

**BÁO CÁO THỰC HÀNH**  
**KIẾN TRÚC MÁY TÍNH – TUẦN 6**  
**Họ và tên: Hà Trung Chiến**  
**MSSV: 20225794**

**Assignment 1:**

```
.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:
#-----
#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:

**Kết quả:**

The screenshot shows the Mars MIPS simulator interface. The 'Text Segment' window displays assembly code with addresses from 0x00400000 to 0x00400028. The 'Labels' window shows labels like 'main', 'continue', 'lock', 'end\_of\_main', 'mspfx', 'loop', 'mdfy', 'test', 'done', and 'mspfx\_end'. The 'Data Segment' window shows memory values at addresses 0x10010000 to 0x10010120. The 'Registers' window on the right shows the state of registers \$zero through \$t10.

Name	Number	Value
\$zero	0	0
\$at	1	268500992
\$v0	2	0
\$v1	3	6
\$a0	4	268500992
\$a1	5	5
\$a2	6	0
\$a3	7	0
\$t0	8	5
\$t1	9	4
\$t2	10	16
\$t3	11	268501008
\$t4	12	-2
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$t8	16	0
\$t9	17	0
\$t10	18	0
\$t11	19	0
\$t12	20	0
\$t13	21	0
\$t14	22	0
\$t15	23	0
\$t16	24	0
\$t17	25	0
\$t18	26	0
\$t19	27	0
\$t20	28	268468224
\$t21	29	2147479548
\$t22	30	0
\$t23	31	0
\$f0		4194324
\$f1		0
\$f2		0
\$f3		0
\$f4		0
\$f5		0
\$f6		0
\$f7		0
\$f8		0
\$f9		0
\$f10		0
\$f11		0
\$f12		0
\$f13		0
\$f14		0
\$f15		0
\$f16		0
\$f17		0
\$f18		0
\$f19		0
\$f20		0
\$f21		0
\$f22		0
\$f23		0
\$f24		0
\$f25		0
\$f26		0
\$f27		0
\$f28		0
\$f29		0
\$f30		0
\$f31		0

→ Tổng lớn nhất là 6 tương ứng với \$v1 và độ dài mảng đến khi có tổng lớn nhất là 4 ứng với \$v0

**Giải thích:**

Khi chạy chương trình giá các giá trị của mảng A sẽ lần lượt được gán vào thanh ghi \$t4, tiếp theo thanh ghi \$t1 sẽ lưu tổng hiện tại của các phần tử mảng “add \$t1,\$t1,\$t4”. Sau đó chúng ta thực hiện so sánh giá trị của thanh ghi \$v1 (lưu tổng lớn nhất của mảng tại thời điểm đó) với \$t1.

- + Nếu  $v1 < t1$  → giá trị  $t5 = 1$  thực hiện “mdfy” lưu giá trị của \$t1 vào \$v1 và tăng \$v0 lên một đơn vị.
  - + Nếu sai →  $t5 = 0$  thực hiện “test” chuyển sang phần tử kế tiếp và đưa giá trị  $t5 = 1$ .
- Cứ thực hiện như vậy cho đến hết mảng ta thu được kết quả đúng như lý thuyết.

## Assignment 2:

#Laboratory Exercise 6, Assignment 2

.data

space: .asciiz " "

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

Aend: .word

length: .word 13

.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:

```

        la $a3, A                # load the address of array into $a0
        lw $a1, length           # load the length of the array into $a1
print_loop:
        lw $t0, 0($a3)           # load the current element into $t0
        li $v0, 1                # print the current element
        move $a0, $t0
        syscall
        li $v0, 4                # print a space
        la $a0, space
        syscall
        addiu $a3, $a3, 4        # move to the next element
        addiu $a1, $a1, -1       # decrement the length
        bgtz $a1, print_loop     # if length > 0, repeat the loop
        li $v0, 10              # exit
        syscall
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 after_sort
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), repeat
        bne $t2, $zero, loop     # next element is new max element
        addi $v0, $t0, 0         # next value is new max value
        addi $v1, $t1, 0         # change completed; now repeat
        j loop
ret:
        j after_max

```

### Kết quả:

Mảng trước khi sắp xếp: 7, -2, 5, 1, 5, 6, 7, 3, 6, 8, 8, 59, 5

Mảng sau khi sắp xếp: -2, 1, 3, 5, 5, 5, 6, 6, 7, 7, 8, 8, 59

Text Segment				Labels
Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c011001	lui \$1,4097	5: main: la \$a0,A # \$a0 = Address(A[0])
	0x00400004	0x34240000	ori \$4,\$1,0	
	0x00400008	0x3c011001	lui \$1,4097	6: la \$a1,Aend
	0x0040000c	0x34250034	ori \$5,\$1,52	
	0x00400010	0x20a5ffff	addi \$5,\$5,-4	7: addi \$a1,\$a1,-4 # \$a1 = Address(A[n-1])
	0x00400014	0x08100008	j 0x00400020	8: j sort \$sort
	0x00400018	0x2402000a	addiu \$2,\$0,10	9: after_sort: li \$v0, 10 #exit
	0x0040001c	0x0000000c	syscall	10: syscall
	0x00400020	0x10850006	beq \$4,\$5,6	21: sort: beq \$a0,\$a1,done #single element list is..
	0x00400024	0x08100010	j 0x00400040	22: j max #call the ma..
	0x00400028	0x8ca80000	lw \$8,0(\$5)	23: after_max: lw \$t0,0(\$a1) #load last element into..
	0x0040002c	0xac480000	sw \$8,0(\$2)	24: sw \$t0,0(\$v0) #copy last e..

  

Label	Address
mips1.asm	
main	0x00400000
after_sort	0x00400018
end_main	0x00400020
sort	0x00400020
after_max	0x00400028
done	0x0040003c
max	0x00400040
loop	0x0040004c
ret	0x0040006c
A	0x10010000

  

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

### Giải thích:

- Gán địa phần tử thứ n-1 vào thanh ghi \$a1 và địa chỉ phần tử đầu tiên vào thanh ghi \$a0.
- Lần lượt duyệt qua các phần tử trong mảng tìm giá trị lớn nhất trong mảng tại thời điểm đó lưu vào \$v1 sau khi tìm được giá trị đó đổi vị trí giá trị đây với vị trí cuối cùng rồi dịch con trỏ xuống vị trí thấp hơn để làm vị trí cuối cùng mới “addi \$a1,\$a1,-4” tiếp tục thực hiện lại như trên ta sẽ thu được mảng mới đã được sắp xếp.

### Assignment 3:

#Laboratory Exercise 6, Assignment 3

.data

space: .asciiz " "

array: .word 5, -6, -2, 7, 1, 10, 9, 4, -3, 2

length: .word 10 # Length of the array

.text

sort:

la \$a0, array

# Load the address of the array into \$a0

lw \$s0, length

# Load the length of the array into \$a1

li \$t0,0 #i=0

out\_loop:

beq \$t0, \$s0, end\_sort

# end program

addi \$t1, \$0, 0

# y=0

addi \$t0, \$t0, 1

# i++

in\_loop:

sub \$t8, \$s0, \$t0

slt \$t9, \$t1, \$t8

beq \$t9, \$zero, out\_loop

sll \$t2, \$t1, 2 #t2 = 4\*y

add \$t3, \$t2, \$a0 #t3 = &A[y]

lw \$t4, 0(\$t3) #t4 = A[y]

addi \$t7, \$t3, 4 #t7 = &A[y+1]

lw \$t5, 0(\$t7) #t5 = A[y+1]

slt \$t6, \$t4, \$t5 #compare A[y] with A[y+1]

```

        bne $t6,$zero, next
        sw $t4,0($t7)          #swap A[y] and A[y+1]
        sw $t5,0($t3)

next:
        addi $t1,$t1,1          #y++
        j in_loop

end_sort:
print:
        la $a3, array           # Load the address of the array into $a3
        lw $a1, length           # Load the length of the array into $a1
        # Loop to print each element of the array
loop:
        lw $t0, 0($a3)          # Load the current element into $t0
        # Print the current element
        li $v0, 1
        move $a0, $t0
        syscall
        # Print a space
        li $v0, 4
        la $a0, space
        syscall
        addiu $a3, $a3, 4        # Move to the next element
        addiu $a1, $a1, -1       # Decrement the length
        sle $s7, $a1, $zero     # If length > 0, repeat the loop
        beq $s7, $zero, loop
        li $v0, 10
        syscall

```

### Kết quả:

Mảng trước khi sắp xếp: 5, -6, -2, 7, 1, 10, 9, 4, -3, 2

Mảng sau khi sắp xếp: -6, -3, -2, 1, 2, 4, 5, 7, 9, 10

The screenshot displays a MIPS development environment with three main panels:

- Text Segment:** A table of assembly instructions with columns for Bkpt, Address, Code, Basic, and Source. The code implements a sorting algorithm, including array loading, element printing, and loop control.
- Labels:** A table mapping labels to memory addresses. Labels include DemPhanTu, Size, loop1, loop2, if\_swap, swap, Exit, A, and Aend.
- Data Segment:** A table showing memory addresses and their values. The values are displayed in decimal and hexadecimal formats. The data shows the state of memory after the sorting process.

