

BÀI TẬP TUẦN 7

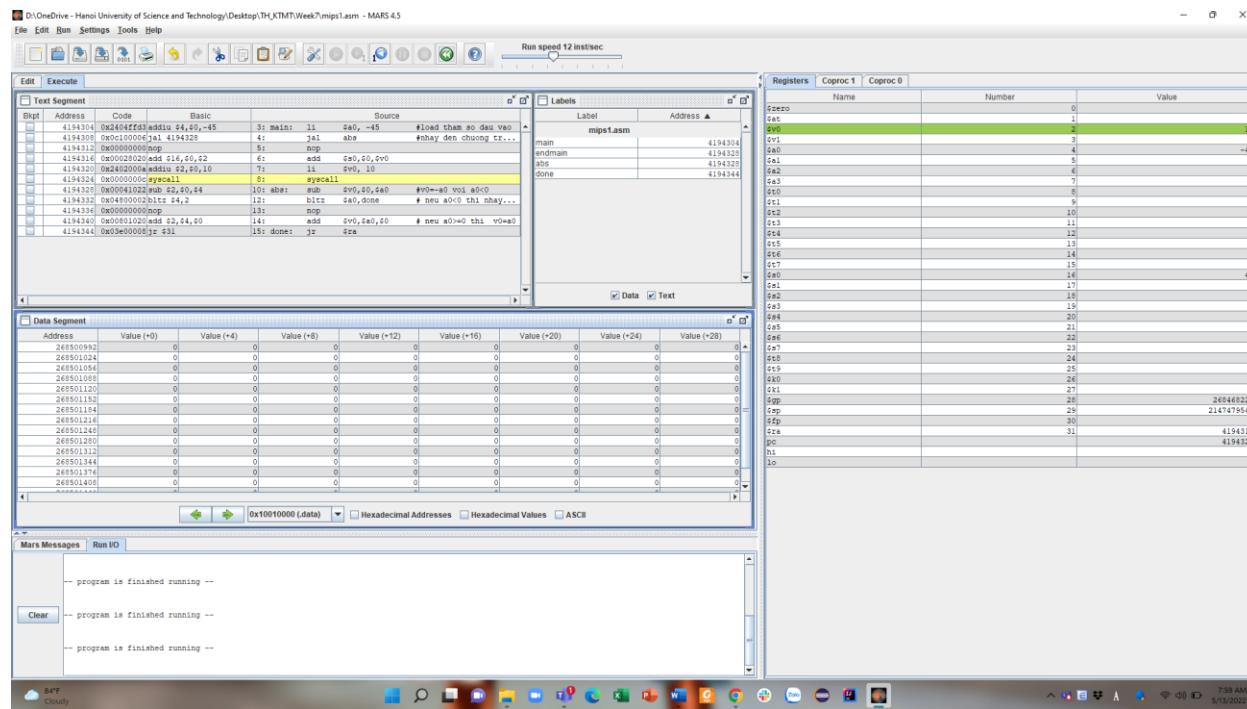
Exercise 1:

- Chương trình:

```
#Laboratory Exercise 7 Home Assignment 1
.text
main: li      $a0, -45          #load tham so dau vao
jal     abs                  #nhay den chuong trinh con abs va luu dia chi tra ve trong thanh ghi $ra
nop
add    $s0,$0,$v0
li      $v0, 10
syscall
endmain:
abs:   sub    $v0,$0,$a0        #v0=- (a0) voi (a0)<0

bltz   $a0,done            # neu (a0)<0 thi nhay den done
nop
add    $v0,$a0,$0          # neu (a0)>=0 thi v0=(a0)
done:  jr    $ra
```

- Kết quả:



- Giải thích:

Trước khi chạy đến chương trình con abs:

+ \$ra là 0. (\$ra = 0)

+ pc ghi địa chỉ của câu lệnh đang trả tới. (pc = 4194304)

Register	Name	Number	Value
zero		0	0
at		1	0
v0		2	0
v1		3	0
ra		4	0
tl		5	0
gp		6	0
sp		7	0
tp		8	0
t0		9	0
t1		10	0
t2		11	0
t3		12	0
t4		13	0
t5		14	0
t6		15	0
d0		16	0
d1		17	0
d2		18	0
d3		19	0
d4		20	0
d5		21	0
d6		22	0
d7		23	0
d8		24	0
d9		25	0
d10		26	0
d11		27	0
d12		28	268468224
d13		29	2147419544
d14		30	0
d15		31	0
pc			4194304
hi			0
lo			0

Sau khi chạy đến chương trình con abs:

+ \$ra ghi địa chỉ của câu lệnh liền sau câu lệnh jal là nop. (\$ra = 4194312)

+ pc ghi địa chỉ của câu lệnh đang trả tới. (pc = 4194328)

D:\OneDrive - Hanoi University of Science and Technology\Desktop\TH_KTMT\Week7\mips1.asm - MARS 4.5

Registers

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$a0	3	0
\$a1	4	-45
\$a2	5	0
\$a3	6	0
\$t0	7	0
\$t1	8	0
\$t2	9	0
\$t3	10	0
\$t4	11	0
\$t5	12	0
\$t6	13	0
\$t7	14	0
\$t8	15	0
\$t9	16	0
\$t10	17	0
\$t11	18	0
\$t12	19	0
\$t13	20	0
\$t14	21	0
\$t15	22	0
\$t16	23	0
\$t17	24	0
\$t18	25	0
\$t19	26	0
\$t20	27	0
\$t21	28	0
\$t22	29	268468224
\$t23	30	214745443
\$t24	31	4194312
\$t25	32	4194328
\$t26	33	0
\$t27	34	0
\$t28	35	0
\$t29	36	0
\$t30	37	0
\$t31	38	0

