

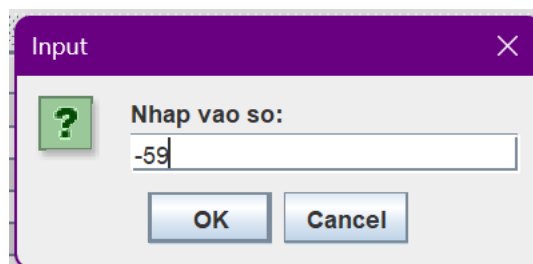
Báo cáo thực hành KTMT Tuần 7

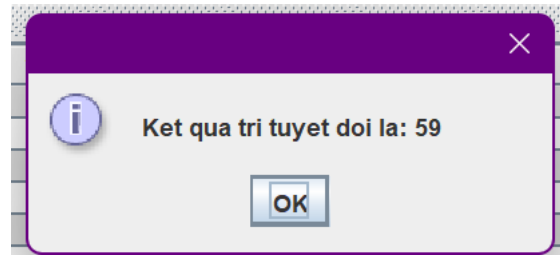
Phạm Thành Lập – 20215076

Assignment 1

```
#Laboratory Exercise 7 Home Assignment 1
.data
Message: .asciiz "Nhap vao so:"
Message1: .asciiz "Ket qua tri tuyet doi la: "
.text
main:  li $v0, 51
      la $a0, Message
      syscall
      #li $a0, -45 #load input parameter
      jal abs #jump and link to abs procedure
      nop
      add $s0, $zero, $v0
      li $v0, 56
      la $a0, Message1
      la $a1, 0($s0)
      syscall
      li $v0, 10 #terminate
      syscall

endmain:
#-----
# function abs
# param[in] $a0 the interger need to be gained the absolute value
# return $v0 absolute value
#-----
abs:
      sub $v0,$zero,$a0 #put -(a0) in v0; in case (a0)<0
      bltz $a0,done #if (a0)<0 then done
      nop
      add $v0,$a0,$zero #else put (a0) in v0
done:
      jr $ra
```





- Chương trình truyền vào tham số đầu vào lưu -45 vào thanh ghi \$a0 lệnh jal sẽ nhảy đến địa chỉ của thẻ cần nhảy đến đồng thời lưu địa chỉ của câu lệnh ngay sau câu lệnh jal vào thanh ghi \$ra
- Sau khi nhảy đến thẻ abs chương trình thực hiện lưu giá trị -(a0) vào thanh ghi \$v0 sau đó so sánh \$a0 với 0 nếu giá trị thanh ghi \$a0 nhỏ hơn 0 thì \$v0 không đổi nhảy đến thẻ done sau đó thực hiện lệnh jr nhảy địa chỉ mà thanh ghi đó trỏ tới sau đó in ra giá trị tuyệt đối, ngược lại nếu \$a0 lớn hơn 0 thì cập nhật lại \$v0 = \$a0 và nhảy về vị trí cũ và in ra.
- Giá trị truyền vào là giá trị âm

```
#Laboratory Exercise 7 Home Assignment 1
.data
Message: .asciiz "Nhap vao so:"
Message1: .asciiz "Ket qua tri tuyet doi la: "
.text
main:    #li $v0, 51
        #la $a0, Message
        #syscall
        li $a0, -45 #load input parameter
        jal abs #jump and link to abs procedure
        nop
        #add $s0, $zero, $v0
        #li $v0, 56
        #la $a0, Message1
        #la $a1, 0($s0)
        #syscall
        li $v0, 10 #terminate
        syscall

endmain:
#-----
# function abs
# param[in] $a0 the interger need to be gained the absolute value
# return $v0 absolute value
#-----
abs:
    sub $v0,$zero,$a0 #put -(a0) in v0; in case (a0)<0
    bltz $a0,done #if (a0)<0 then done
    nop
    add $v0,$a0,$zero #else put (a0) in v0
done:
    jr $ra
```

\$v0	2	45
\$v1	3	0
\$a0	4	-45

- Sự thay đổi giá trị thanh ghi

Trạng thái	\$v0	\$ra	\$pc
Ban đầu	0x00000000	0x00000000	0x00400000
Sau khi khởi tạo giá trị	-	-	0x00400004
Sau lệnh jal	-	0x00400008	0x00400014
Sau lệnh sub	0x0000002d	-	0x00400018
Sau lệnh bltz	-	-	0x00400024
Sau lệnh jr	-	-	0x00400008
Sau lệnh syscall	0x0000000a	-	0x00400010

- Giá trị truyền vào là giá trị dương

\$v0	2	63
\$v1	3	0
\$a0	4	63

- Sự thay đổi giá trị thanh ghi

Trạng thái	\$v0	\$ra	\$pc
Ban đầu	0x00000000	0x00000000	0x00400000
Sau khi khởi tạo giá trị	-	-	0x00400004
Sau lệnh jal	-	0x00400008	0x00400014
Sau lệnh sub	0xfffffc1	-	0x00400018
Sau lệnh bltz	-	-	0x00400020
Sau lệnh add	0x0000003f	-	0x00400024
Sau lệnh jr	-	-	0x00400008
Sau lệnh syscall	-	-	0x00400010

Assignment 2

- **TEXTCODE**

#Laboratory Exercise 7, Home Assignment 2

.data

Message: .asciiz "Nhap so thu 1:"

Message1: .asciiz "Nhap so thu 2:"

Message2: .asciiz "Nhap so thu 3:"

Message3: .asciiz "So lon nhat trong 3 so la: "

.text

main: #li \$a0,2 #load test input

#li \$a1,6

#li \$a2,9

li \$v0, 51

la \$a0, Message

syscall

move \$s0, \$a0

la \$a0, Message1

syscall

move \$s1, \$a0

la \$a0, Message2

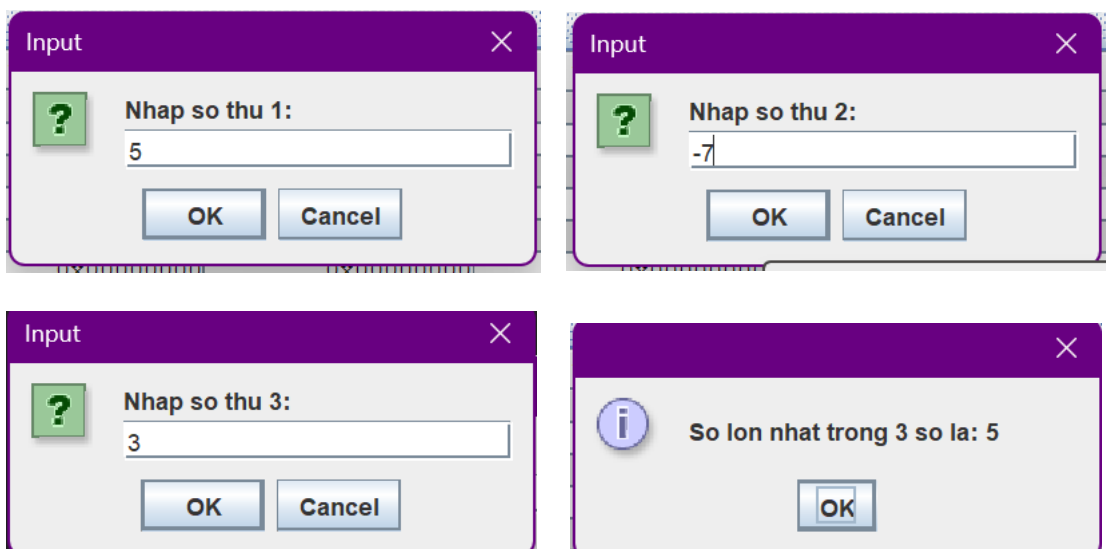
```

syscall
move $a2, $a0
move $a0, $s0
move $a1, $s1
jal max #call max procedure
nop
move $s0, $v0
li $v0, 56
la $a0, Message3
la $a1, 0($s0)
syscall
li $v0, 10 #terminate
syscall

endmain:
#-----
#Procedure max: find the largest of three integers
#param[in] $a0 integers
#param[in] $a1 integers
#param[in] $a2 integers
#return $v0 the largest value
#-----
max:  add $v0,$a0,$zero #copy (a0) in v0; largest so far
      sub $t0,$a1,$v0 #compute (a1)-(v0)
      bltz $t0,okay #if (a1)-(v0)<0 then no change
      nop
      add $v0,$a1,$zero #else (a1) is largest thus far
okay: sub $t0,$a2,$v0 #compute (a2)-(v0)
      bltz $t0,done #if (a2)-(v0)<0 then no change
      nop
      add $v0,$a2,$zero #else (a2) is largest overall
done: jr $ra #return to calling program

