

# THỰC HÀNH KIẾN TRÚC MÁY TÍNH

IT-3280

Họ tên	Thân Cát Ngọc Lan
MSSV	20225646

## Assignment 1:

### \*Code

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

### \*Kết quả



➔ Nếu tối ưu hơn thì nhảy để thay đổi 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

```
1  .data
2  A: .word 7,-2, 5, 1, 5,6,7,3,6,8,8,59,5
3  Aend: .word
4  .text
5  main:
6      la $a0,A #$a0 = Address(A[0])
7      la $a1,Aend
8      addi $a1,$a1,-4 #$a1 = Address(A[n-1])
9      j sort #sort
10 after_sort: li $v0, 10 #exit
11      syscall
12 end_main:
13
14 sort: beq $a0,$a1,done #single element list is sorted
15      j max #call the max procedure
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
19      addi $a1,$a1,-4 #decrement pointer to last element
20      j sort #repeat sort for smaller list
21 done: j after_sort
22
23 max:
24      addi $v0,$a0,0 #init max pointer to first element
25      lw $v1,0($v0) #init max value to first value
26      addi $t0,$a0,0 #init next pointer to first
27 loop:
28      beq $t0,$a1,ret #if next=last, return
29      addi $t0,$t0,4 #advance to next element
30      lw $t1,0($t0) #load next element into $t1
31      slt $t2,$t1,$v1 #(next)<(max) ?
32      bne $t2,$zero,loop #if (next)<(max), repeat
33      addi $v0,$t0,0 #next element is new max element
34      addi $v1,$t1,0 #next value is new max value
35      j loop #change completed; now repeat
36 ret:
37      j after_max
38
```

## \*Kết quả:

Edit

Execute

Text Segment

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x3c011001	lui \$1,4097	6: la \$a0,A \$a0 = Address(A[0])
<input type="checkbox"/>	0x00400004	0x34240000	ori \$4,\$1,0	
<input type="checkbox"/>	0x00400008	0x3c011001	lui \$1,4097	7: la \$a1,Aend
<input type="checkbox"/>	0x0040000c	0x34250034	ori \$5,\$1,52	
<input type="checkbox"/>	0x00400010	0x20a5fffc	addi \$5,\$5,-4	8: addi \$a1,\$a1,-4 \$a1 = Address(A[n-1])
<input type="checkbox"/>	0x00400014	0x08100008	j 0x00400020	9: j sort #sort
<input type="checkbox"/>	0x00400018	0x2402000a	addiu \$2,\$0,10	10: after_sort: li \$v0, 10 #exit
<input type="checkbox"/>	0x0040001c	0x0000000c	syscall	11: syscall
<input type="checkbox"/>	0x00400020	0x10850006	beq \$4,\$5,6	14: sort: beq \$a0,\$a1,done #single element list is sorted
<input type="checkbox"/>	0x00400024	0x08100010	j 0x00400040	15: j max #call the max procedure
<input type="checkbox"/>	0x00400028	0x8ca80000	lw \$8,0(\$5)	16: after_max: lw \$t0,0(\$a1) #load last element into \$t0
<input type="checkbox"/>	0x0040002c	0xac480000	sw \$8,0(\$2)	17: sw \$t0,0(\$v0) #copy last element to max location

Labels

Label	Address
mips1.asm	
main	0x00400000
after_sort	0x00400018
end_main	0x00400020
sort	0x00400020
after_max	0x00400028
done	0x0040003c
max	0x00400040
loop	0x0040004c
ret	0x0040006c
A	0x10010000

☒ Data ☒ Text

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)	
0x10010000	-2	1	3	5	5	5	6	6	
0x10010020	7	7	8	8	59	0	0	0	
0x10010040	0	0	0	0	0	0	0	0	
0x10010060	0	0	0	0	0	0	0	0	
0x10010080	0	0	0	0	0	0	0	0	
0x100100a0	0	0	0	0	0	0	0	0	
0x100100c0	0	0	0	0	0	0	0	0	
0x100100e0	0	0	0	0	0	0	0	0	
0x10010100	0	0	0	0	0	0	0	0	
0x10010120	0	0	0	0	0	0	0	0	

0x10010000 (.data)

☒ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

## \*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  $O(n^2)$  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:

```
.data
array: .word 5, 2, 4, 6, 1 # Mảng cần sắp xếp
.text
main:
    la $a0, array # Load địa chỉ của mảng vào $a0
    li $t0, 5 # Kích thước mảng
    li $t2, 1 # Điều kiện vòng lặp ngoài
    add $t0, $t0, -1 # vì so sánh 2 phần tử liên tiếp nên i chạy từ 0 đến n-2
outer_loop:
    li $t3, 0 # Khởi 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 # Lấy địa chỉ của phần tử array[i]
    lw $t5, 0($t4) # Load giá trị của phần tử array[i]

    addi $t6, $t1, 1
    sll $t6, $t6, 2 # t6 = 4*(i+1)
    add $t6, $t6, $a0 # Lấy địa chỉ của phần tử array[i+1]
    lw $t7, 0($t6) # Load giá trị của phần tử array[i+1]

    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]

    sw $t7, 0($t4) # t7 = array[i]
    sw $t5, 0($t6) # t5 = array[i+1]

    li $t3, 1 # Có đổi chỗ

no_swap:
    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
    syscall
```

#### \*Kết quả:

