Assignment 1

Mã nguồn:

```
.data
A: .word 2, 0, -1, 9, -4, 4, -8, 4

.text
main: la $a0,A
 li $a1,5


#------------------------------------------------------------------


#Procedure mspfx
# @brief find the maximum-sum prefix in a list of integers
# @param[in] a0 the base address of this list(A) need to be processed
# @param[in] a1 the number of elements in list(A)
# @param[out] v0 the length of sub-array of A in which max sum reachs.
# @param[out] v1 the max sum of a certain sub-array
#------------------------------------------------------------------
#Procedure mspfx
#function: find the maximum-sum prefix in a list of integers
#the base address of this list(A) in $a0 and the number of
#elements is stored in a1
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
```

Lưu ý: Vì đây là một chương trình nhỏ được cắt ra từ một chương trình lớn, nên sẽ có một số vòng lặp phục vụ cho việc nhận tín hiệu từ bàn phím. Ta cần phải xoá các dòng code

```
 j mspfx
 nop
continue:
lock: j lock
 nop
end_of_main:
```

và

```
done: j continue
mspfx_end:
```

để tránh vòng lặp vô hạn.

Giải thích:

- Khởi tạo giá trị:

```
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
```

Các câu lệnh này thực hiện khởi tạo giá trị cho các biến. $v0 là độ dài của prefix có tổng lớn nhất. $v1 là tổng của prefix lớn nhất. $t0 là giá trị của biến đếm i. $t1 chứa giá trị tổng của prefix hiện tại.

- Vòng lặp:

```
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?
```

Các câu lệnh này là thân của vòng lặp. 4 câu lệnh đầu thực hiện gán giá trị A[i] cho thanh ghi $t4. Câu lệnh `add $t1,$t1,$t4` thực hiện cộng A[i] cho tổng prefix hiện tại. Câu lệnh `slt $t5,$v1,$t1` và `bne $t5,$zero,mdfy` thực hiện rẽ đến nhánh mdfy nếu *max tổng prefix* $<$ *tổng prefix hiện tại*. Nếu không thì *jump* đến nhánh test.

- Nhánh mdfy:

```
mdfy: addi $v0,$t0,1 #new max-sum prefix has length i+1
addi $v1,$t1,0 #new max sum is the running sum
```

Câu lệnh `addi $v0,$t0,1` thực hiện cập nhật giá trị cho độ dài của max prefix mới. Câu lệnh `addi $v1,$t1,0` cập nhật giá trị *max tổng prefix* $=$ *tổng prefix hiện tại*.

- Nhánh test:

```
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
```

3 câu lệnh này thực hiện cập nhật giá trị cho biến đếm i, nếu $i<n$ thì quay lại nhãn *loop*. Nếu không thì kết thúc chương trình.

Assignment 2

Mã nguồn:

```
.data
A: .word 7, -2, 5, 1, 5,6,7,3,6,8,8,59,5
Aend: .word

.text
main: la $a0,A #$a0 = Address(A[0])
 la $a1,Aend
 addi $a1,$a1,-4 #$a1 = Address(A[n-1])
 j sort #sort
after_sort: li $v0, 10 #exit
 syscall
end_main:
#------------------------------------------------------------
#procedure sort (ascending selection sort using pointer)
#register usage in sort program
#$a0 pointer to the first element in unsorted part
#$a1 pointer to the last element in unsorted part
#$t0 temporary place for value of last element
#$v0 pointer to max element in unsorted part
#$v1 value of max element in unsorted part
#------------------------------------------------------------
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 after_sort
#Procedure max
#function: fax the value and address of max element in the list
#$a0 pointer to first element
#$a1 pointer to last element
#------------------------------------------------------------------------
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,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
```

Lưu ý: 2 mảng A và Aend được khởi tạo liên tiếp nhau, vậy nên địa chỉ của con trỏ Aend cũng nằm ngay sau địa chỉ con trỏ của A[n]. Vậy nếu ta muốn trỏ đến phần tử cuối cùng của mảng A, ta chỉ cần câu lệnh `addi $a1,$a1,-4` là $a1 sẽ trỏ đến phần tử A[n].

Giải thích:

- Bắt đầu chương trình:

```
main: la $a0,A #$a0 = Address(A[0])
 la $a1,Aend
 addi $a1,$a1,-4 #$a1 = Address(A[n-1])
 j sort #sort
```

  Load địa chỉ các mảng vào các thanh ghi. Thanh ghi $a0 chứa địa chỉ của A[i], thanh ghi $a1 chứa địa chỉ của A[n]. Sau đó chương trình nhảy đến nhánh *sort*.

- Nhánh *sort*:

```
sort: beq $a0,$a1,done #single element list is sorted
 j max #call the max procedure
```

  Lệnh `beq $a0,$a1,done` sẽ nhảy đến nhánh done nếu chương trình đã duyệt qua hết tất cả phần tử trong mảng. (Địa chỉ thanh ghi $a0 = $a1). Nếu không thì nhảy đến nhánh max.

- Nhánh *max* và *loop*:

```
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,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
```

  - Nhánh *max* khởi tạo các giá trị ban đầu: thanh ghi $v0 chứa địa chỉ của phần tử có giá trị lớn nhất của dãy con chưa được sắp xếp, thanh ghi. $v1 chứa giá trị của phần tử lớn nhất của dãy con chưa được sắp xếp, thanh ghi $t0 chứa địa chỉ của phần tử tiếp theo.
  - Nhánh *loop*:
    * Câu lệnh `beq $t0,$a1,ret` thực hiện rẽ đến nhánh *ret* nếu đã duyệt đến phần tử cuối cùng của dãy con chưa được sắp xếp.
    * 2 câu lệnh `addi` và `lw` thực hiện gán giá trị phần tử tiếp theo cho thanh ghi $t1. (*Cách duyệt đến phần tử kế tiếp này là khác so với cách duyệt ở Assignment 1*).
    * 2 câu lệnh `slt $t2,$t1,$v1` và `bne $t2,$zero,loop` sẽ thực hiện quay lại nhánh loop này nếu *phần tử hiện tại* $\leq$ *max*.
    * Nếu *phần tử hiện tại* $>$ *max* thì thực hiện câu lệnh `addi $v0,$t0,0` (gán *phần tử hiện tại* = *phần tử max*) và `addi $v1,$t1,0`