```

- **RESULT**



- Chương trình truyền vào 3 giá trị 2, 6, 9 vào 3 thanh ghi \$a0, \$a1, \$a2

- Sau đó câu lệnh jal nhảy đến thẻ max và lưu địa chỉ của lệnh tiếp theo vào thanh ghi \$ra
- Thẻ max truyền giá trị thanh ghi \$a0 vào thanh ghi \$v0 sau đó so sánh giá trị thanh ghi \$a1 và \$v0 nếu \$a1 lớn hơn thì cập nhật lại \$v0 = \$a1 sau đó sẽ so sánh \$v0 và \$a2 nếu \$a2 lớn hơn thì cập nhật lại \$v0 = \$a2, còn lại nếu \$a1 nhỏ hơn thì sẽ so sánh \$s2 và \$v0 thì cập nhật lại như trường hợp trên còn ngược lại sẽ nhảy đến thẻ done và nhảy đến câu lệnh ngay sau câu lệnh jal

```
#Laboratory Exercise 7, Home Assignment 2
.text
main:    li $a0,2 #load test input
         li $a1,6
         li $a2,9
         jal max #call max procedure
         li $v0, 10 #terminate
         syscall

endmain:
#-----
#Procedure max: find the largest of three integers
#param[in] $a0 integers
#param[in] $a1 integers
#param[in] $a2 integers
#return $v0 the largest value
#-----
max:     add $v0,$a0,$zero #copy (a0) in v0; largest so far
         sub $t0,$a1,$v0 #compute (a1)-(v0)
         bltz $t0,okay #if (a1)-(v0)<0 then no change
         nop
         add $v0,$a1,$zero #else (a1) is largest thus far
okay:    sub $t0,$a2,$v0 #compute (a2)-(v0)
         bltz $t0,done #if (a2)-(v0)<0 then no change
         nop
         add $v0,$a2,$zero #else (a2) is largest overall
done:    jr $ra #return to calling program
```

\$v0	2	0x00000009
------	---	------------

- Sự thay đổi giá trị thanh ghi:

Trạng thái	\$v0	\$ra	\$pc
Ban đầu	0x00000000	0x00000000	0x00400000
Sau khi khởi tạo giá trị	-	-	0x0040000c
Sau lệnh jal	-	0x00400010	0x00400018
Sau lệnh add	0x00000002	-	0x0040001c
Sau lệnh sub	-	-	0x00400020
Sau lệnh bltz	-	-	0x00400024
Sau lệnh add \$v0,\$a1,\$zero	0x00000003	-	0x0040002c
Sau lệnh sub	-	-	0x00400030
Sau lệnh bltz	-	-	0x00400034
Sau lệnh add \$v0,\$a2,\$zero	0x00000009	-	0x0040003c

Sau lệnh jr	-	-	0x00400010
Sau lệnh syscall	0x0000000c	-	0x00400018

Assignment 3

```
#Laboratory Exercise 7, Home Assignment 3
.text
data:
    li $s0, 4
    li $s1, 5
push:  addi $sp,$sp,-8 #adjust the stack pointer
        sw $s0,4($sp) #push $s0 to stack
        sw $s1,0($sp) #push $s1 to stack
work:  nop
        nop
        nop
pop:    lw $s0,0($sp) #pop from stack to $s0
        lw $s1,4($sp) #pop from stack to $s1
        addi $sp,$sp,8 #adjust the stack pointer
```

