

Báo cáo thực hành KTMT

Lê Quốc Đăng

20225801

Assignment 1

- **Code**

```
.data
    A: .word -2, 6, -1, 3, -2
    message1: .asciiz "Tong day prefix lon nhat co gia tri la: "
    message2: .asciiz "\n"
    message3: .asciiz "Do dai chuoai: "

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

continue:
    add $t6,$v0,$zero #store value of $v0 into $t6
    la $a0, message1 #print message1
    li $v0,4
    syscall
    add $a0,$v1,$zero #print $v1 (max sum)
    li $v0,1
    syscall
    la $a0, message2 #print message2
    li $v0,4
    syscall

    la $a0, message3 #print message3
```

```

    li $v0,4
    syscall
    add $a0,$t6,$zero print $t6 (length)
    li $v0,1
    syscall
    j end_of_main#-----
-----

#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 reaches.
# @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 $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
done:
        j continue
mspfx_end:
end_of_main:

```

• Giải thích

- Tổng quan: thuật toán để tìm ra tổng dãy tiền tố lớn nhất. Khởi tạo max_sum = 0, tạo vòng lặp để tính cur_sum. Nếu max_sum < cur_sum thì cập nhật max_sum. Sau đó đưa ra độ dài và tổng max đó.

- Chi tiết:

Mspfx: Khởi tạo các giá trị ban đầu bằng 0

- \$v0: độ dài dãy tổng tiền tố max
- \$v1: giá trị tổng max của dãy tổng tiền tố
- \$t0: chỉ số của mảng
- \$t1: tổng dãy tiền tố đang xét

Loop: vòng lặp duyệt mảng

```

add $t2,$t0,$t0
add $t2,$t2,$t2
add $t3,$t2,$a0

```

- Lấy địa chỉ của phần tử thứ i bằng phương pháp indexing (lấy địa chỉ của mảng – A[0] + 4*i). Vì 1 word có 4 byte nên phải nhân 4.

```
lw $t4,0($t3)
```

- Lấy giá trị A[i]

```

add $t1,$t1,$t4

slt $t5,$v1,$t1

bne $t5,$zero,mdfy

```

- Cộng vào tổng dãy tiền tố đang xét (\$t1), Kiểm tra xem \$t1 có lớn hơn tổng tiền tố max (\$v0). Nếu tối ưu hơn thì nhảy để thay đổi giá trị tối ưu mới (\$v0)

Test:

```

addi $t0,$t0,1

slt $t5,$t0,$a1

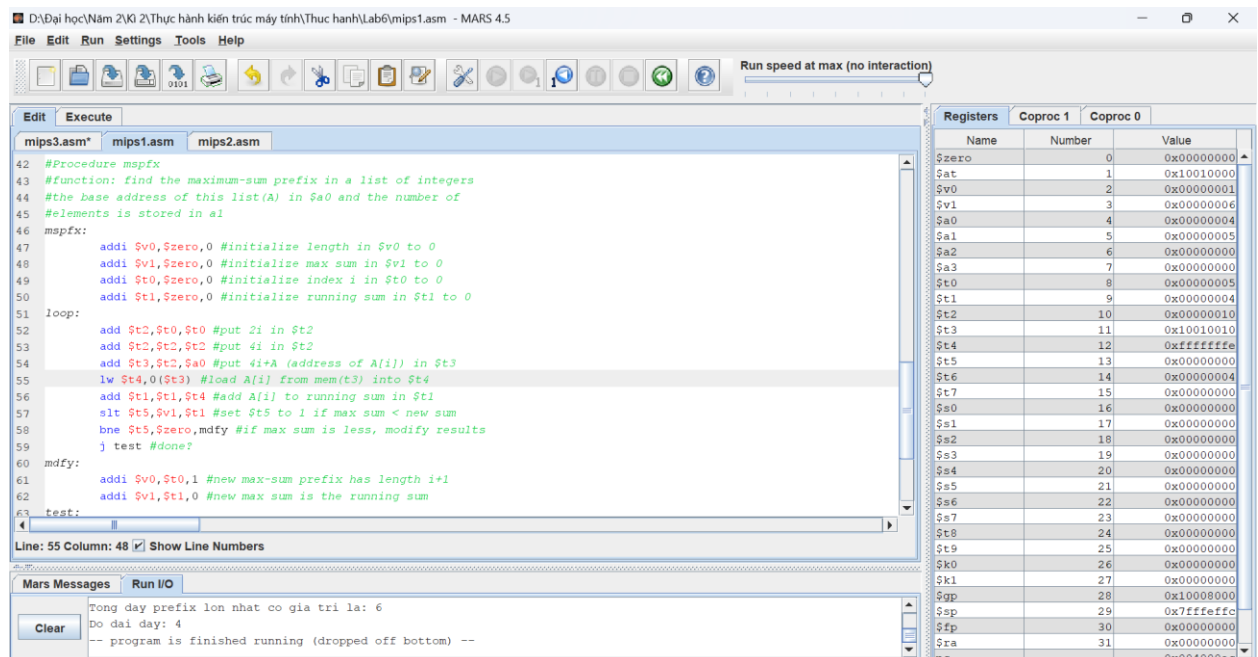
bne $t5,$zero,loop

```

- Tăng i lên 1 đơn vị và kiểm tra điều kiện kết thúc vòng lặp. Nếu $i < n$ quay trở lại loop.

