BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH – TUẦN 6

Họ và tên: Nguyễn Mạnh Tùng

MSSV: 20225682

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:
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:
```

- Chạy chương trình và kiểm tra kết quả:

+ Vòng lặp thứ nhất:

$v0	2	0
$v1	3	0
$a0	4	268500992
$a1	5	5
$a2	6	0
$a3	7	0
$t0	8	1
$t1	9	-2

- $v0 và $v1 chưa được cập nhật kết quả
- $t0=1 (vị trí của phần tử thứ 2 trong mảng)
- $t1=-2 (tổng của phần tử đầu tiên với 0)

+ Vòng lặp thứ 2:

$v0	2	2
$v1	3	4
$a0	4	268500992
$a1	5	5
$a2	6	0
$a3	7	0
$t0	8	2
$t1	9	4

- Giá trị của $t0 và $t1 được cập nhật cho $v0 và $v1, thể hiện đang tính tổng của 2 phần tử đầu của mảng (-2 + 6 = 4)

+ Vòng lặp thứ 3:

$v0	2	2
$v1	3	4
$a0	4	268500992
$a1	5	5
$a2	6	0
$a3	7	0
$t0	8	3
$t1	9	3

- Gía trị của $t0 và $t1 thay đổi nhưng do 3<4 nên giá trị độ dài và tổng mới không được cập nhật cho $v0 và $v1

+ Vòng lặp thứ 4:

$v0	2	4
$v1	3	6
$a0	4	268500992
$a1	5	5
$a2	6	0
$a3	7	0
$t0	8	4
$t1	9	4

- Gía trị của $v0 và $v1 được cập nhật (vì tổng mới là 6>4)

+ Vòng lặp thứ 5:

$v0	2	4
$v1	3	6
$a0	4	268500992
$a1	5	5
$a2	6	0
$a3	7	0
$t0	8	5
$t1	9	4

- Kết thúc lặp, kết quả tổng mới bằng 5 < 6 nên không cập nhật giá trị cho $v0 và $v1
- $v0=4 (độ dài các phần tử có tổng max), $v1=6 (giá trị của tổng)

➔ Kết quả đúng với lí thuyết.

Assignment 2:

```
.data
A:      .word       3,2,4,1,5
Aend:       .word
size:   .word 5
msg: .asciiz " "
newl: .asciiz "\n"
.text
main:       la $a0,A      #$a0 = Address(A[0])
        la $a1,Aend
        lw $t6, size
        addi $a1,$a1,-4 #$a1 = Address(A[n-1])
        j sort          #sort
after_sort:   li $v0, 10 #exit
```

```
                syscall
end_main:
sort:   beq $a0,$a1,done        #single element list is sorted
        la $a0, A
        addi $t8, $zero, 0
        j print
        #j max                  #call the max procedure
after_print:  li $v0, 4
              la $a0, newl
              syscall
              la $a0, A
              j max
print: beq $t6, $t8, after_print
        add $t7, $zero, $a0
        lw $t0, ($a0)
        move $a0, $t0
        li $v0, 1
        syscall
        li $v0 4
        la $a0, msg
        syscall
        add $a0, $zero, $t7
        addi $a0, $a0, 4
```

```
        addi $t8, $t8, 1
        j print
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
max:
        addi $v0,$a0,0      #init max pointer to first element
        lw $v1,($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:

```
3 2 4 1 5
3 2 4 1 5
3 2 1 4 5
1 2 3 4 5

-- program is finished running --
```

➔ Kết quả đúng với lí thuyết

Assignment 3:

- Code:

```
.data
    A: .word  7, -2, 5, 1, 5, 3, 6, 59
    Aend: .word
.text
    la $a0, A           # $a0 = Địa chỉ(A[0])
    la $a1, Aend
    la $a2, Aend
    addi $a1, $a1, -4   # $a1 = Địa chỉ(A[n-1])
    move $a3, $a1
    j sort              # Gọi hàm sắp xếp
after_sort:
    li $v0, 10          # Gọi hàm thoát
    syscall
end_main:
```

```
sort:
        li $s0, 0                 # Khởi tạo biến đếm
        move $t7, $a1       # Địa chỉ phần tử cuối cùng của mảng vào $t7
loop:
        beq $t7, $a0, end_loop   # Trỏ đến pt cuối -> kết thúc
        add $t0, $s0, $s0    # Tính chỉ số của hai phần tử cần so sánh
        add $t0, $t0, $t0
        add $t1, $a0, $t0
        lw $t2, 0($t1)        # Load giá trị của hai phần tử
        addi $t3, $t1, 4
        lw $t4, 0($t3)
        slt $t5, $t2, $t4      # So sánh hai giá trị
        bne $t5, $zero, tang_$s0   # Nếu phần tử thứ nhất nhỏ hơn phần tử thứ
        hai, nhảy tới nhãn tang_$s0
        move $t6, $t4       # Hoán đổi hai giá trị
        move $t4, $t2
        move $t2, $t6
        sw $t2, 0($t1)
        sw $t4, 0($t3)
        addi $s0, $s0, 1    # Tăng biến đếm
        addi $t7, $t7, -4   # Giảm con trỏ
        j loop           # Lặp lại vòng lặp
tang_$s0:
```

```
        addi $s0, $s0, 1    # Tăng biến đếm
        addi $t7, $t7, -4   # Giảm con trỏ
        j loop              # Lặp lại vòng lặp
end_loop:
        addi $a3, $a3, -4   # Giảm con trỏ
#print
print:
        li $s1, 0           # Khởi tạo biến đếm
        la $s4, A           # Load địa chỉ của mảng A vào $s4
print_char:
        add $s2, $s1, $s1   # Tính chỉ số của phần tử cần in
        add $s2, $s2, $s2
        add $s3, $s4, $s2
        lw $s5, 0($s3)      # Load giá trị của phần tử cần in
        beq $s3, $a2, in_xuong_dong  # Nếu đến phần tử cuối cùng, in xuống
dòng
        li $v0, 1           # Gọi hàm in số
        move $a0, $s5
        syscall
        addi $s6, $a2, -4
        beq $s3, $s6, qua   # Nếu không phải phần tử cuối cùng, in khoảng trắng
        li $v0, 11          # Gọi hàm in ký tự
        li $a0, ' '
```

```
        syscall
qua:
        addi $s1, $s1, 1    # Tăng biến đếm
        j print_char       # Lặp lại quá trình in
in_xuong_dong:
        li $v0, 11         # Gọi hàm in ký tự
        li $a0, '\n'
        syscall
        la $a0, A          # Reset con trỏ về đầu mảng
beq $a3, $a0, after_sort  # Nếu con trỏ cuối cùng đến đầu mảng, kết thúc in
j sort                 # Ngược lại, quay lại sắp xếp và in
```

