Laboratory Exercise 6

Array and Pointer

Trần Khánh Quỳnh – 20225762

1. Assignment 1

-Chương trình:

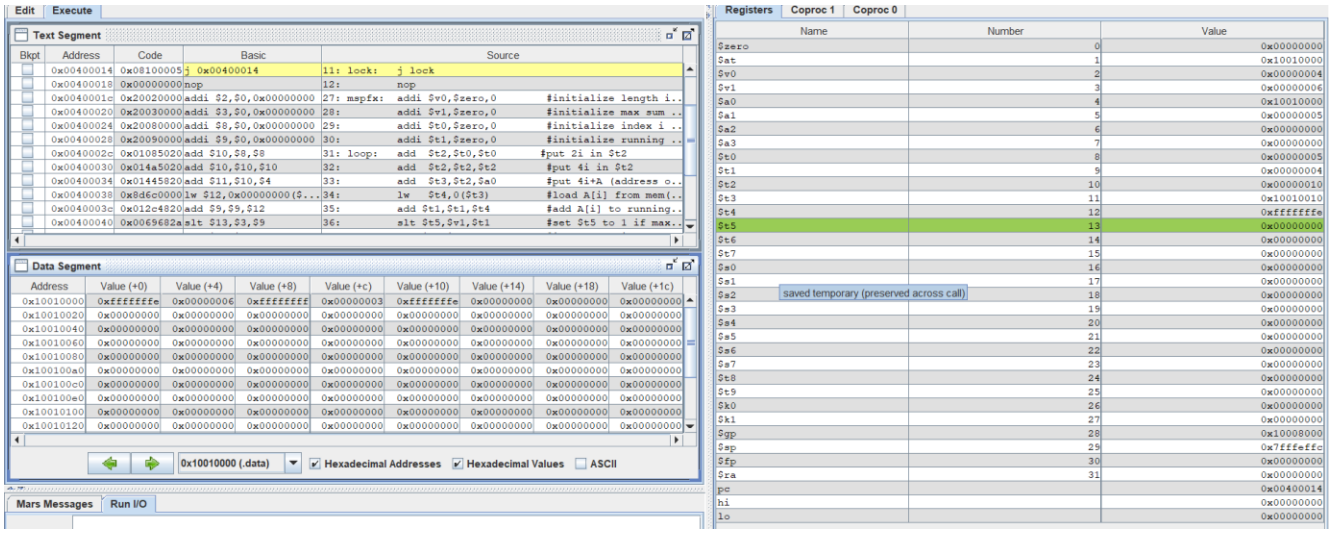
```
.data
A: .word -2,6, -1, 3, -2

.text
main:    la $a0, A
         li $a1, 5
         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:
```

-Kết quả chạy chương trình:



-Nhận xét kết quả: Dữ liệu nhập vào là mảng A[i] = {-2, 6, -1, 3, -2}. Kết quả tổng prefix lớn nhất của mảng là khi có độ dài bằng 4, kết quả tổng lớn nhất là 6 (6 = -2 + 6 + -2 + 3). Vì vậy, kết quả khi chạy cho ra $v0 = 0x4 và $v1 = 0x6

-Chạy từng bước chương trình:

+Chương trình trên tính tổng prefix lớn nhất trong 1 mảng có sẵn. Ở trong hàm main, chương trình nạp địa chỉ của mảng A vào thanh ghi $a0, đặt n (độ dài của mảng) là 5 vào thanh ghi $a1. Vì vậy, $a0 = 0x10010000 và $a1 = 0x5. Sau khi nạp xong 2 dữ liệu này thì chương trình nhảy đến hàm mspfx

+Trong hàm mspfx, chương trình khởi tạo độ dài để tính tổng bằng 0, khởi tạo tổng lớn nhất có giá trị = 0, i = 0 và tổng tạm thời = 0.

+ Chạy vòng lặp lần đầu tiên:

- Vì lúc này i = 0 nên 4i = 0 hay $t2 = 0
- Thanh ghi $t3 sẽ trỏ đến A[0], hay $t3 = 0x10010000
- Lưu giá trị từ địa chỉ lưu ở $t3 vào $t4 nên giá trị ở thanh ghi bằng -2 ($t4 = 0xfffffffe). Tổng thức thời $t1 = 0xfffffffe
- Vì maxsum > newsum nên chạy xuống hàm test, tăng i lên 1 →$t0 = 0x1. Tại hàm test kiểm tra xem i < n chưa. Lúc này i hay $t0 = 0x1 vẫn nhỏ hơn $a1= 0x5 nên $t5 = 0x1 và tiếp tục quay lại vòng lặp

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x10010000
$a1	5	0x00000005
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000001
$t1	9	0xfffffffe
$t2	10	0x00000000
$t3	11	0x10010000
$t4	12	0xfffffffe
$t5	13	0x00000001
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000

+Chạy vòng lặp lần thứ 2:

- Vì i = 1 → $t2 = 0x4; thanh ghi $t3 = 0x10010004 và trỏ đến A[1]
- Lưu giá trị từ địa chỉ lưu ở $t3 vào $t4 nên giá trị lúc này bằn A[1] = 6 ($t4 = 0x6). Tổng tức thời newsum $t1 = 0x00000004 (4 = -2 + 6), maxsum $v1 = 0x00000000
- Vì lúc này maxsum < newsum nên chạy xuống hàm mdfy. Ở đây tăng độ dài cần tính lên 1, tức $v0 lúc này = i + 1 = 2. Cập nhật maxsum = sum hiện tại, tức $v1 = 4.
- Sau khi chạy xong mdfy thì tiếp tục xuống hàm test, tăng i lên 1. Vì i = 2 vẫn nhỏ hơn 5 nên tiếp tục thực hiện lại vòng lặp

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x00000002
$v1	3	0x00000004
$a0	4	0x10010000
$a1	5	0x00000005
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000002
$t1	9	0x00000004
$t2	10	0x00000004
$t3	11	0x10010004
$t4	12	0x00000006
$t5	13	0x00000001
$t6	14	0x00000000
$t7	15	0x00000000

temporary (not preserved across call)

+Các vòng lặp còn lại chạy tương tự. Khi i = 5 = n thì chạy xuống done, rồi nhảy đến continue và chương trình bị khóa lại.

2. Assignment 2

-Chương trình:

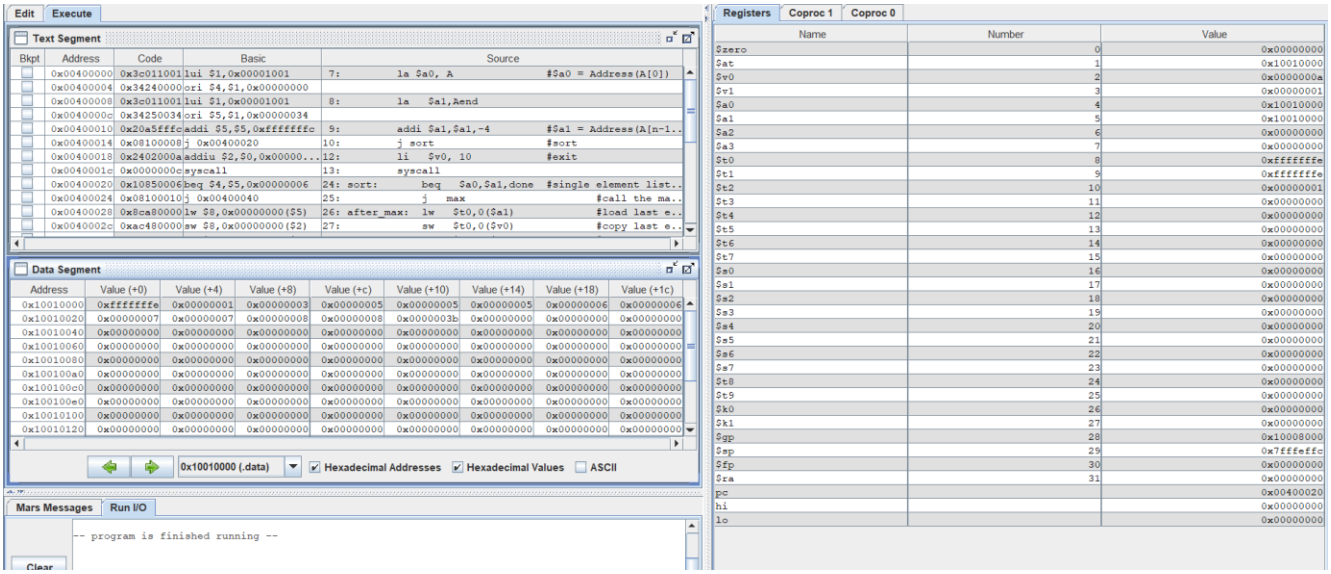
```
.data
A: .word 7, -2, 5, 1, 5, 6, 7, 3, 6, 8, 8, 59, 5
Aend: .word

.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
```

