Bài thực hành số 6

Lớp: 139365 – Học phần: Thực hành Kiến Trúc Máy Tính

Họ và tên: Đinh Thị Hồng Phúc **MSSV: 20215118**

Bài 1.

```
#Assignment 1
.data
        A: .word -2, 6, -1, 3, -2
        B: .asciiz "\nMax = "
        C: .asciiz " "
.text
main:
        la $a0, A
        li $a1, 5
        j mspfx
        nop
lock:
        j lock
        nop
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?
```

```
29  mdfy:
30          addi $v0, $t0, 1 #new max-sum prefix has length i+1
31          addi $v1, $t1, 0 #new max sum is the running sum
32  test:
33          addi $t0, $t0, 1 #advance the index i
34          slt $t5, $t0, $a1 #set $t5 to 1 if i<n
35          bne $t5, $zero, loop #repeat if i<n
36  done:
37          j continue
38  mspfx_end:
39  continue:
40  end_of_main:
41          move $s0, $v0
42          li $t0, 0
43          la $s1, A
44  print:
45          add $t2, $t0, $t0
46          add $t2, $t2, $t2
47          add $t3, $t2, $s1
48          lw $t4, 0($t3)
49
50          li $v0, 1
51          move $a0, $t4
52          syscall
```

```
54          li $v0, 4
55          la $a0, C
56          syscall
57
58          addi $t0, $t0, 1
59          slt $t5, $t0, $s0
60          bne $t5, $zero, print
61
62          li $v0, 4
63          la $a0, B
64          syscall
65
66          li $v0, 1
67          move $a0, $v1
68          syscall
```

*Thực hiện gõ chương trình vào công cụ **MARS***

Giải thích:

- Hàm *main* khai báo thanh ghi a0 khởi tạo là giá trị địa chỉ của mảng, a1 khởi tạo là số lượng phần tử của mảng
- Hàm *mspfx* để tìm tiền tố có tổng lớn nhất. Trong vòng lặp, sử dụng các thanh ghi để lưu trữ các giá trị trung gian và điều khiển các bước tính toán.

Vòng lặp này duyệt quá các phần tử trong mảng, tính tổng các phần tử từ đầu đến phần tử hiện tại và so sánh giá trị của tổng này với giá trị lớn nhất đã tìm thấy cho đến thời điểm hiện tại. Nếu tổng mới lớn hơn tổng lớn nhất đã tìm thấy thì cập nhật độ dài dãy con và giá trị tổng lớn nhất hiện tại

- Sau cùng, in ra giá trị các phần tử trong dãy con lớn nhất bằng cách sử dụng vòng lặp (*print*) và hàm syscall để in ra từng số nguyên và tổng lớn nhất

```
-2 6 -1 3
Max = 6
```

*Thực hiện chạy chương trình với **MARS***

Bài 2.

```
#Assignment 2
.data
        A: .word 7, -2, 5, 1, 5, 6, 7, 3, 6, 8, 8, 59, 5
        Aend: .word
        B: .asciiz " "
.text
main:
        la $a0, A #$a0 = Address(A[0])
        la $a1, Aend
        addi $a1, $a1, -4 #$a1 = Address(A[n-1])
        j sort #sort
end_main:
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 end
```

```
24  max:
25          addi $v0, $a0, 0 #init max pointer to first element
26          lw $v1, 0($v0) #init max value to first value
27          addi $t0, $a0, 0 #init next pointer to first
28  loop:
29          beq $t0, $a1, ret #if next=last, return
30          addi $t0, $t0, 4 #advance to next element
31          lw $t1, 0($t0) #load next element into $t1
32          slt $t2, $t1, $v1 #(next)<(max) ?
33          bne $t2, $zero, loop #if (next)<(max), repeat
34          addi $v0, $t0, 0 #next element is new max element
35          addi $v1, $t1, 0 #next value is new max value
36          j loop #change completed; now repeat
37  ret:
38          j after_max
39  end:
40          li $s0, 13
41          li $t0, 0
42          la $s1, A
```

```
43  print:
44          add $t2, $t0, $t0
45          add $t2, $t2, $t2
46          add $t3, $t2, $s1
47          lw $t4, 0($t3)
48
49          li $v0, 1
50          move $a0, $t4
51          syscall
52
53          li $v0, 4
54          la $a0, B
55          syscall
56
57          addi $t0, $t0, 1
58          slt $t5, $t0, $s0
59          bne $t5, $zero, print
```

*Thực hiện gõ chương trình vào công cụ **MARS***

Giải thích:

- Hàm *main* khởi tạo một số thanh ghi chứa địa chỉ đầu và cuối của mảng
- Hàm *sort* dùng để sắp xếp mảng, sử dụng một số thanh ghi lưu trữ con trỏ đầu, cuối và con trỏ đến phần tử lớn nhất của mảng, giá trị phần tử lớn nhất của mảng
- Hàm *max* dùng để tìm phần tử lớn nhất của mảng, duyệt qua tất cả các phần tử của mảng và so sánh với phần tử lớn nhất hiện tại (lưu trữ con trỏ trong

v0 và giá trị trong v1). Sau đó hoán đổi với phần tử cuối cùng của mảng. Sau khi phần tử lớn nhất được hoán đổi với phần tử cuối cùng, con trỏ đến phần tử cuối giảm xuống 1 và tiếp tục lặp với mảng từ phần tử đầu đến phần tử cuối trừ 1 cho đến khi mảng được sắp xếp hoàn toàn

