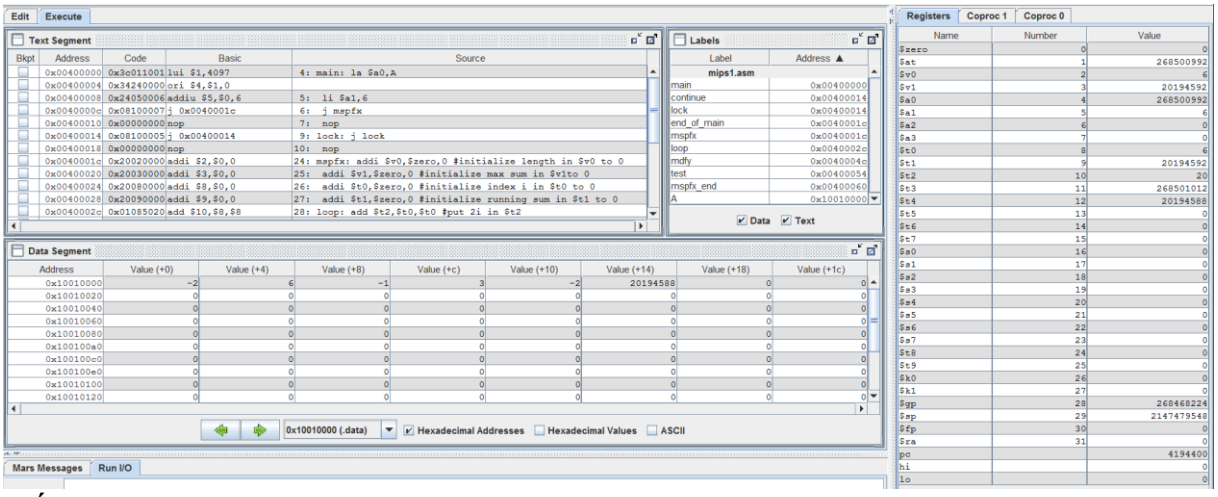
Assignment 1:

- Mã nguồn:

mips1.asm

```
.data
A: .word -2, 6, -1, 3, -2, 20194588
.text
main: la $a0,A
 li $a1,6
 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
#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:
```

- Màn hình chạy:



- Kết quả = 20194592 => Chính xác
- Giải thích:
  - Dòng 1,2: Khai báo mảng
  - Dòng 4: load địa chỉ A vào a0
  - Dòng 5: Gán a1 = 6 (= số phần tử)
  - Dòng 6: Chạy mspfx
  - Mspfx (dòng 24 – 35):
    - Dòng 24-27: Khởi tạo các giá trị
    - Dòng 28,30: Lấy từng phần tử của A
    - Dòng 31: load phần tử hiện tại (t3) vào t4
    - Dòng 32: This running sum
    - Dòng 33-34: So sánh running sum với max sum, nếu max sum nhỏ hơn => thực hiện modify (Gán lại max sum dòng 36,37)
    - Dòng 35:kiểm tra điều kiện lặp với hàm test
  - Dòng 41: Sau khi xong thì lặp vô hạn để chờ tín hiệu tiếp

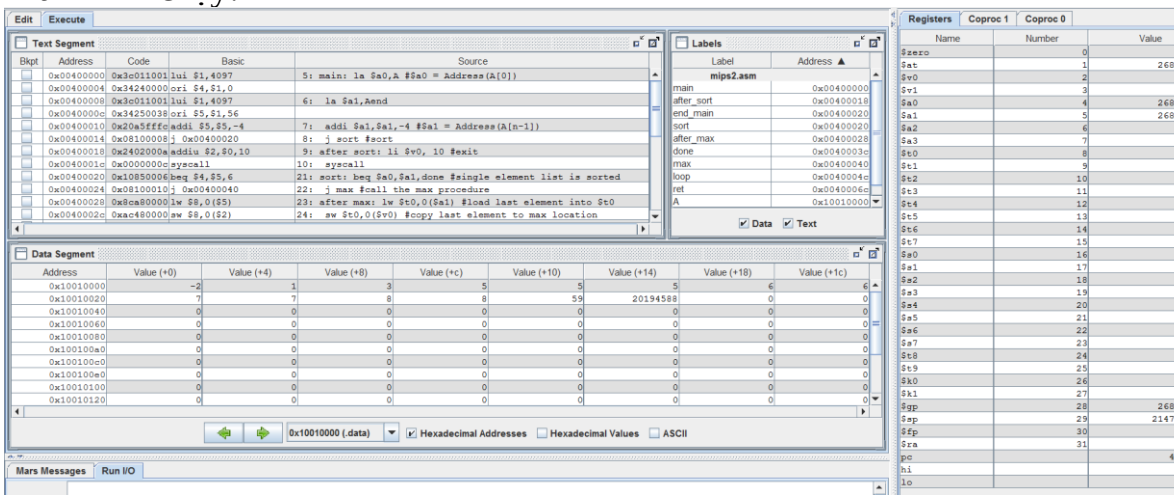
Assignment 2:

- Mã nguồn:

```
.data
A: .word 7, -2, 5, 1, 5,6,7,3,6,8,8,59,5, 20194588
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
```

```
43  slt $t2,$t1,$v1 #(next)<(max) ?
44  bne $t2,$zero,loop #if (next)<(max), repeat
45  addi $v0,$t0,0 #next element is new max element
46  addi $v1,$t1,0 #next value is new max value
47  j loop #change completed; now repeat
48  ret:
49  j after_max
50
```

- Màn hình chạy:



- Giải thích:
  - Dòng 1-3: Khai báo mảng
  - Dòng 5-7: load địa chỉ của phần tử đầu, cuối của A vào a0,a1
  - Dòng 8: Thực hiện hàm sort
  - Dòng 9-11: Sau khi sort xong thì end chương trình
  - Hàm sort (Dòng 21-22):Nếu còn 1 phần tử thôi thì nhảy đến hàm max
  - Hàm done (Dòng 28): Nhảy đến after_sort
  - Hàm after_sort: Thoát khỏi chương trình
  - Hàm loop (Dòng 39-47):
    - Dòng 40: So sánh next = last thì thoát
    - Dòng 41,42: Lấy phần tử tiếp theo
    - Dòng 42: So sánh next và max
    - Dòng 43: Nếu < => tiếp tục loop
    - Dòng 45,46: Cập nhận next và max mới
    - Dòng 47: Tiếp tục loop

Assignment 3:

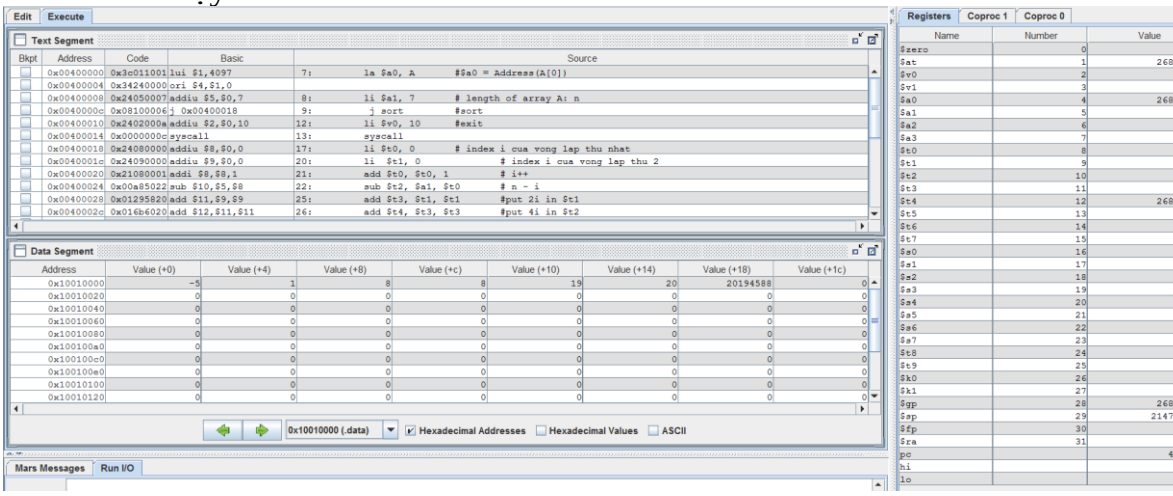
- Mã nguồn:

mips1.asm | mips2.asm | mips3.asm

```
1  # Bubble sort algorithm
2  .data
3          A: .word 20, 19, -5, 1, 8, 8, 20194588
4
5  .text
6  main:
7          la $a0, A         #$a0 = Address(A[0])
8          li $a1, 7         # length of array A: n
9           j sort #sort
10
11 after_sort:
12         li $v0, 10        #exit
13         syscall
14 end_main:
15
16 sort:
17         li $t0, 0         # index i cua vong lap thu nhat
18
19 loop_1:
20         li  $t1, 0                # index i cua vong lap thu 2
21         add $t0, $t0, 1           # i++
22        sub $t2, $a1, $t0        # n - i
23
24 loop_2:
25         add $t3, $t1, $t1         #put 2i in $t1
26         add $t4, $t3, $t3         #put 4i in $t2
27         add $t4, $t4, $a0
28         lw  $a2, 0($t4)           # load A[i]
29         lw  $a3, 4($t4)           # Load A[i+1]
30 if:
31         slt $t7, $a2, $a3         # If A[i] < A[i+1]
32         bne $t7, $zero, end_if
33
34         # Swap
35         sw $a3, 0($t4)
36         sw $a2, 4($t4)
37
38 end_if:
39
40         add $t1, $t1, 1
41
42         slt $s1, $t1, $t2         # neu t1 < t2 return 1; else return 0
```

```
43          beq $s1, $zero, end_loop_2
44          j   loop_2
45  end_loop_2:
46
47          slt $s2, $t0, $a1
48          beq $s2, $zero, end_loop_1
49          j   loop_1
50  end_loop_1:
```

- Màn hình chạy:



- Giải thích:
  - Dòng 1-3: Khai báo mảng
  - Dòng 7-8: load mảng vào a0 và gán a1 = số lượng phần tử
  - Dòng 9: nhảy đến hàm sort
  - Hàm sort(Dòng 16): Khởi tạo t0 (index i) = 0
  - Chạy loop_1 (Dòng 19): Khởi tạo index j của vòng lặp thứ 2, j++, t2 = n – j
  - loop_2 (Dòng 24): Lấy ra A[j], A[j+1]
  - if (Dòng 30): Tiến hành so sánh A[j] và A[j+1], nếu bé hơn thì end_if, còn > thì phải swap
  - end_if: Tăng j = i+j, so sánh với n-j để kiểm tra điều kiện lặp
  - end_loop_2: So sánh i với n để kiểm tra điều kiện lặp

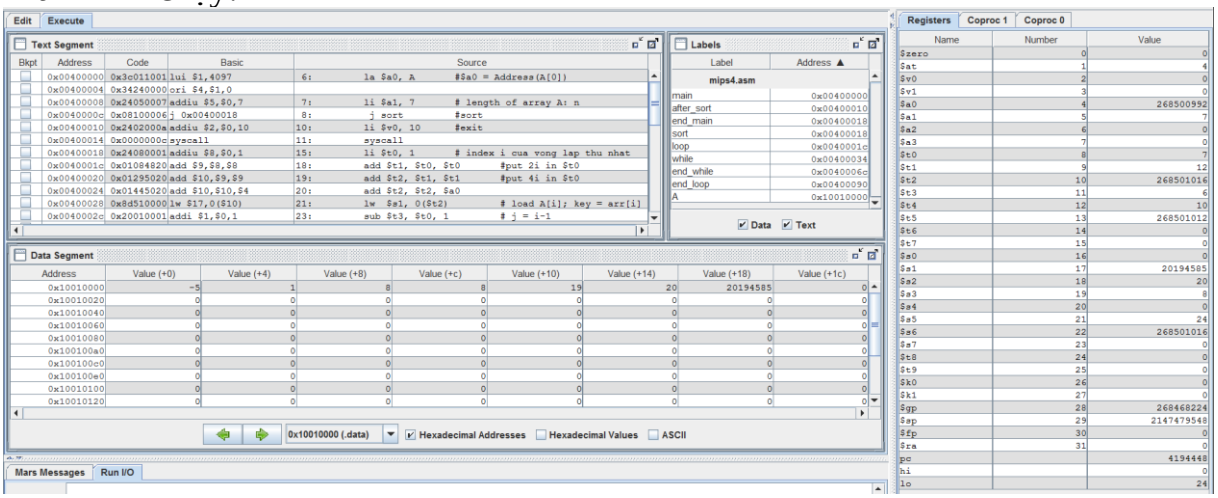
Assignment 4:

- Mã nguồn:

mips1.asm | mips2.asm | mips3.asm | mips4.asm

```
# Insertion sort algorithm
.data
        A: .word 20, 19, -5, 1, 8, 8, 20194585
.text
main:
        la $a0, A       #$a0 = Address(A[0])
        li $a1, 7       # length of array A: n
         j sort #sort
after_sort:
        li $v0, 10      #exit
        syscall
end_main:
sort:

        li $t0, 1       # index i cua vong lap thu nhat

loop:   # for i in range(1, len(arr))
        add $t1, $t0, $t0       #put 2i in $t0
        add $t2, $t1, $t1       #put 4i in $t0
        add $t2, $t2, $a0
        lw  $s1, 0($t2)         # load A[i]; key = arr[i]

         sub $t3, $t0, 1          # j = i-1

while:

        slt $t8, $t3, $zero     # if j < 0
        bne $t8, $zero, end_while

        add $t4, $t3, $t3       #put 2i in $t3
        add $t5, $t4, $t4       #put 4i in $t3
        add $t5, $t5, $a0
        lw  $s2, 0($t5)         #  arr[j]

        slt $t9, $s1, $s2       # key < arr[j]
        beq $t9, $zero, end_while

        # Swap
        lw $s3, 4($t5)
        sw $s3, 0($t5)          # arr[j + 1] = arr[j]
        sw $s2, 4($t5)

```

- Màn hình chạy:



- Giải thích:
  - Dòng 2,3: Khởi tạo mảng
  - Dòng 6,7: load địa chỉ của A vào a0 và khởi tạo độ dài của A vào a1
  - Dòng 8: Nhảy đến hàm sort
  - Hàm sort (Dòng 13-15): Khởi tạo i = 0
  - Hàm loop (Dòng 17-23) : Lấy ra A[i] và cho j = i-1
  - Hàm while (Dòng 25-44):
    - Dòng 27,28: So sánh j với 0, nếu <0 thì end_while
    - Dòng 30-33: Lấy ra A[j+1]
    - Dòng 35-42: So sánh A[j] và A[j+1], nếu > thì tiến hành end_while
    - đổi chỗ, giảm j
  - End_while (dòng 45):
    - Tăng j = j+1
    - Lấy ra A[j+1] và A[j] để đổi chỗ
    - Kiểm tra điều kiện I <n để end_loop