# THỰC HÀNH KIẾN TRÚC MÁY TÍNH

## IT-3280

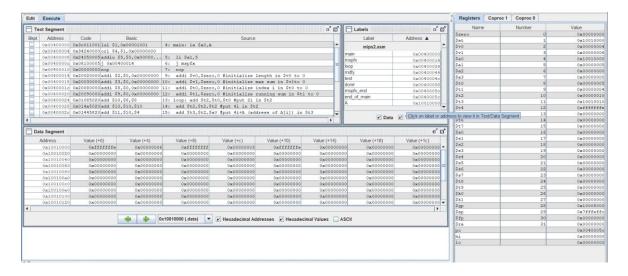
Họ tên	Thân Cát Ngọc Lan
MSSV	20225646

### **Assignment 1:**

### \*Code

```
mips3.asm
                        mips2.asm
 mips1.asm
   .data
   A: .word -2, 6, -1, 3, -2
   .text
   main: la $a0,A
 5
    li $a1,5
    j mspfx
 6
    nop
   mspfx:
9
    addi $v0,$zero,0 #initialize length in $v0 to 0
    addi $v1,$zero,0 #initialize max sum in $v1to 0
10
11
    addi $t0,$zero,0 #initialize index i in $t0 to 0
    addi $t1,$zero,0 #initialize running sum in $t1 to 0
12
   loop: add $t2,$t0,$t0 #put 2i in $t2
13
14
    add $t2,$t2,$t2 #put 4i in $t2
    add $t3,$t2,$a0 #put 4i+A (address of A[i]) in $t3
15
16
    lw $t4,0($t3) #load A[i] from mem(t3) into $t4
    add $t1,$t1,$t4 #add A[i] to running sum in $t1
17
18
    slt $t5,$v1,$t1 #set $t5 to 1 if max sum < new sum
    bne $t5,$zero,mdfy #if max sum is less, modify results
19
     j test #done?
20
   mdfy:
21
    addi $v0,$t0,1 #new max-sum prefix has length i+1
22
    addi $v1,$t1,0 #new max sum is the running sum
23
24
   test:
25
    addi $t0,$t0,1 #advance the index i
    slt $t5,$t0,$a1 #set $t5 to 1 if i<n
26
   bne $t5,$zero,loop #repeat if i<n
27
    done: j end of main
29
   mspfx end:
30
    end of main:
```

## \*Kết quả



### \*Giải thích:

Tổng quan: Tìm dãy tổng tiền tố lớn nhất. Đưa ra độ dài và tổng của dãy đó.

Mspfx: -> Khởi tạo các giá trị ban đầu bằng 0

- **\$v0:** độ dài dãy tổng tiền tố max

- \$v1: giá trị tổng max của dãy tổng tiền tố

- **\$t0:** chỉ số của mảng

- **\$t1:** tổng dãy tiền tố đang xét

Loop: -> vòng lặp duyệt mảng

- add \$t2,\$t0,\$t0
  add \$t2,\$t2,\$t2
  add \$t3,\$t2,\$a0
  - → Lấy địa chỉ của phần tử thứ i bằng phương pháp indexing (lấy địa chỉ của mảng A[0] + 4\*i). Vì 1 word chiếm 4byte trong bộ nhớ nên phải nhân 4.
- lw \$t4,0(\$t3)
  - → Load giá trị A[i]

Các bước tiếp theo thì tuân theo thuật toán: Cộng dồn vào tổng dãy tiền tố đang xét (\$t1), Kiểm tra xem \$t1 có tối ưu hơn tổng tiền tố max hiện tại (\$v0).

- bne \$t5,\$zero,mdfy

→ Nếu tối ưu hơn thì nhảy để thay đối giá trị tối ưu mới (\$v0)

Test: -> Kiểm tra điều kiện của vòng lặp. Nếu (i<n) thì nhảy tới loop.

