

Bài 1

.data

A: .word -2, 7, -1, 3, -2, 6, -3

.text

main:

la \$a0,A

li \$a1,7

j mspfx

nop

continue:

lock:

j lock

nop

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:

```

Xét với ví dụ mảng có 7 phần tử lần lượt là -2, 7, -1, 3, -2, 6, -3

Vì mảng nhập vào biết trước có 7 phần tử nên ở main: ta phải li \$a1, 7 để gán số lượng phần tử tối đa của mảng để có điểm kết thúc được viết ở test:

Tổng tạm thời của prefix được lưu ở \$t1, phần tử đang được xét được lưu ở \$t0

Tổng max prefix được lưu ở \$v1 và prefix đó kết thúc ở phần tử được lưu ở \$v0

Ví dụ đang ở phần tử thứ 2 là 7

Registers	Coproc 1	Coproc 0
Name	Number	Value
\$zero	0	0
\$at	1	268500992
\$v0	2	2
\$v1	3	5
\$a0	4	268500992
\$a1	5	7
\$a2	6	0
\$a3	7	0
\$t0	8	1
\$t1	9	5
\$t2	10	4
\$t3	11	268500996
\$t4	12	7
\$t5	13	1
\$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
pc		4194388
hi		0
lo		0

\$t4 lưu giá trị vừa đọc được

\$v1 hiện tại đang lưu max-prefix hiện tại là 5 và được tính tới phần tử thứ 2

\$a3 lưu địa chỉ của phần tử đó

Kết quả cuối cùng là :

Registers	Coproc 1	Coproc 0	
Name	Number	Value	
\$zero	0	0	
\$at	1	268500992	
\$v0	2	6	
\$v1	3	11	
\$a0	4	268500992	
\$a1	5	7	
\$a2	6	0	
\$a3	7	0	
\$t0	8	7	
\$t1	9	8	
\$t2	10	24	
\$t3	11	268501016	
\$t4	12	-3	
\$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	
pc		4194324	
hi		0	
lo		0	

\$t4 đang lưu giá trị -3 là giá trị cuối cùng của mảng

\$v1 lưu giá trị max prefix cuối cùng là 11 được tính tới phần tử thứ 6

→ Vậy kết quả thực tiễn đúng với lý thuyết cần tính

Bài 2

.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: find 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

```

xét với chuỗi cơ bản bài cho là 7, -2, 5, 1, 5,6,7,3,6,8,8,59,5

Data Segment									
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)	
0x10010000	7	-2	5	1	5	6	7	3	▲
0x10010020	6	8	8	59	5	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	
0x100100e0	0	0	0	0	0	0	0	0	
0x10010100	0	0	0	0	0	0	0	0	
0x10010120	0	0	0	0	0	0	0	0	▼

Chuỗi ban đầu chưa được sắp xếp có dạng như trên ảnh

Sau lượt chạy đầu tiên

The screenshot shows a debugger interface with two main panels. The top panel displays assembly code with columns for Address, Code, Basic, and Source. Instruction 26, `addi $a1,$a1,-4`, is highlighted in yellow. The bottom panel shows the Data Segment with columns for Address and various offset values (+0, +4, +8, +c, +10, +14, +18, +1c). The value 59 is visible at address 0x10010020. On the right, the Registers window shows the state of various registers, with `$v1` containing the value 59.

Ta thấy được 59 là số lớn nhất nên được cất vào \$v1 để swap với số cuối cùng

Sau lượt đó ta thấy số 59 và phần tử 5 cuối cùng của mảng được swap cho nhau và bắt đầu chạy 1 vòng mới

Kết quả cuối cùng:

Sau đúng số phần tử vòng chạy thì ta được kết quả mảng sau khi đã sắp xếp thu được là :

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
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

→ Vậy kết quả đã đúng với lý thuyết của selection sort:

Chọn số có giá trị lớn nhất của mảng và đổi chỗ nó với giá trị cuối cùng chưa được sắp xếp và có độ phức tạp của thuật toán là $O(n^2)$

Bài 3

.data

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

Aend: .word

space: .ascii " "

.text

main:

```
la $s0, A
li $t0, 0          #i = 0
li $t1, 0          #j = 0
li $s1, 13         #n = 13 ( số phần tử)
li $s2, 13         #n-i for inner loop
add $t2, $zero, $s0    # addr by i
add $t3, $zero, $s0    # addr by j
```

```
addi $s1, $s1, -1
```

outer_loop:

```
li $t1, 0          #j = 0
addi $s2, $s2, -1  #n-1
add $t3, $zero, $s0    #reset addr 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

```

Kết quả trả về :

```

-- program is finished running --
-2 1 3 5 5 5 6 6 7 7 8 8 59

```

➔ đúng với lý thuyết