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x7ffffe00	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000005	0x00000004	0x00000000
0x7ffffe04	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe08	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe0c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe10	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe14	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe18	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe1c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe20	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe24	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe28	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe2c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe30	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe34	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe38	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe3c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe40	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe44	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe48	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe4c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe50	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe54	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe58	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe5c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe60	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe64	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe68	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe6c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe70	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe74	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe78	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe7c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe80	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe84	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe88	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe8c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe90	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe94	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe98	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffe9c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffea0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffea4	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffea8	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffeac	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffffead	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

\$s0	15	0x00000000
\$s1	16	0x00000005
\$s2	17	0x00000004

- Chương trình dùng để hoán đổi 2 số bằng cách dùng stack sau khi khởi tạo giá trị 2 thanh ghi \$s0 và \$s1 thì thực hiện hàm push chuyển vị trí con trỏ stack hiện tại về địa chỉ 2 ô trước đây, đẩy giá trị của \$s0 vào địa chỉ bên phải của stack hiện tại sau đó đẩy giá trị \$s1 vào địa chỉ của stack hiện tại.
- Hàm pop dùng để lấy giá trị từ stack truyền lại vào \$s0 và \$s1, truyền giá trị của địa chỉ stack hiện tại vào \$s0 và giá trị của địa chỉ bên tay phải vào \$s1 sau đó trả lại giá trị của ban đầu của stack
- Sự thay đổi giá trị thanh ghi:

Trạng thái	\$s0	\$s1	\$sp
Ban đầu	0x00000000	0x00000000	0x7fffffc
Sau khi khởi tạo giá trị	0x00000004	0x00000005	-
Sau lệnh addi (Hàm push)	-	-	0x7fffff4
Sau 2 lệnh lw (Hàm pop)	0x00000005	0x00000004	-
Sau lệnh addi (Hàm pop)	-	-	0x7fffffc

- | Data Segment | | | | | | | | |
|--------------|------------|------------|------------|------------|-------------|-------------|-------------|-------------|
| Address | Value (+0) | Value (+4) | Value (+8) | Value (+c) | Value (+10) | Value (+14) | Value (+18) | Value (+1c) |
| 0x7ffffe0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000004 | 0x00000000 |

- Push 4 vào stack

- | Data Segment | | | | | | | | |
|--------------|------------|------------|------------|------------|-------------|-------------|-------------|-------------|
| Address | Value (+0) | Value (+4) | Value (+8) | Value (+c) | Value (+10) | Value (+14) | Value (+18) | Value (+1c) |
| 0x7ffffe0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000005 | 0x00000004 | 0x00000000 |

- Push 5 vào stack

Assignment 4

- **TEXTCODE**

#Laboratory Exercise 7, Home Assignment 4

.data

Message: .asciiz "Ket qua tinh giai thua la: "

Message1: .asciiz "Nhap so can tinh gia thua:"

.text

main: jal WARP

print: add \$a1, \$v0, \$zero # \$a0 = result from N!

li \$v0, 56

la \$a0, Message

syscall

quit: li \$v0, 10 #terminate

syscall

endmain:

#-----

#Procedure WARP: assign value and call FACT

#-----

WARP: sw \$fp,-4(\$sp) #save frame pointer (1)

addi \$fp,\$sp,0 #new frame pointer point to the top (2)

addi \$sp,\$sp,-8 #adjust stack pointer (3)

sw \$ra,0(\$sp) #save return address (4)

#li \$a0,6 #load test input N

li \$v0, 51

la \$a0, Message1

syscall

jal FACT #call fact procedure

nop

lw \$ra,0(\$sp) #restore return address (5)

addi \$sp,\$fp,0 #return stack pointer (6)

lw \$fp,-4(\$sp) #return frame pointer (7)

jr \$ra

wrap_end:

#-----

#Procedure FACT: compute N!