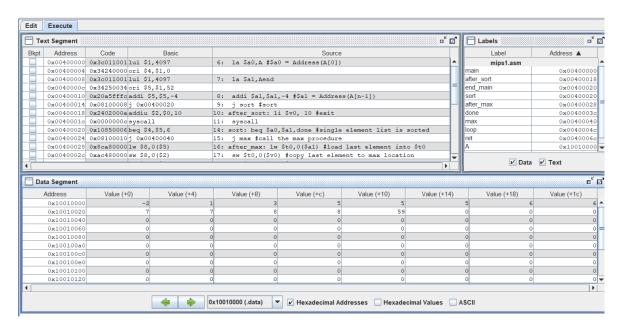
\*Nhận xét: Kết quả chạy đúng

## **Assignment 2:**

#### \*Code

```
.data
   A: .word 7,-2, 5, 1, 5,6,7,3,6,8,8,59,5
   Aend: .word
   . text
 5
   main:
    la $a0, A #$a0 = Address(A[0])
 7
    la $a1,Aend
8
    addi a1,a1,-4 #$a1 = Address(A[n-1])
    j sort #sort
   after sort: li $v0, 10 #exit
10
    syscall
11
   end main:
12
13
   sort: beq $a0,$a1,done #single element list is sorted
14
   j max #call the max procedure
15
16
   after max: lw $t0,0($a1) #load last element into $t0
17
    sw $t0,0($v0) #copy last element to max location
18
    sw $v1,0($a1) #copy max value to last element
    addi $a1,$a1,-4 #decrement pointer to last element
19
20
    j sort #repeat sort for smaller list
21
    done: j after sort
22
23
   max:
   addi $v0,$a0,0 #init max pointer to first element
24
   lw $v1,0($v0) #init max value to first value
   addi $t0,$a0,0 #init next pointer to first
26
   loop:
27
28 beq $t0,$a1,ret #if next=last, return
   addi $t0,$t0,4 #advance to next element
30 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
35 j loop #change completed; now repeat
36 ret:
   j after max
37
```

## \*Kết quả:



#### \*Giải thích:

Tổng quan: Sắp xếp mảng theo phương pháp Selection Sort.

## Trong code trên:

- Xác định phần tử lớn nhất trong mảng (\$v1) bằng cách duyệt mảng (loop)
- lw \$t0,0(\$a1) sw \$t0,0(\$v0) sw \$v1,0(\$a1)
  - → Hoán đổi phần tử max tìm được với phần tử có địa chỉ cuối của mảng
- addi \$a1,\$a1,-4
  - → Giảm địa chỉ của phần tử cuối. Mảng lúc này còn n-1 phần tử
- Kiểm tra xem địa chỉ của phần tử đầu(\$a0) có bằng địa chỉ phần tử cuối
   (\$a1). Nếu bằng chứng tỏ dãy đã sắp xếp. Ngược lại lặp lại các bước trên.

Thuật toán selection sort hoạt động với tốc độ O(n2) trong trường hợp xấu nhất.

\*Nhận xét: Duyệt mảng (loop) trong code trên dùng phương pháp pointer updating method. Bằng cách cộng (hoặc trừ) 4 địa chỉ của phần tử hiện tại để được

địa chỉ của phần tử tiếp theo (hoặc trước đó) của nó. Vì 1 word bằng 4 byte trong bộ nhớ.

### **Assignment 3:**

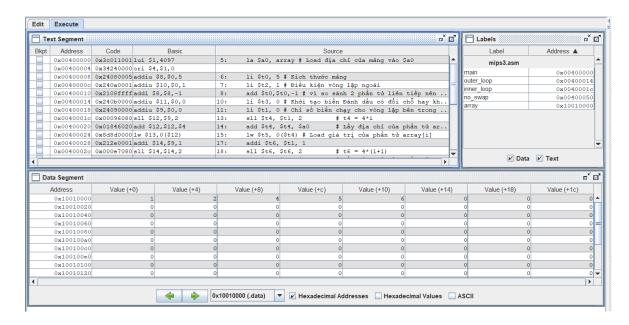
#### \*Code:

```
array: .word 5, 2, 4, 6, 1 # Mảng cần sắp xếp
    la $aO, array # Load địa chỉ của mảng vào $aO
    li $t0, 5 # Kich thước mảng
    li $t2, 1 # Điều kiện vòng lặp ngoài
   add $t0,$t0,-1 # vì so sánh 2 phần tử liên tiếp nên i chay từ 0 đến n-2
outer_loop:
   li $t3, 0 # Khới tạo biến Đánh dấu có đổi chỗ hay không?
    li $t1, 0 # Chỉ số biến chạy cho vòng lặp bên trong (đổi chỗ 2 phần tử liên tiếp)
inner loop:
    sll $t4, $t1, 2  # t4 = 4*i
add $t4, $t4, $a0  # Lây địa chỉ của phần tử array[i]
    lw $t5, O($t4) # Load giá trị của phần tử array[i]
    addi $t6, $t1, 1
    add $t6, $t6, 2  # t6 = 4*(i+1)

add $t6, $t6, $a0  # Láy địa chỉ của phần tử array[i+1]

lw $t7, 0($t6)  # Load giá trị của phần tử array[i]
    addi $t1, $t1, 1 # tăng chỉ số cho vòng lặp trong
    ble $t5, $t7, no_swap #So sánh giá trị 2 phần tử array[i] và array[i+1]
                      # t7 = array[i]
# t5 = array[i+1]
    sw $t7, 0($t4)
    sw $t5, 0($t6)
    li $t3, 1 # Có đổi chỗ
    blt $t1, $t0, inner_loop # Kiểm tra điều kiện vòng lặp trong
    beq $t3, 1, outer_loop # Kiểm tra điều kiện vòng lặp ngoài
    li v0, 10 # Nếu không có sự đổi chỗ trong vòng lặp cuối thì mảng đã được sắp xếp
```

## \*Kết quả:



\*Nhận xét: Mảng ban đầu (5,2,4,6,1) đã được sắp xếp tăng dần. Giải thích chi tiết trong code.