-Kết quả chạy:



-Nhận xét kết quả:

Đoạn chương trình sử dụng thuật toán Selection sort để sắp xếp mảng theo thứ tự tăng dần. Ý tưởng bài toán là tìm ra phần tử lớn nhất trong danh sách có n phần tử, hoán đổi nó với phần tử cuối cùng của danh sách rồi tiếp tục sắp xếp n-1 thành phần còn lại cho đến khi danh sách sắp xếp xong.

Mảng A[i] = { 7, -2, 5, 1, 5, 6, 7, 3, 6, 8, 8, 59, 5} sau khi được sắp xếp thì sẽ thành

{-2, 1, 3, 5, 5, 5, 6, 6, 7, 7, 8, 8, 59}

→ Chương trình chạy đúng, trong data segment ta thấy mảng đã được sắp xếp theo đúng thứ tự: 0xfffffffe, 0x1, 0x3, 0x5, 0x5, 0x5, 0x6, 0x6, 0x7, 0x7, 0x8, 0x8, và 0x3b

-Chạy từng bước:

+Đầu tiên chương trình lưu địa chỉ mảng A vào $a0, lưu địa chỉ của phần tử A[n-1] vào $a1 → $a0 = 0x10010000 và $a1 = 0x10010030

+Chương trình nhảy đến hàm sort. Tại hàm sort kiểm tra xem $a0 = $a1 hay không, tức kiểm tra xem đã sắp xếp hết tất cả phần tử trong mảng hay chưa. Do lúc này giá trị tại $a0 đang nhỏ hơn $a1 nên tiếp tục chạy tiếp lệnh bên dưới, nhảy đến hàm max chứ không nhảy xuống done

+Ở hàm max, chương trình khởi tạo con trỏ trỏ đến phần tử đầu tiên trong mảng → $v0 có giá trị là 0x10010000. Lưu giá trị tại địa chỉ con trỏ trỏ đến vào $v1, lúc này đang trỏ đến phần tử đầu tiên trong mảng nên $v1 = 0x7. Khởi tạo con trỏ next trỏ vào phần tử đầu trong mảng → $t0 = 0x10010000

+Chạy vòng lặp:

- Nếu như next = Aend thì sẽ nhảy đến return. Tuy nhiên hiện tại next != Aend nên tiếp tục chạy chương trình bên dưới. Cộng thêm 4 vào $t0 để nó trỏ đến phần tử tiếp theo, rồi lấy giá trị lưu được tại địa chỉ lưu ở $t0 gán vào $t1. → $t0 = 0x10010004 và $t1 = 0xfffffffe, tức trỏ đến A[1] và ta có A[1] = -2
- Kiểm tra xem giá trị mà con trỏ next trỏ vào có nhỏ hơn max không, cụ thể kiểm tra $t1 và $v1. Lúc này $t1 < $v1 nên $t2 = 0x1 → Tiếp tục thực hiện lại vòng lặp