#param[in] \$a0 integer N

#return \$v0 the largest value

#-----

FACT: sw \$fp,-4(\$sp) #save frame pointer

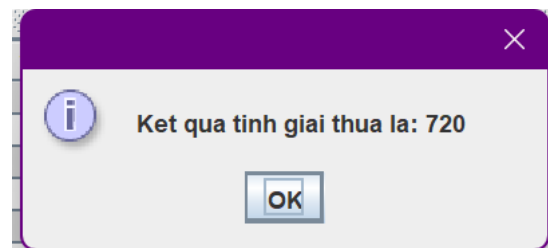
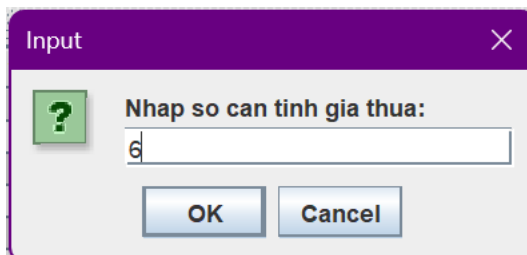
addi \$fp,\$sp,0 #new frame pointer point to stack's top

```

addi $sp,$sp,-12 #allocate space for $fp,$ra,$a0 in stack
sw $ra,4($sp) #save return address
sw $a0,0($sp) #save $a0 register
slti $t0,$a0,2 #if input argument N < 2
beq $t0,$zero,recursive #if it is false ((a0 = N) >=2)
nop
li $v0,1 #return the result N!=1
j done
nop
recursive:
addi $a0,$a0,-1 #adjust input argument
jal FACT #recursive call
nop
lw $v1,0($sp) #load a0
mult $v1,$v0 #compute the result
mflo $v0
done: lw $ra,4($sp) #restore return address
lw $a0,0($sp) #restore a0
addi $sp,$fp,0 #restore stack pointer
lw $fp,-4($sp) #restore frame pointer
jr $ra #jump to calling
fact_end:

```

• RESULT



- Sự thay đổi giá trị thanh ghi: (với \$a0 = 3)

	\$pc	\$ra	\$sp	\$fp
Ban đầu	4194304	0	2147479548	0
jal WARP	4194336	4194308	2147479548	0
jal FACT (1)	4194380	4194360	2147479540	2147479548
jal FACT (2)	4194380	4194332	2147479528	2147479540
jal FACT (3)	4194380	4194332	2147479516	2147479528
j done	4194348	4194332	2147479504	2147479516
jr \$ra (1)	4194332	4194332	2147479516	2147479528
jr \$ra (2)	4194380	4194332	2147479528	2147479540

Trước jr \$ra (3)	4194364	4194360	2147479540	2147479548
Sau jr \$ra (3)	4194360	4194360	2147479540	2147479548
Tại print	4194308	4194308	2147479548	0

Assignment 5

- **TEXTCODE**

.data

a: .word 0

A: .space 32

Message: .asciiz "Nhap so: "

Message1: .asciiz "Gia tri lon nhat la: "

Message2: .asciiz " o thanh ghi \$s"

Message3: .asciiz "Gia tri nho nhat la: "

Newline: .asciiz "\n"

.text

start:

la \$a0, A

addi \$t0, \$a0, 0

input:

beq \$t1, 8, end_input #if i = 8 end_input

li \$v0, 4

la \$a0, Message

syscall

li \$v0, 5

syscall #input number

move \$t2, \$v0

sw \$t2, 0(\$t0) # \$t0 = A[i]

addi \$t0, \$t0, 4 # address of A[i]

addi \$t1, \$t1, 1 # i++

j input

end_input:

```

    la $t0, A
    addi $a0, $zero, 0
    addi $t1, $zero, 0
    addi $t2, $zero, 0    #reset register
load_value:
    lw $s0, 0($t0) #load $s0
    lw $s1, 4($t0) #load $s1
    lw $s2, 8($t0) #load $s2
    lw $s3, 12($t0) #load $s3
    lw $s4, 16($t0) #load $s4
    lw $s5, 20($t0) #load $s5
    lw $s6, 24($t0) #load $s6
    lw $s7, 28($t0) #load $s7
main:    jal WARP
print:    add $a1, $v0, $zero
    li $v0, 4
    la $a0, Message1
    syscall
    li $v0, 1
    addi $a0, $a1, 0
    syscall
    li $v0, 4
    la $a0, Message2
    syscall
    li $v0, 1
    addi $a0, $t2, 0
    syscall    # print lagerst
    li $v0, 4
    la $a0, Newline
    syscall
    la $a0, Message3
    syscall

