**REPORT LAB 6: ASSEMBLY LANGUAGE**

**STUDENT: PHI QUOC HUNG – 20184270**

**Assignment 1:**

**- Source 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-prefix sum in a list of integers

# @param[in] a0: thebase address of thelist(A) need to be processed

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

# @param[out]v0: the length of sub-listof Athat has the max sum.

# @param[out]v1: the max prefix sum

#-----------------------------------------------------------------

#Procedure mspfx

#function: find the maximum-prefix sum in a list of integers

#the base address of the list(A) is stored in $a0

#the number of elements is stored in a1

mspfx: addi $v0,$zero,0 #initialize the length in $v0 to0

addi $v1,$zero,0 #initialize the max sum in $v1to 0

addi $t0,$zero,0 #initialize the index i in $t0 to 0

addi $t1,$zero,0 #initialize the 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 the running sum in $t1

slt $t5,$v1,$t1 #set $t5 to 1 if max sum < new sum

bne $t5,$zero,mdfy #if the 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 #incrementthe 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:

- Explain the result:

In the mspfx function:

First, initialize all the parameters that we need to use while performing the algorithms: length, index, running sum and max sum. In the loop, we increase the value of $t2 to 4 unit corresponding to the increment of i. Then we load the value of $t3 to $t4, add that value to the running sum(stored in the register $t5).

We consider 2 cases:

+ If the running sum is greater than the max sum, increase value of length by 1 and store the running sum to $v1. After that we jump to test and check if the value of i is less than n. If not, end the loop

+ If the running sum is not greater than the max sum, run test and continue

The final result is:

$v0 – the length of max-prefix sum: 4

$v1 – the max prefix-sum: 6

**Assignment 2:**

**- Source 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:

#--------------------------------------------------------------

#sort procedure(selection sort using pointerfor ascending order)

#$a0: pointer to the first element in the unsorted part

#$a1: pointer to the last element in the unsorted part

#$t0: temporary place for the value of the last element

#$v0: pointer tothemax element in the unsorted part

#$v1: value of the max element in the unsorted part

#--------------------------------------------------------------

sort:

beq $a0, $a1, done #single-element list is sorted

j max #call 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 the last element

j sort #repeat sort for a smaller list

done:

j after\_sort

#---------------------------------------------------------------------

#Procedure max

#function: fax the value and address of max element in the list

#$a0 pointer to the first element

#$a1 pointer to the last element

#---------------------------------------------------------------------

max:

addi $v0, $a0, 0 #initializemax pointer to first element

lw $v1, 0($v0) #initializemax value to first value

addi $t0, $a0, 0 #initializenext pointer to first

loop:

beq $t0, $a1, ret #if next=last, return

addi $t0, $t0, 4 #move 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

- Explain the result:

First: At main, we load address of A and Aend, store the A[0] address and A[n-1] address. The array will be stored between those memory bound.

The sort function will stop when there is only 1 remainder element.

The max function init pointers and value to be considered.

The loop function:

+ Check the value of next is equal to last. If true, jump to ret

+ Move to next element and load it into $s1. Do it till next value is greater than current max value and assign that value to max value.

The ret function call the after\_max function:

+ Load last element to a temporary register $t0, copy its address and assign max value to that.

+ Then we decrease the pointer to 4 unit to sort a smaller array.

When we complete the loop, syscall to print out all the value

The final result will look like this:Graphical user interface, application, table, Excel

Description automatically generated

**Assginment 3:**

**- Source 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])

addi $a0,$a0,-4

la $a1, Aend

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

j sort #sort

after\_sort:

li $v0, 10

syscall

end\_main:

sort:

addi $a0, $a0, 4 #increase $a0 by 1 word

beq $a0, $a1, after\_sort #if $a0 = $a1, go to after\_sort function

addi $a2, $a0, 0 #store value of $a0 to $a2

j loop

loop:

beq $a2, $a1, sort #if $a2 = $a1, sort

lw $v0, 0($a0) #store the element into v0

addi $a2, $a2, 4 #increase value of $a2 to 4

lw $v1, 0($a2) #store the new value at $a2 to $v1

slt $t3, $v1, $v0 #if v1 < v0 so swap the 2 elements

bne $t3, $zero, swap

j loop

swap:

lw $t0,0($a0)

sw $t0,0($a2)

sw $v1,0($a0)

j loop

- Explain the code:

All of the implementation is the loop. Here we use 2 count variable as $a2 and $a0, and loop until the value at $a2 < $a0($v1 < $v0), then we swap 2 value at those registers.

I’m so sorry that I can’t complete the ex4 ☹ I will do it at the next lesson sir!