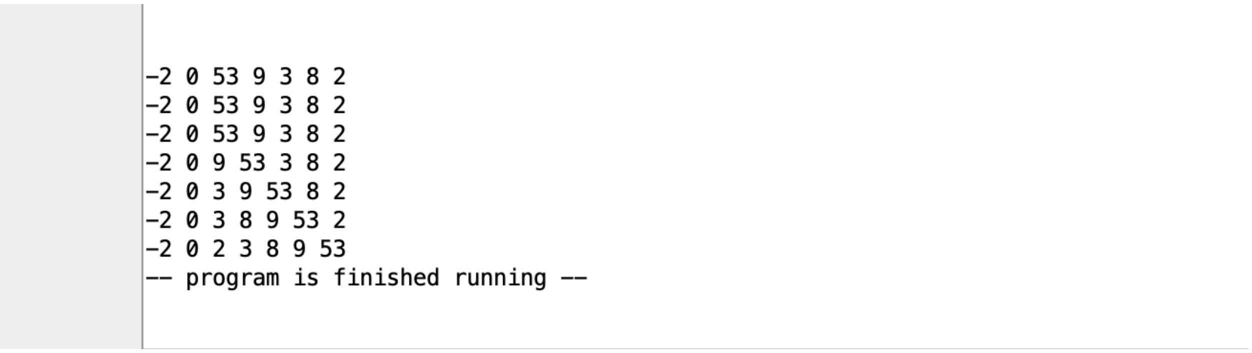
The sorted array is stored in word A with address range from 0x1001000 to 0x10010030

Assignment 3

Source Code and explanation

```
.data
A: .word -2 0 53 9 3 8 2     # Array A initialized with some values
Aend: .word                # End marker for array A
space: .asciiz " "         # String for space character
newline: .asciiz "\n"     # String for newline character

.text
    la $a0, A              # Load address of array A into $a0
    la $a1, Aend           # Load address of end marker for array A into
$a1
    addi $a1, $a1, -4      # Move $a1 to the last element of array A
    add $s0, $a0, $zero    # Copy address of array A to $s0

loop:
    lw $k0, 0($s0)         # Load current element of the array into $k0
    addi $t0, $s0, -4      # Move to the previous element in the array

while:
    slt $s1, $t0, $a0      # Set $s1 to 1 if $t0 < $a0, i.e., if we are at
the beginning of the array
    bne $s1, $zero, end_while    # Branch out of the while loop if we are
at the beginning of the array
    lw $t1, 0($t0)         # Load the element before the current element
into $t1
    slt $s1, $k0, $t1      # Set $s1 to 1 if $k0 < $t1
    beq $s1, $zero, end_while    # Branch out of the while loop if $k0 is
not less than $t1
Do:
    sw $t1, 4($t0)         # Store $t1 at the next position
    addi $t0, $t0, -4      # Move to the previous element in the array
    j while                # Jump to the beginning of the while loop
    nop

end_while:
    sw $k0, 4($t0)         # Store $k0 at the next position

Display:
    add $a2, $a0, $zero    # Copy address of array A to $a2
    add $s1, $a0, $zero    # Copy address of array A to $s1
    li $v0, 4              # Load the syscall number for printing string into $v0
    la $a0, newline        # Load the address of newline string into $a0
    syscall                # Print newline character

Print_integer:
    li $v0, 1              # Load the syscall number for printing integer into $v0
    lw $a0, 0($s1)         # Load the integer at the current position into $a0
    syscall                # Print the integer

    li $v0, 4              # Load the syscall number for printing string into $v0
    la $a0, space          # Load the address of space string into $a0
    syscall                # Print space character

    beq $s1, $a1, Display_loop   # Branch to OutPrint_running if $s1 equals $a1 (end of array)
    addi $s1, $s1, 4      # Move to the next integer in the array
    j Print_integer       # Jump to PrintInt
Display_loop:
    addi $a0, $a2, 0      # Copy the address of array A into $a0
    beq $s0, $a1, exit    # Branch to exit if $s0 equals $a1 (end of array)
    addi $s0, $s0, 4      # Move to the next integer in the array
    j loop                 # Jump to the beginning of the loop
    nop

exit:
    li $v0, 10             # Load the syscall number for exit into $v0
    syscall                # Exit the program
```

Run Results:



```
-2 0 53 9 3 8 2
-2 0 53 9 3 8 2
-2 0 53 9 3 8 2
-2 0 9 53 3 8 2
-2 0 3 9 53 8 2
-2 0 3 8 9 53 2
-2 0 2 3 8 9 53
-- program is finished running --
```