⇒ Kết thúc vòng lặp in ra tổng tiền tố lớn nhất và độ dài của nó

• Kết quả



⇒ Kết quả đúng

Assignment 2

- Code

.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

```

- **Giải thích**

- Tổng quan:

Ý tưởng của selection sort:

1. Xác định phần tử nhỏ nhất (hoặc lớn nhất) trong danh sách.
2. Hoán đổi phần tử nhỏ nhất (hoặc lớn nhất) với phần tử cuối cùng của danh sách.
3. Tiếp tục sắp xếp danh sách còn lại (trừ phần tử đã sắp xếp) bằng cách lặp lại bước 1 và 2 cho đến khi danh sách được sắp xếp hoàn toàn.

- Chi tiết

Khởi tạo:

- \$a0 trỏ đến phần tử đầu tiên của mảng
- \$a1 trỏ đến phần tử cuối cùng chưa được sắp xếp của mảng
- \$t0 chứa giá trị của phần tử cuối cùng chưa được sắp xếp
- \$v0 trỏ đến phần tử có giá trị lớn nhất của mảng chưa được sắp xếp
- \$v1 chứa giá trị của phần tử có giá trị lớn nhất của mảng chưa được sắp xếp

sort:

```
beq $a0,$a1,done
```

```
j max
```

- Kiểm tra xem nếu mảng chưa được sắp xếp còn 1 phần tử thì kết thúc chương trình. Không thì gọi đến max

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
```

- Hoán đổi giá trị của \$v0 với \$a1. Bỏ qua phần tử đã được sắp xếp bằng cách \$a1 = \$a1 - 4. Tiếp tục sắp xếp với mảng chưa được sắp xếp.

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
```

- Khởi tạo giá trị max = A[0] và thực hiện vòng lặp

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
```

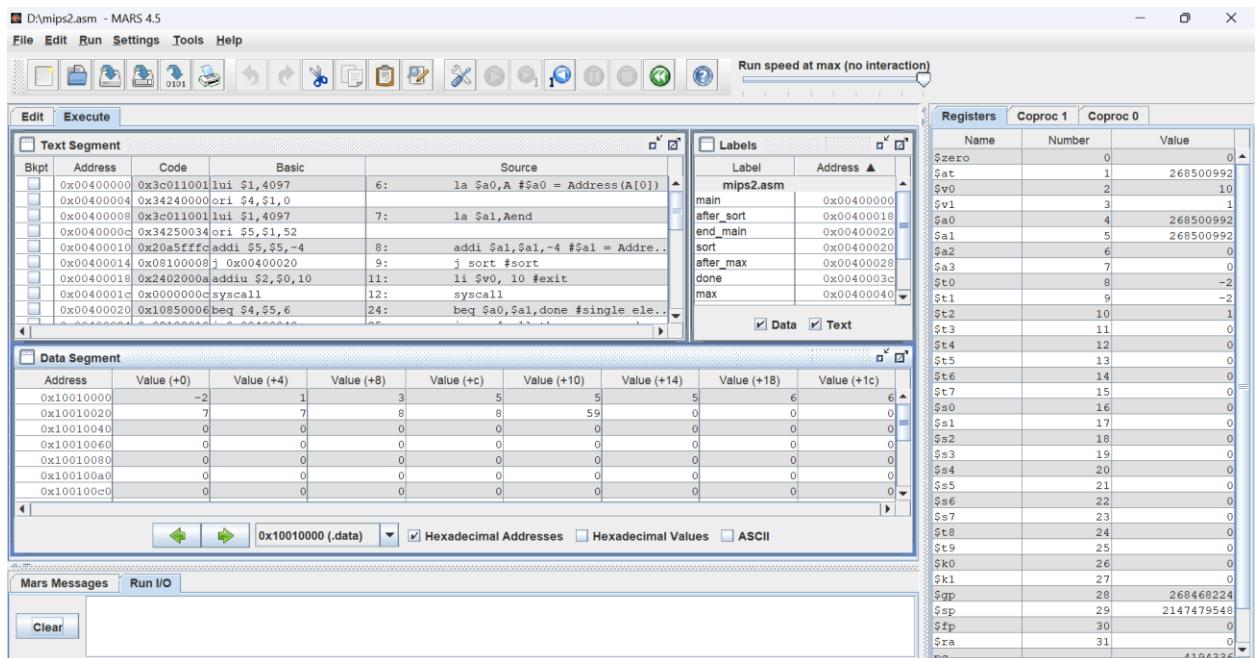
```
slt $t2,$t1,$v1 #(next)<(max) ?
```

```

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

