Báo cáo tuần 6
Thực hành kiến trúc máy tính

Họ tên: Phan Minh Anh Tuấn
MSSV: 20205227

Mục lục

1 Assignment 1

```
.data
A: .word 2 0 -2 0 5 2 2 -7
.text
main:
        la $a0,A
        li $a1,8
        j mspfx
        nop
mspfx:
        addi $v0,$zero,0 #initialize length in $v0 to 0
        addi $v1,$zero,0 #initialize max sum in $v1 to 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
```

Hình 1: Code của Assignment 1

Trong đó:

- $a0 lưu địa chỉ phần tử đầu của mảng A
- $a1 lưu số phần tử của mảng A
- mspfx là chương trình khởi tạo các biến độ dài mảng, tổng lớn nhất, index và tổng hiện thời
- loop là vòng lặp thực hiện để tính tổng
- mdfy là chương trình cập nhật tổng lớn nhất
- test là chương trình kiểm tra điều kiện lặp

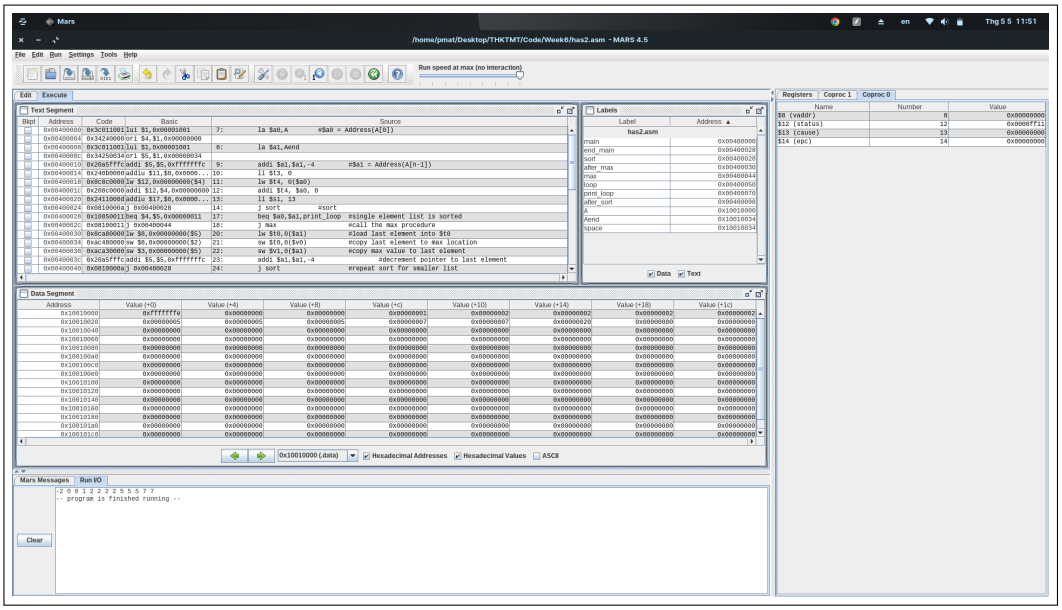
Giải thích thuật toán: Đi từ đầu đến cuối mảng, nếu max sum ($v1) $<$ running sum ($t1) thì cập nhật max sum = running sum, nếu không vẫn giữ nguyên.

2 Assignment 2

```
.data
A: .word 7, -2, 5, 1, 5, 2, 0, 2, 0, 5, 2, 2, 7
Aend: .word
space: .asciiz " "
.text
main:
        la $a0,A        #$a0 = Address(A[0])
        la $a1,Aend
        addi $a1,$a1,-4         #$a1 = Address(A[n-1])
        li $t3, 0
        lw $t4, 0($a0)
        addi $t4, $a0, 0
        li $s1, 13
        j sort          #sort
end_main:
sort:
        beq $a0,$a1,print_loop  #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
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,after_max           #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

print_loop:
        li $v0, 1
        lw $a0, 0($t4)
        syscall
        li $v0, 4
        la $a0, space
        syscall
        addi $t4, $t4, 4
        addi $t3, $t3, 1
        bne $t3, $s1, print_loop
after_sort:
        li $v0, 10      #exit
        syscall
```

Hình 2: Code của Assignment 2

Giải thích thuật toán Selection Sort: Bắt đầu bằng việc chọn vị trí đầu làm mốc, đi từ đầu đến cuối mảng (trừ vị trí mốc) duyệt tìm phần tử bé nhất. Sau khi duyệt xong đổi chỗ phần tử bé nhất và phần tử tại vị trí mốc, lúc này vị trí mốc đã được xếp đúng vị trí. Tiếp tục với các phần tử vị trí cao hơn.



Hình 3: Kết quả của Assignment 2

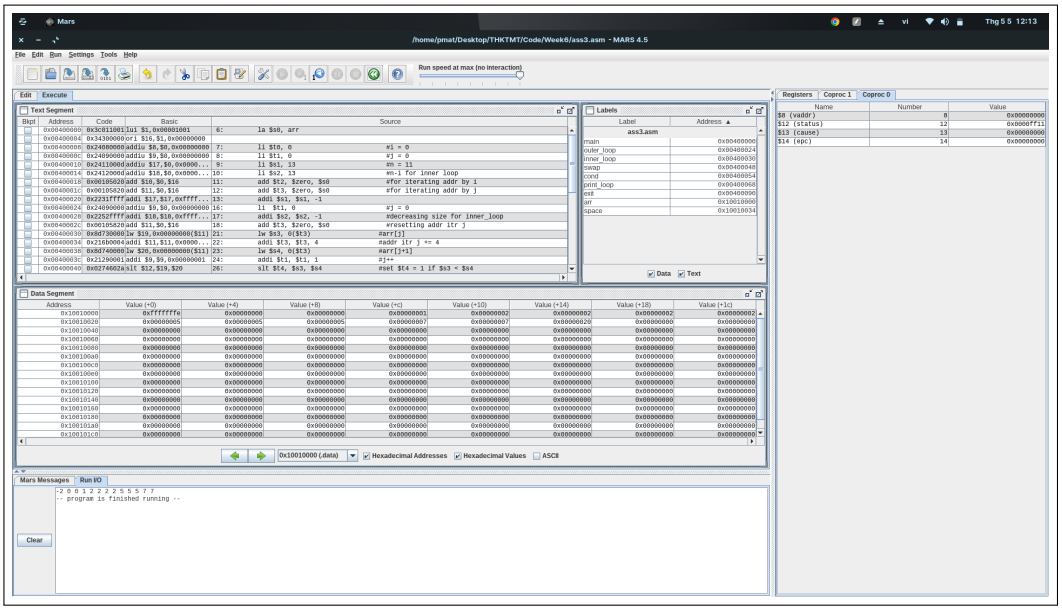
Mảng đầu vào: 7, -2, 5, 1, 5, 2, 0, 2, 0, 5, 2, 2, 7
Mảng sau khi đã sắp xếp: -2, 0, 0, 1, 2, 2, 2, 2, 5, 5, 5, 7, 7

3 Assignment 3

```
.data
arr: .word 7, -2, 5, 1, 5, 2, 0, 2, 0, 5, 2, 2, 7
space: .asciiz " "
.text
main:
        la $s0, arr
        li $t0, 0                       #i = 0
        li $t1, 0                       #j = 0
        li $s1, 13                      #n = 11
        li $s2, 13                      #n-i for inner loop
        add $t2, $zero, $s0             #for iterating addr by i
        add $t3, $zero, $s0             #for iterating addr by j
        addi $s1, $s1, -1

outer_loop:
        li  $t1, 0                      #j = 0
        addi $s2, $s2, -1               #decreasing size for inner_loop
        add $t3, $zero, $s0             #resetting addr itr j

inner_loop:
        lw $s3, 0($t3)                 #arr[j]
        addi $t3, $t3, 4               #addr itr j += 4
        lw $s4, 0($t3)                 #arr[j+1]
        addi $t1, $t1, 1               #j++

        slt $t4, $s3, $s4              #set $t4 = 1 if $s3 < $s4
        bne $t4, $zero, cond
swap:
        sw $s3, 0($t3)
        sw $s4, -4($t3)
        lw $s4, 0($t3)

cond:
        bne $t1, $s2, inner_loop      #j != n-i
        addi $t0, $t0, 1                #i++
        bne $t0, $s1, outer_loop         #i != n
        li $t0, 0
        addi $s1, $s1, 1

print_loop:
        li $v0, 1
        lw $a0, 0($t2)
        syscall
        li $v0, 4
        la $a0, space
        syscall

        addi $t2, $t2, 4                #addr itr i += 4
        addi $t0, $t0, 1                #i++
        bne $t0, $s1, print_loop        #i != n

exit:
        li $v0, 10
        syscall
```

Hình 4: Code của Assignment 3

Giải thích thuật toán Bubble Sort: Đi từ đầu đến cuối mảng so sánh hai phần tử kề nhau, nếu chúng chưa đứng đúng thứ tự thì đổi chỗ (swap). Có thể tiến hành từ trên xuống (bên trái sang) hoặc từ dưới lên