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x10010000
$v1	3	0x00000007
$a0	4	0x10010000
$a1	5	0x10010030
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x10010004
$t1	9	0xfffffffe
$t2	10	0x00000001
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000

- Tại vòng lặp thứ 2 tiếp tục kiểm tra xem next = last hay không, rồi trỏ con trỏ đến giá trị kế tiếp là A[2]→ $t0 = 0x10010008. Lưu lại giá trị ở địa chỉ đó vào $t1 nên $t1 = 0x5. Kiểm tra next < max. Lúc này $t1 = 0x5, $v1 = 0x7 nên vẫn thỏa mãn và tiếp tục thực hiện lại vòng lặp
- Tại vòng lặp thứ 3, 4, 5, qua mỗi vòng thì $t0 tăng lên 4 đơn vị, lần lượt trỏ đến A[3], A[4], A[5] và lưu vào $t1. Next < max nên không cập nhật giá trị max mới
- Tại vòng lặp thứ 6, $t0 = 0x10010018, tức đang trỏ đến A[6] = 7 nên $t1 = 0x7. Vì lúc này next = max nên nhảy đến các lệnh bên dưới, $v0 lúc này trỏ đến A[6], $v0 = 0x10010018. Giá trị max mới là 7 nên $v1 = 0x7. Sau khi update giá trị mới thì tiếp tục quay trở lại vòng lặp
- Vòng lặp thứ 7, 8 thực hiện tương tự vòng lặp 3, 4, 5
- Vòng lặp 9, 10: do 8 lúc này là giá trị lớn nhất nên cập nhật giá trị lớn nhất tương tự vòng lặp 6
- Vòng lặp 11: 59 lúc này là giá trị lớn nhất nên $v1 = 0x3b, tương tự vòng 9, 10, 6
- Vòng lặp 12 tương tự 3,4,5,7,8

+Vậy lúc này giá trị lớn nhất là 59, $v1 = 0x59, $v0 = 0x1001002c. Thực hiện hàm after max, vì vậy nên lúc này ở địa chỉ 0x1001002c sẽ lưu giá trị 0x5 còn ở 0x10010034 sẽ

lưu giá trị 0x3b. Sau khi đưa phần tử lớn nhất xuống cuối thì thực hiện lại việc sắp xếp tương tự với phần tử còn lại, ta được kết quả như đã chạy ở trên.

3. Assignment 3:

-Chương trình:

(Sắp xếp tăng dần)

```
.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 of A[0]
    la $a1, Aend        # $a1 = Address of A[n]
    addi $a2, $a0, 4    # $a2 = Address of A[1] = i
    j sort
after_sort:
    li $v0, 10          #Exit
    syscall
end_main:
sort:
    beq $a2, $a1, done  # Check if the list is sorted or not
    j loop
after_loop:
    addi $a2, $a2, 4    # Increment pointer to the next element
    j sort
done:
    j after_sort        # If sorted, exit
loop:
    addi $t0, $a2, -4   # j = i - 1
    lw $t1, 0($a2)      # $t1 = A[i] (current element)
check1:
    slt $t2, $t0, $a0   # Check if j < 0 (start of array)
    bne $t2, $0, return
```

```
check2:
    lw $t3, 0($t0)      # $t3 = A[j] (previous element)
    slt $t2, $t1, $t3   # Check if A[j] > A[i]
    beq $t2, $0, return
    sw $t3, 4($t0)      # A[j+1] = A[j] (swap elements)
    addi $t0, $t0, -4   # j = j - 1
    j check1
return:
    sw $t1, 4($t0)      # A[j+1] = key (insert key in the correct position)
    j after_loop        # Continue with the next element
```

(Sắp xếp giảm dần)

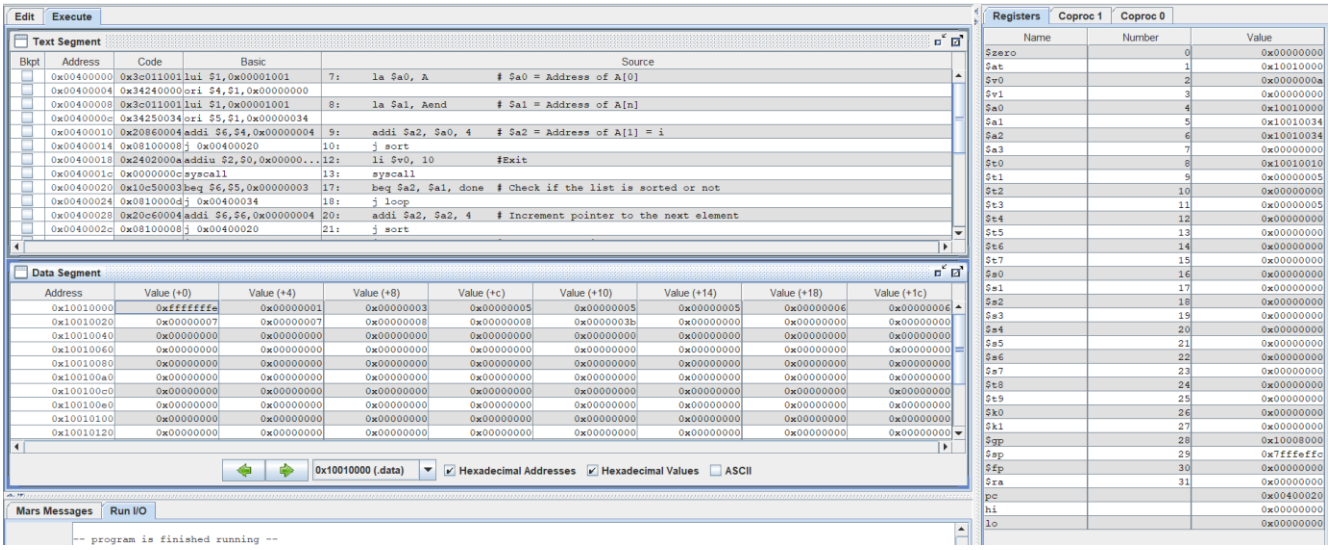
```
.data
A:      .word 7, -2, 5, 1, 5, 6, 7, 3, 6, 8, 8, 59, 5
Aend:   .word

.text
main:
    la $a0, A           # $a0 = Address of A[0]
    la $a1, Aend        # $a1 = Address of A[n]
    addi $a2, $a0, 4    # $a2 = Address of A[1] = i
    j sort
after_sort:
    li $v0, 10          #Exit
    syscall
end_main:
sort:
    beq $a2, $a1, done  # Check if the list is sorted or not
    j loop
after_loop:
    addi $a2, $a2, 4    # Increment pointer to the next element
    j sort
done:
    j after_sort        # If sorted, exit
loop:
    addi $t0, $a2, -4   # j = i - 1
```

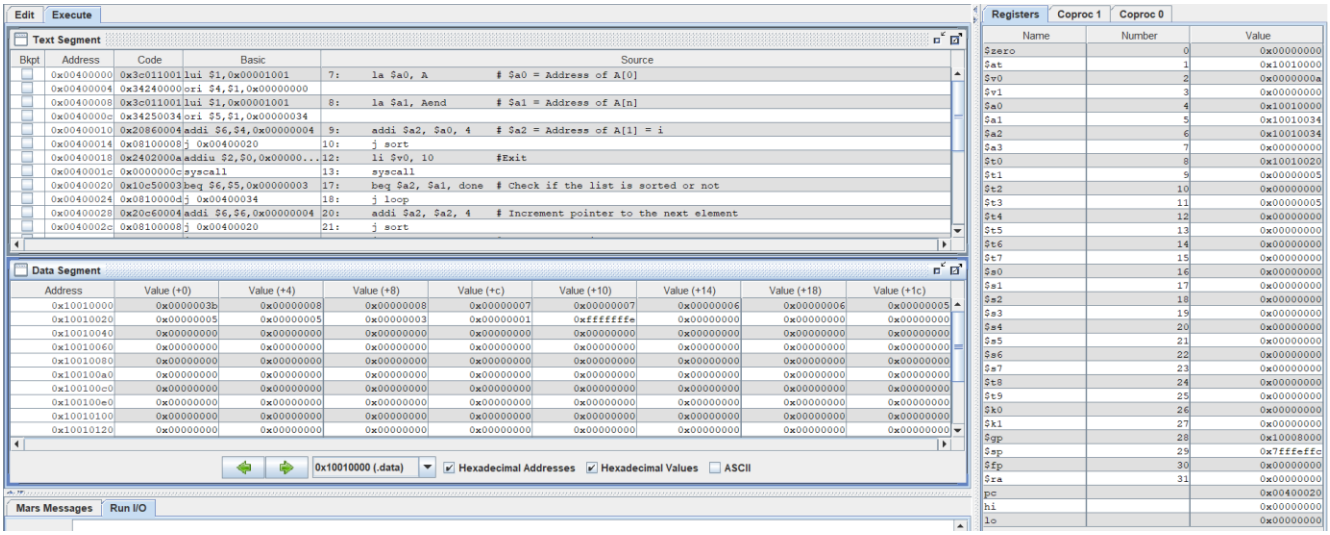
```
    lw $t1, 0($a2)       # $t1 = A[i] (current element)
check1:
    slt $t2, $t0, $a0    # Check if j < 0 (start of array)
    bne $t2, $0, return
check2:
    lw $t3, 0($t0)       # $t3 = A[j] (previous element)
    slt $t2, $t3, $t1  # Check if A[j] < A[i]
    beq $t2, $0, return
    sw $t3, 4($t0)       # A[j+1] = A[j] (swap elements)
    addi $t0, $t0, -4    # j = j - 1
    j check1
return:
    sw $t1, 4($t0)       # A[j+1] = key (insert key in the correct position)
    j after_loop         # Continue with the next element
```

-Kết quả chạy:

(Sắp xếp tăng dần)



(Sắp xếp giảm dần)



Edit | Execute

Text Segment

Bkpt	Address	Code	Basic		Source
	0x00400000	0x3c011001	lui $1,0x00001001	7:	la $a0, A # $a0 = Address of A[0]
	0x00400004	0x34240000	ori $4,$1,0x00000000		
	0x00400008	0x3c011001	lui $1,0x00001001	8:	la $a1, Aend # $a1 = Address of A[n]
	0x0040000c	0x34250034	ori $5,$1,0x00000034		
	0x00400010	0x20860004	addi $6,$4,0x00000004	9:	addi $a2, $a0, 4 # $a2 = Address of A[1] = i
	0x00400014	0x08100008	j 0x00400020	10:	j sort
	0x00400018	0x2402000a	addiu $2,$0,0x00000...	12:	li $v0, 10 #Exit
	0x0040001c	0x0000000c	syscall	13:	syscall
	0x00400020	0x10c50003	beq $6,$5,0x00000003	17:	beq $a2, $a1, done # Check if the list is sorted or not
	0x00400024	0x0810000d	j 0x00400034	18:	j loop
	0x00400028	0x20c60004	addi $6,$6,0x00000004	20:	addi $a2, $a2, 4 # Increment pointer to the next element
	0x0040002c	0x08100008	j 0x00400020	21:	j sort

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x0000003b	0x00000008	0x00000008	0x00000007	0x00000007	0x00000006	0x00000006	0x00000005
0x10010020	0x00000005	0x00000005	0x00000003	0x00000001	0xfffffffe	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
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

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

Mars Messages | Run I/O

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x10010000
$a1	5	0x10010034
$a2	6	0x10010034
$a3	7	0x00000000
$t0	8	0x10010020
$t1	9	0x00000005
$t2	10	0x00000000
$t3	11	0x00000005
$t4	12	0x00000000
$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		0x00400020
hi		0x00000000
lo		0x00000000