- Kết quả:

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

-- program is finished running --
```

➔ Kết quả đúng với lí thuyết

Assignment 4:

- Code:

```
.data
A: .word 7, -2, 5, 1, 5, 3, 6, 59
```

```
Aend: .word
.text
        la $a0, A               # $a0 = Địa chỉ của A[0]
        la $a1, Aend            # $a1 = Địa chỉ của phần tử cuối cùng của A
        add $a2, $a0, 4         # $a2 = Địa chỉ của A[1]
        la $a3, Aend            # giảm dần trong quá trình in
        add $a1, $a1, -4   # giảm dần trong quá trình sắp xếp
        j sort

after_sort:
        li $v0, 10
        syscall
end_main:

sort:
        li $s0, 1               # biến đếm
loop:
        li $s1, 0               # kiểm tra cho việc chèn phần tử
        add $t0, $s0, $s0  # Tính chỉ số của phần tử cần chèn
        add $t0, $t0, $t0
        add $t0, $a0, $t0  # Địa chỉ của phần tử cần chèn
        addi $t2, $t0, -4    # Địa chỉ của phần tử trước phần tử cần chèn
        move $t5, $t0           # Sao chép địa chỉ của phần tử cần chèn
```

```
loop1:
        lw $t1, 0($t0)              # Load giá trị của phần tử cần chèn
        lw $t3, 0($t2)              # Load giá trị của phần tử trước phần tử cần chèn
        slt $t4, $t1, $t3      # So sánh giá trị
        beq $t4, $zero, tru_tiep# Nếu giá trị phía trước nhỏ hơn, chuyển sang bước tiếp theo
        move $t5, $t2               # Lưu lại địa chỉ của phần tử trước phần tử cần chèn
        li $s1, 1                   # Đánh dấu có phần tử cần chèn
tru_tiep:
        beq $t2, $a0, chen          # Nếu đã đến đầu mảng, chèn phần tử vào đây
        addi $t2, $t2, -4     # Di chuyển về phần tử trước đó
        j loop1
chen:
        beq $s1, 0, print    # Nếu không có phần tử cần chèn, chuyển sang in mảng
loop_chen:
        addi $t0, $t0, -4     # Di chuyển về phần tử cần chèn
        lw $s2, 0($t0)              # Lưu giá trị của phần tử cần chèn
        sw $s2, 4($t0)              # Dịch phải giá trị của các phần tử
        bne $t0, $t5, loop_chen # Lặp lại quá trình cho tới khi đến vị trí cần chèn
        sw $t1, 0($t5)              # Chèn phần tử vào vị trí cần chèn
```

```
#print
print:
        li $s4, 0               # Khởi tạo biến đếm
print_char:
        la $a0, A               # Load địa chỉ của mảng
        add $s5, $s4, $s4 # Tính chỉ số của phần tử cần in
        add $s5, $s5, $s5
        add $s5, $a0, $s5
        lw $s6, 0($s5)                  # Load giá trị của phần tử cần in
        beq $s5, $a3, in_xuong_dong# Nếu đến cuối mảng, in xuống dòng
        li $v0, 1               # Gọi hàm in số
        move $a0, $s6
        syscall
        addi $s7, $a3, -4
        beq $s5, $s7, skip# Nếu không phải phần tử cuối cùng, in một khoảng
trắng
        li $v0, 11              # Gọi hàm in ký tự
        li $a0, ' '
        syscall
 skip:
        addi $s4, $s4, 1    # Tăng biến đếm
        j print_char            # Lặp lại quá trình in
 in_xuong_dong:
```

```
        li $v0, 11
        syscall
        la $a0, A               # Reset con trỏ về đầu mảng
beq $a1, $a2, after_sort # Nếu đã in hết mảng, kết thúc
addi $s0, $s0, 1            # Tăng biến đếm
addi $a2, $a2, 4            # Di chuyển con trỏ
j loop                      # Quay lại vòng lặp
```

- Kết quả:

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

-- program is finished running --
```

➔ Kết quả đúng với lí thuyết