**Bài thực hành số 5**

Lớp: 147799 – Học phần: Thực Hành Kiến Trúc Máy Tính

Họ và tên: Lương Văn Khanh - MSSV: 20225728

**Asg1:**

* Code :

.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

# @param[in] a1 the number of elements in list(A)

# @param[out] v0 the length of sub-array of A in which max sum

# @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

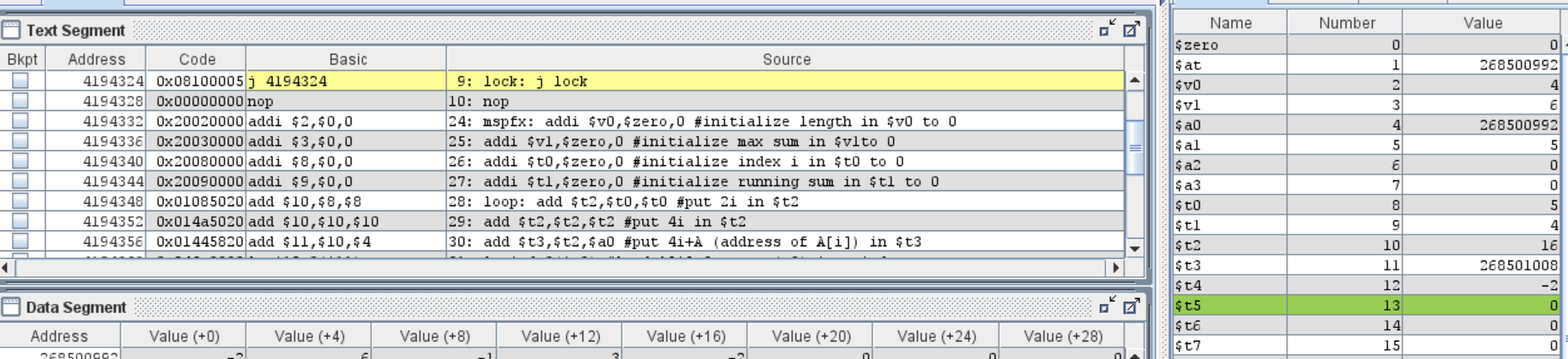
done: j continue

mspfx\_end:

* Ta khởi tạo mảng A gồm 5 phần tử . Tìm tổng các tiền tố lớn nhất trong mảng.
* Kết quả chạy : Thanh ghi $v1 hiện kết quả tổng lớn nhất là 6.

Thanh ghi $v0 hiện kết quả độ dài dãy con có tổng lớn nhất là: i=4.

Sau khi duyệt hết phần tử của mảng và không tìm thấy tổng lớn hơn, chương trình thực hiện lệnh jump to continue và kết thúc vòng lặp.



**Asg2:**

* Code:

.data

space: .asciiz " "

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

Aend: .word

length: .word 13

.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:

la $a3, A # Load the address of the array into $a0

lw $a1, length # Load the length of the array into $a1

# Loop to print each element of the array

print\_loop:

lw $t0, 0($a3) # Load the current element into $t0

# Print the current element

li $v0, 1

move $a0, $t0

syscall

# Print a space

li $v0, 4

la $a0, space

syscall

addiu $a3, $a3, 4 # Move to the next element

addiu $a1, $a1, -1 # Decrement the length

bgtz $a1, print\_loop # If length > 0, repeat the loop

# Exit the program

li $v0, 10

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

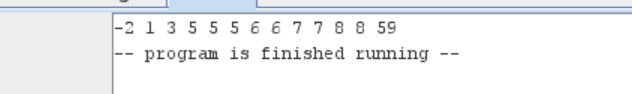
addi $v1,$t1,0 #next value is new max value

j loop #change completed; now repeat

ret:

j after\_max

* Khai báo mảng A có 13 phần tử . Kết quả mảng sau khi sắp xếp được in ra màn hình run I/O như sau



**Asg3:**

* Code thuật toán Bubble sort :

.data

space: .asciiz " "

array: .word 5, 3, 8, 4, 2, 1

length: .word 6 # Length of the array

.text

sort:

la $a0, array # Load the address of the array into $a0

lw $s0, length # Load the length of the array into $a1

li $t0,0 #i=0

outer\_loop:

bge $t0,$s0,end\_sort #end if done

addi $t1,$0,0 #y=0

addi $t0,$t0,1 #i++

inner\_loop:

sub $t8,$s0,$t0

bge $t1,$t8,outer\_loop

sll $t2, $t1, 2 #t2 = 4\*y

add $t3, $t2, $a0 #t3 = &A[y]

lw $t4, 0($t3) #t4 = A[y]

addi $t7,$t3,4 #t7 = &A[y+1]

lw $t5, 0($t7) #t5 = A[y+1]

slt $t6,$t4,$t5 #compare A[y] with A[y+1]

bnez $t6,next

sw $t4,0($t7) #swap A[y] and A[y+1]

sw $t5,0($t3)

next:

addi $t1,$t1,1 #y++

j inner\_loop

end\_sort:

print:

la $a3, array # Load the address of the array into $a0

lw $a1, length # Load the length of the array into $a1

# Loop to print each element of the array

loop:

lw $t0, 0($a3) # Load the current element into $t0

# Print the current element

li $v0, 1

move $a0, $t0

syscall

# Print a space

li $v0, 4

la $a0, space

syscall

addiu $a3, $a3, 4 # Move to the next element

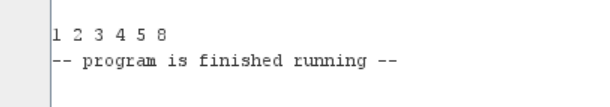
addiu $a1, $a1, -1 # Decrement the length

bgtz $a1, loop # If length > 0, repeat the loop

# Exit the program

li $v0, 10

syscall

* Nhập mảng ban đầu gồm 6 phần tử chưa được sắp xếp
* Kết quả sau khi chạy thuật toán : 

**Ass4:**

* Code thuật toán Insertion sort :

.data

space: .asciiz " "

A: .word 1, 8, 9, 2, 3

Aend: .word

.text

main:

la $a0, A #$a0 = Address(A[0])

la $a1, Aend #$a1 = Address(Aend)

addi $v0, $a0, 0 #init max pointer to first element

addi $a1, $a1, -4 #$a1 = Address(A[n-1])

j sort

after\_sort: li $v0, 10 #exit

syscall

end\_main:

sort: beq $a0, $a1, after\_sort #branch to after\_sort if element list is sorted

j swap #call the swap procedure

swap:

addi $a0, $a0, 4 #i=i+1

lw $s0, 0($a0) #$s0 = key = A[0]

addi $t0, $a0, -4 #j=i-1

after\_swap:

slt $t1, $t0, $v0 #$t1 = 1 if j < i

beq $t1, $zero, inner\_loop #branch to inner\_loop if $t1 = 0

j outer\_loop

inner\_loop:

lw $s1, 0($t0) #$s1 = A[j]

slt $t2, $s0, $s1 #St2=1 if A[i] < A[j]

beq $t2, $zero, outer\_loop #branch to outer\_loop if $t2 = 0

addi $v1, $t0, 4 #$v1 = j+1

sw $s1, 0($v1) #$s1 = A[j+1]

addi $t0, $t0, -4 #j=j-1

j after\_swap

outer\_loop:

addi $t0, $t0, 4 #j=j+1

sw $s0, 0($t0) #$s0 = key = A[j+1]

j sort

* Kết quả chạy : Mảng A sẽ được sắp xếp theo thứ tự tăng dần