Hình 5: Kết quả của Assignment 3

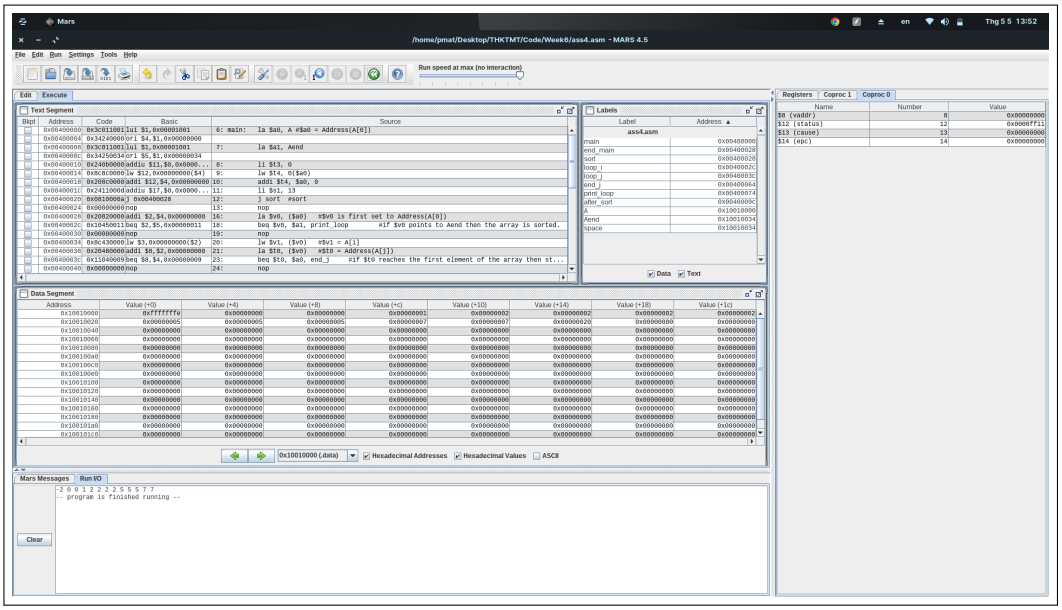
Mảng đầu vào: 7, -2, 5, 1, 5, 2, 0, 2, 0, 5, 2, 2, 7
Mảng sau khi đã sắp xếp: -2, 0, 0, 1, 2, 2, 2, 2, 5, 5, 5, 7, 7

4 Assignment 4

```
.data
A: .word 7, -2, 5, 1, 5, 2, 0, 2, 0, 5, 2, 2, 7
Aend: .word
space: .asciiz " "
.text
main:   la $a0, A #$a0 = Address(A[0])
        la $a1, Aend
        li $t3, 0
        lw $t4, 0($a0)
        addi $t4, $a0, 0
        li $s1, 13
        j sort  #sort
        nop
end_main:
sort:
        la $v0, ($a0)   #$v0 is first set to Address(A[0])
loop_i:
        beq $v0, $a1, print_loop        #if $v0 points to Aend then the array is sorted.
        nop
        lw $v1, ($v0)   #$v1 = A[i]
        la $t0, ($v0)   #$t0 = Address(A[j])
loop_j:
        beq $t0, $a0, end_j     #if $t0 reaches the first element of the array then stop looping
        nop
        lw $t1, -4($t0) #$t1 = A[j - 1]
        slt $t2, $v1, $t1       #A[i](tmp) < A[j - 1]?
        beq $t2, $0, end_j      #if true then push A[j - 1] to A[j], else stop looping
        nop
        sw $t1, ($t0)   #A[j] = A[j -1]
        addi $t0, $t0, -4       #j--
        j loop_j
        nop
end_j:
        sw $v1, ($t0)   #A[j] = tmp, store the value to the empty element
        addi $v0, $v0, 4        #i++
        j loop_i
        nop

print_loop:
        li $v0, 1
        lw $a0, 0($t4)
        syscall
        li $v0, 4
        la $a0, space
        syscall
        addi $t4, $t4, 4
        addi $t3, $t3, 1
        bne $t3, $s1, print_loop
after_sort:
        li $v0, 10      #exit
        syscall
```

Hình 6: Code của Assignment 4

Giải thích thuật toán Insertion Sort: muốn sắp mảng theo trật tự, ta bắt đầu từ phần tử thứ 2, so với các phần tử đứng trước nó để chèn vào vị trí thích hợp.



Hình 7: Kết quả của Assignment 4

Mảng đầu vào: 7, -2, 5, 1, 5, 2, 0, 2, 0, 5, 2, 2, 7
Mảng sau khi đã sắp xếp: -2, 0, 0, 1, 2, 2, 2, 2, 5, 5, 5, 7, 7