```

- Kiểm tra nếu \$t0 trở đến phần tử cuối cùng của mảng chưa được sắp xếp thì gọi đến ret.
- Sử dụng pointer updating method: trở đến phần tử tiếp theo bằng cách \$t0 = \$t0 + 4, bởi 1 word chiếm 4 byte nên cộng 4 để trở tới ô nhớ tiếp theo.
- So sánh phần tử hiện tại với phần tử max. Nếu lớn hơn thì cập nhật, không thì tiếp tục vòng lặp.
- **Kết quả**



- ⇒ Dãy được sắp xếp theo thứ tự tăng dần
- ⇒ Kết quả đúng

Assignment 3

- **Ý tưởng của bubble sort như sau:**
 1. Bắt đầu từ đầu danh sách, so sánh phần tử thứ i với phần tử thứ i+1.
 2. Nếu phần tử thứ i lớn hơn (hoặc nhỏ hơn) phần tử thứ i+1, hoán đổi chúng.
 3. Tiếp tục lặp lại bước 1 và 2 cho đến khi đi qua tất cả các phần tử trong danh sách.
 4. Lặp lại quá trình trên cho đến khi không có phần tử nào được hoán đổi nữa.
- **Code**

.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-2])
        j sort #sort
after_sort:
        li $v0, 10 #exit
        syscall
end_main:
#-----
#procedure sort (ascending bubble sort using pointer)
#register usage in sort program
#$a0 trỏ đến phần tử đầu tiên của mảng
#$a1 trỏ đến phần tử cuối cùng của mảng
#$t0 check = 1 nếu có phần tử được hoán đổi, = 0 nếu ko có phần tử nào
được hoán đổi
#$t1 trỏ đến A[i]
#$t2 trỏ đến A[i+1]
#$v0 = A[i]
#$v1 = A[i+1]
#-----
sort:
        add $t1,$a0,$zero #$t1 trỏ đến A[0]
        add $t0,$zero,$zero #$t0 = 0
        j loop
check:
        beq $t0,$zero,done #nếu ko có phần tử nào được hoán đổi thì nhảy
đến done

```

```

        j reset #tiếp tu vong lap moi
reset:
    add $t0,$zero,$zero
    add $t1,$a0,$zero
    j loop
swap:
    sw $v0,0($t2)
    sw $v1,0($t1)
    li $t0,1
    j next_loop #tiếp tục vòng lặp với phần tử tiếp theo
next_loop:
    addi $t1,$t1,4
    j loop
loop:
    beq $t1,$a1,check
    lw $v0,0($t1) #$v0 = A[i]
    addi $t2,$t1,4
    lw $v1,0($t2) #$v0 = A[i+1]
    sgt $t5,$v0,$v1 #$v0>$v1
    bne $t5,$zero,swap #Nếu $v0>$v1 thì swap
    j next_loop #tiếp tục vòng lặp với phần tử tiếp theo
done:
    j after_sort

```

- **Kết quả**

D:\Đại học\Năm 2\Kì 2\Thực hành kiến trúc máy tính\Thực hành\Lab6\mips3.asm - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c011001	lui \$1,4097	6: la \$a0,A#\$a0 = Address(A{0})
	0x00400004	0x34240000	ori \$4,\$1,0	
	0x00400008	0x3c011001	lui \$1,4097	7: la \$a1,Aend
	0x0040000c	0x34250034	ori \$5,\$1,52	
	0x00400010	0x20a5fffc	addi \$5,\$5,-4	8: addi \$a1,\$a1,-4#\$a1 = Address(A{0})
	0x00400014	0x08100008	j sort	9: j sort \$sort
	0x00400018	0x2402000a	addiu \$2,\$0,10	11: li \$v0,10 \$exit
	0x0040001c	0x0000000c	syscall	12: syscall
	0x00400020	0x00804820	add \$9,\$4,\$0	26: add \$t1,\$a0,\$zero \$t1 trở d...

Labels

Label	Address
main	0x00400000
after_sort	0x00400018
end_main	0x00400020
sort	0x00400020
check	0x0040002c
reset	0x00400034
swap	0x00400040

☒ 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

0x10010000 (.data) ☒ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

Registers

Name	Number	Value
\$zero	0	0
\$at	1	268500992
\$v0	2	10
\$v1	3	59
\$a0	4	268500992
\$a1	5	268501040
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	268501040
\$t2	10	268501040
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0

Mars Messages

Run I/O

Tong day prefix lon nhat co gia tri la: 6
Do dai day: 4
-- program is finished running (dropped off bottom) --

Clear

Dãy được sắp xếp theo thứ tự tăng dần

⇒ Kết quả đúng (Giải thích chi tiết trong code)