**­­­­Computer Architecture Lab Report Week 6**

**Full name: Nguyễn Hồng Phúc**

**Student ID: 20225659**

Assignment 1

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 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:**

**A screenshot of a computer

Description automatically generated**

Tổng dãy con lớn nhất $v1 = 6 => Đúng với lý thuyết.

Assignment 2

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:

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

#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:**

Nhập mảng có 13 phần tử

Các phẩn tử lúc chưa được sắp xếp là:

A screenshot of a computer

Description automatically generated

Các phần tử sau khi đã được sắp xếp:

A screenshot of a computer

Description automatically generated

=>Kết quả đúng với lý thuyết

Assignment 3

Ý tưởng của bubble sort như sau:  
1. Bắt đầu từ đầu danh sách, so sánh phần tử thứ i với phần tử thứ i+1.  
2. Nếu phần tử thứ i lớn hơn (hoặc nhỏ hơn) phần tử thứ i+1, hoán đổi  
chúng.  
3. Tiếp tục lặp lại bước 1 và 2 cho đến khi đi qua tất cả các phần tử trong  
danh sách.  
4. Lặp lại quá trình trên cho đến khi không có phần tử nào được hoán  
đổi nữa

**Code:**

.data

A: .word 4, 10, 2, -3, 9, 6

Aend: .word

.text

la $a0, A

la $a1, Aend

li $s0, 0 # count = 0 (count la bien dem phan tu)

li $s1, -1 # i = -1 (i trong loopi)

DemPhanTu:

beq $a1, $a0, Size # So sanh dia chi hien tai trong a1 voi dia chi co so cua mang A

addi $a1, $a1, -4 # dia chi a1 giam de den tung dia chi cua tung phan tu trong mang

addi $s0, $s0, 1 # So luong phan tu tang thêm 1

j DemPhanTu

Size:

addi $t0, $s0, -1 # t0 = So luong phan tu cua mang A - 1

loop1:

addi $s1, $s1, 1 # i++

li $s2, 0 # j = 0 (j trong loop 2)

beq $s1, $t0, Exit # Neu i = size - 1 thì thoat

loop2:

sub $t2, $t0, $s1 # t2 = (size - 1) - i

beq $s2, $t2, loop1 # Neu j = (size - 1) - i thi nhay den loop1

if\_swap:

sll $t3, $s2, 2 # Tính offset cua dia chi A[j]

add $s3, $a0, $t3 # Tính dia chi A[j]

lw $v0, 0($s3) # Load giá tri A[j]

addi $s3, $s3, 4 # Tính dia chi cua A[j+1]

lw $v1, 0($s3) # Load giá tri A[j+1]

sle $t4, $v0, $v1 # Neu A[j] <= A[j+1] thì t4 = 1;

# A[j] > A[j+1] thì t4 = 0

beq $t4, $zero, swap # t4 = 0 thì nhay den swap

addi $s2, $s2, 1 # j++

j loop2

swap: sw $v0, 0($s3) # Ghi A[j] vào A[j+1]

addi $s3, $s3, -4 # Tính dia chi cua A[j] = dia chi cua A[j+1] - 4

sw $v1, 0($s3) # Ghi A[j+1] vào A[j]

addi $s2, $s2, 1 # j++

j loop2

Exit:

li $v0, 10

syscall

Kết quả:

Mảng trước khi sắp xếp là 4, 10, 2, -3, 9, 6

A screenshot of a computer

Description automatically generated

Mảng sau khi được sắp xếp

A screenshot of a computer

Description automatically generated

Assignment 4

Ý tưởng của insertion sort như sau:  
1. Bắt đầu với phần tử đầu tiên của danh sách, coi như rằng danh sách  
con đầu tiên chỉ chứa phần tử đó.  
2. Lặp lại cho đến khi tất cả các phần tử được sắp xếp.  
3. Trong mỗi lần lặp, chọn một phần tử trong danh sách chưa được sắp  
xếp và chèn nó vào vị trí đúng trong danh sách con đã được sắp xếp  
trước đó.  
4. Sau khi chèn, danh sách con đã được sắp xếp được mở rộng một phần  
tử.

Code:

.data

A: .word 3, -4, 2, 12, 8, 44, 60, 6, -11, 100

Aend: .word

.text

la $a0, A

la $a1, Aend

li $s0, 0 # count = 0 (count la bien dem phan tu)

li $s1, 0 # key = 0

li $s2, 0 # j = 0

li $s3, 1 # i = 1

DemPhanTu:

beq $a1, $a0, Loop # So sanh dia chi hien tai trong a1 voi dia chi co so cua mang A

addi $a1, $a1, -4 # Dia chi a1 giam de den tung dia chi cua tung phan tu trong mang

addi $s0, $s0, 1 # So luong phan tu tang thêm 1

j DemPhanTu

Loop:

beq $s3, $s0, Exit # Neu i = So luong phan tu có trong mang thì thoát

sll $t0, $s3, 2 # Tính Offset cua dia chi A[i]

add $s4, $a0, $t0 # Tính dia chi cua A[i]

lw $s1, 0($s4) # Load giá tri A[i] = key

addi $s2, $s3, -1 # j = i - 1

While:

slt $t1, $s2, $zero # Neu j >= 0 thì t1 = 0

sll $t0, $s2, 2 # Tính offset cua dia chi A[j]

add $s5, $a0, $t0 # Tính dia chi cua A[j]

lw $t3, 0($s5) # Load giá tri A[j] = thanh ghi t3

sle $t4, $t3, $s1 # Neu key >= t3 thì t4 = 0

add $t1, $t1, $t4

bne $t1, $zero, loop\_continue # Neu t1 = 0 thì dung while

addi $s5, $s5, 4 # Tính dia chi cua A[j+1]

sw $t3, 0($s5) # Ghi giá tri A[j] vào A[j+1]

addi $s2, $s2, -1 # j = j - 1

j While

loop\_continue:

addi $s5, $s5, 4 # Tính địa chỉ của A[j+1]

sw $s1, 0($s5) # Ghi giá trị key vào A[j+1]

addi $s3, $s3, 1 # i++

j Loop

Exit: li $v0, 10

Syscall

Kết quả:

Mảng lúc chưa được sắp xếp: 3, -4, 2, 12, 8, 44, 60, 6, -11, 100

A screenshot of a computer

Description automatically generated

Mảng sau khi được sắp xếp:

A screenshot of a computer

Description automatically generated

=> Kết quả đúng với lý thuyết.