- Cuối cùng, in ra mảng được sắp xếp

```
-2 1 3 5 5 5 6 6 7 7 8 8 59
```

*Thực hiện chạy chương trình với **MARS***

Bài 3.

```
#Assignment 3
.data
        A: .word 7, -3, 5, 1, 5, 6, 7, 3, 6, 8, 8, 59, 5
        B: .asciiz " "
.text
        li $s0, 12      #n-1
        la $a0, A       #Address of A
        li $t0, 0       #i
        li $t1, 0       #j
for1:
        slt $t2, $t0, $s0       #i < n - 1
        beq $t2, $zero, end_for1
        li $t1, 0               #j=0
        sub $s1, $s0, $t0       #j = n - i - 1
for2:
        slt $t3, $t1, $s1       #j < n - i - 1
        beq $t3, $zero, end_for2

        mul $t4, $t1, 4
        add $t5, $t4, $a0       #A[j]
        lw $t6, 0($t5)
        lw $t7, 4($t5)
```

```
24          slt $t3, $t7, $t6       #if (arr[j] > arr[j + 1])
25          bne $t3, $zero, swap
26          addi $t1, $t1, 1        #j++
27          j for2
28  swap:
29          sw $t6, 4($t5)
30          sw $t7, 0($t5)
31          addi $t1, $t1, 1
32          j for2
33  end_for2:
34          addi $t0, $t0, 1
35          j for1
36  end_for1:
37          li $s0, 13
38          li $t0, 0
39          la $s1, A
```

```
39          la $s1, A
40  print:
41          add $t2, $t0, $t0
42          add $t2, $t2, $t2
43          add $t3, $t2, $s1
44          lw $t4, 0($t3)
45
46          li $v0, 1
47          move $a0, $t4
48          syscall
49
50          li $v0, 4
51          la $a0, B
52          syscall
53
54          addi $t0, $t0, 1
55          slt $t5, $t0, $s0
56          bne $t5, $zero, print
```

*Thực hiện gõ chương trình vào công cụ **MARS***

Giải thích:

- Chương trình được thực hiện trong 2 vòng lặp lồng nhau. Vòng *for1* duyệt qua từng phần tử của mảng, bắt đầu từ phần tử đầu tiên, kết thúc trước phần tử cuối cùng. Vòng *for2* duyệt qua các phần tử liên tiếp của mảng và so sánh chúng
- Hàm *for2* các phần tử liên tiếp được so sánh với nhau, nếu bên trái lớn hơn bên phải thì hoán đổi vị trí. Lặp đến khi vòng lặp *for2* kết thúc
- Cuối cùng in ra mảng đã được sắp xếp

```
-3 1 3 5 5 5 6 6 7 7 8 8 59
```

*Thực hiện chạy chương trình với **MARS***

Bài 4.

```
#Assignment 4
.data
        A:.word 7, -4, 5, 1, 5, 6, 7, 3, 6, 8, 8, 59
        B: .asciiz " "
.text
main:
        la $a0, A
        addi $a1, $a0, 0        #A[key-1]
        li $t0, 1               #i
max:
        addi $a1, $a1, 4        #A[key]
        lw $t2, 0($a1)          #key
        addi $t1, $t0, -1       #j = i - 1;
        addi $t3, $a1, 0        #A[j+1]
        addi $t5, $a1, -4       #A[j]
loop:
        slt $t4, $t1, $zero
        bne $t4, $0, end_loop

        lw $t6, 0($t5)          #value of A[j]
        slt $t4, $t2, $t6
        bne $t4, 1, end_loop

        sw $t6, 0($t3)          #A[j + 1] = A[j];
        addi $t3, $t3, -4
```

```
26          addi $t5, $t5, -4
27          addi $t1, $t1, -1       #j = j - 1;
28          j loop
29  end_loop:
30          sw $t2, 0($t3)          #arr[j + 1] = key;
31          addi $t0, $t0, 1
32          beq $t0, 13, end
33          j max
34  end:
35          li $s0, 13
36          li $t0, 0
37          la $s1, A
38  print:
39          add $t2, $t0, $t0
40          add $t2, $t2, $t2
41          add $t3, $t2, $s1
42          lw $t4, 0($t3)
43
44          li $v0, 1
45          move $a0, $t4
46          syscall
```

```
48          li $v0, 4
49          la $a0, B
50          syscall
51
52          addi $t0, $t0, 1
53          slt $t5, $t0, $s0
54          bne $t5, $zero, print
```

*Thực hiện gõ chương trình vào công cụ **MARS***

Giải thích: Chương trình sử dụng 2 vòng lặp

- Vòng lặp while (*max*) được sử dụng để tìm vị trí chính xác của phần tử hiện tại trong mảng A, sử dụng biến t5 để lưu trữ địa chỉ của A[j], t6 lưu trữ giá trị của nó. Nếu A[j] lớn hơn phần tử hiện tại thì di chuyển A[j] sang phải để tạo chỗ cho phần tử hiện tại. Kết thúc, phần tử hiện tại được chèn vào vị trí đúng bằng cách sao chép giá trị vào địa chỉ của A[j+1]
- Vòng lặp for (*loop*) tiếp tục đến khi tất cả phần tử được duyệt, kết thúc chuỗi được sắp xếp
- Cuối cùng in ra mảng đã được sắp xếp

```
-4 1 3 5 5 5 6 6 7 7 8 8 59
```

*Thực hiện chạy chương trình với **MARS***