```

```

li $v0, 1
addi $a0, $v1, 0
syscall

li $v0, 4
la $a0, Message2
syscall      # print smallest

li $v0, 1
addi $a0, $t3, 0
syscall

quit:      li $v0, 10 # terminate
          syscall

endmain:

WARP:     la $t0, a      # address of A[-1]
          addi $a0,$a0,-1 # j = -1
          sw $fp,-4($sp) # save frame pointer
          addi $fp,$sp,0 # new frame pointer to top
          addi $sp,$sp,-8 # next stack
          sw $ra,0($sp) # save return adress
          jal stack
          nop
          lw $ra,0($sp) # restore address from stack
          addi $sp,$fp,0 # return stack pointer
          lw $fp,-4($sp) # return frame pointer
          jr $ra

wrap_end:

stack:     sw $fp,-4($sp) # save frame pointer
          addi $fp,$sp,0 # new frame pointer to top
          addi $sp,$sp,-16 # create space for $ra, $a0, $A[i]( value of register s(j))
          sw $ra,8($sp) # save return address
          sw $a0,4($sp) # save number of register save value
          lw $t1,0($t0) # $t1 = A[i] = value of s(j)
          sw $t1,0($sp) # save s(j)

```

```

bne $a0,7,recursive #if j = 7 recursive
nop
lw $v0, 0($sp) # save max value
lw $v1, 0($sp) # save min value
lw $t2, 4($sp) # save number of register save max value
lw $t3, 4($sp) # save number of register save min value
j min_max
nop

```

recursive:

```

addi $a0,$a0,1 # j++
addi $t0, $t0, 4 # address of A[j]
jal stack
nop
j find_max
nop

```

min_max:

```

lw $ra,8($sp) # save return address
lw $t1, 0($sp) # save temp value
lw $t4, 4($sp) # save temp number of register
addi $sp,$fp,0 # restore stack pointer
lw $fp,-4($sp) # restore frame pointer
jr $ra

```

find_max:

```

bgt $t1, $v0, max #if temp_value > max
j find_min

```

max:

```

addi $v0, $t1, 0 # max = temp_value
addi $t2, $t4, 0 # number of register = temp number of register
nop

```

find_min:

```

blt $t1, $v1, min # if temp_vale < min
j min_max

```

min:

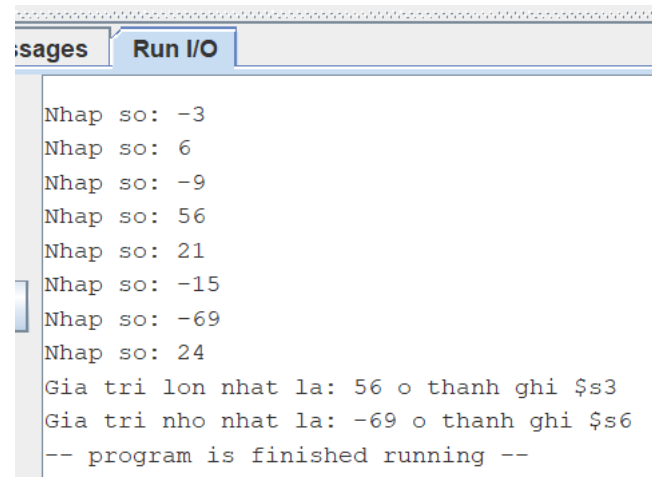
```
addi $v1, $t1, 0 # min = temp value
```

```
addi $t3, $t4, 0 # number of register = temp number of register
```

```
j min_max
```

```
nop
```

- **RESULT**



```
Messages Run I/O
Nhap so: -3
Nhap so: 6
Nhap so: -9
Nhap so: 56
Nhap so: 21
Nhap so: -15
Nhap so: -69
Nhap so: 24
Gia tri lon nhat la: 56 o thanh ghi $s3
Gia tri nho nhat la: -69 o thanh ghi $s6
-- program is finished running --
```

\$s0	16	-3
\$s1	17	6
\$s2	18	-9
\$s3	19	56
\$s4	20	21
\$s5	21	-15
\$s6	22	-69
\$s7	23	24