Sau khi kết thúc chương trình:

- + \$ra ghi địa chỉ của câu lệnh liền sau câu lệnh jal. (\$ra = 4194312)
- + pc ghi địa chỉ của câu lệnh đang trả tới. (pc = 4194328)

D:\OneDrive - Hanoi University of Science and Technology\Desktop\TH_KTMT\Week7\mips1.asm - MARS 4.5

Registers

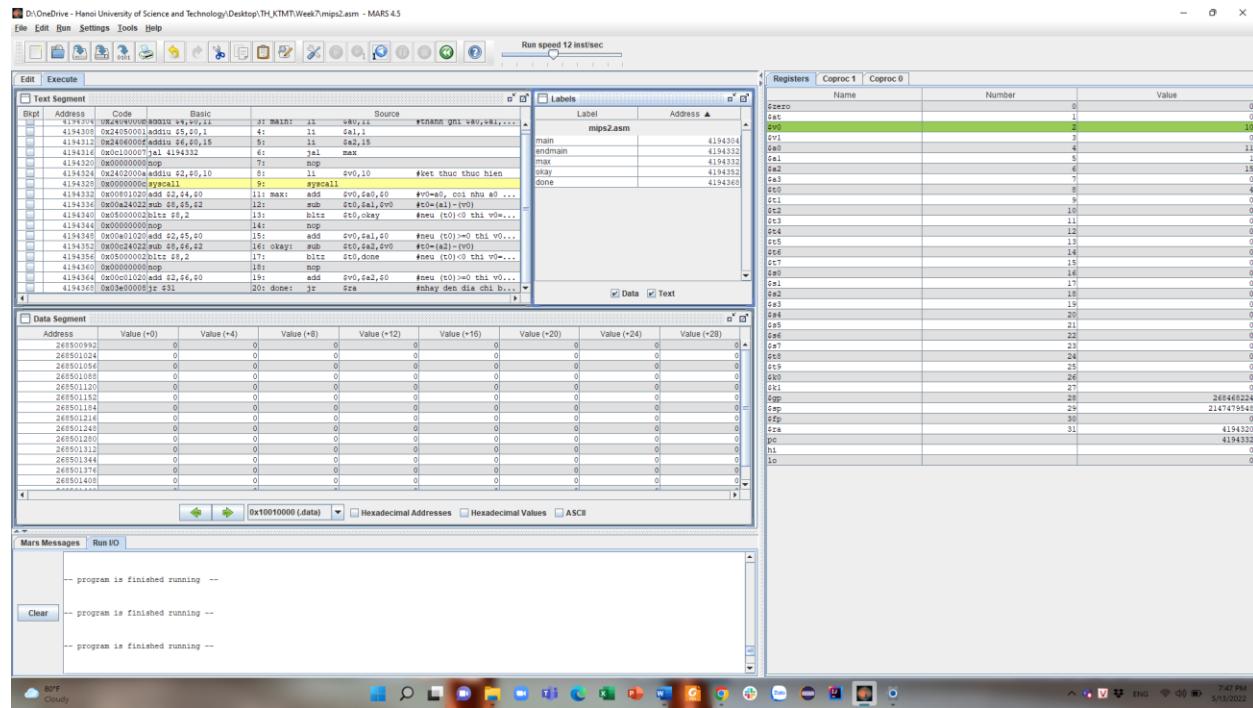
Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$a0	3	0
\$a1	4	-45
\$a2	5	0
\$a3	6	0
\$t0	7	0
\$t1	8	0
\$t2	9	0
\$t3	10	0
\$t4	11	0
\$t5	12	0
\$t6	13	0
\$t7	14	0
\$t8	15	0
\$t9	16	45
\$t10	17	0
\$t11	18	0
\$t12	19	0
\$t13	20	0
\$t14	21	0
\$t15	22	0
\$t16	23	0
\$t17	24	0
\$t18	25	0
\$t19	26	0
\$t20	27	0
\$t21	28	268468224
\$t22	29	214745443
\$t23	30	4194312
\$t24	31	4194328
\$t25	32	0
\$t26	33	0
\$t27	34	0
\$t28	35	0
\$t29	36	0
\$t30	37	0

Exercise 2:

- Chương trình:

```
#Laboratory Exercise 7, Home Assignment 2
.text
main: li      $a0,11          #than ghi $a0,$a1,$a2 luu cac so de test
      li      $a1,1
      li      $a2,15
      jal     max
      nop
      li      $v0,10          #ket thuc thuc hien
      syscall
endmain:
max:   add    $v0,$a0,$0      #v0=a0, coi nhu a0 la so lon nhat
      sub    $t0,$a1,$v0      #t0=(a1)-(v0)
      blitz $t0,okay        #neu (t0)<0 thi v0=(a0)
      nop
      add    $v0,$a1,$0      #neu (t0)>=0 thi v0=(a1), coi nhu a1 la so lon nhat cho den hien tai
okay:  sub    $t0,$a2,$v0      #t0=(a2)-(v0)
      blitz $t0,done         #neu (t0)<0 thi v0=(a0)
      nop
      add    $v0,$a2,$0      #neu (t0)>=0 thi v0=(a2)
done:  jr    $ra              #nhay den dia chi ben trong thanh ghi $ra
```

- Kết quả:

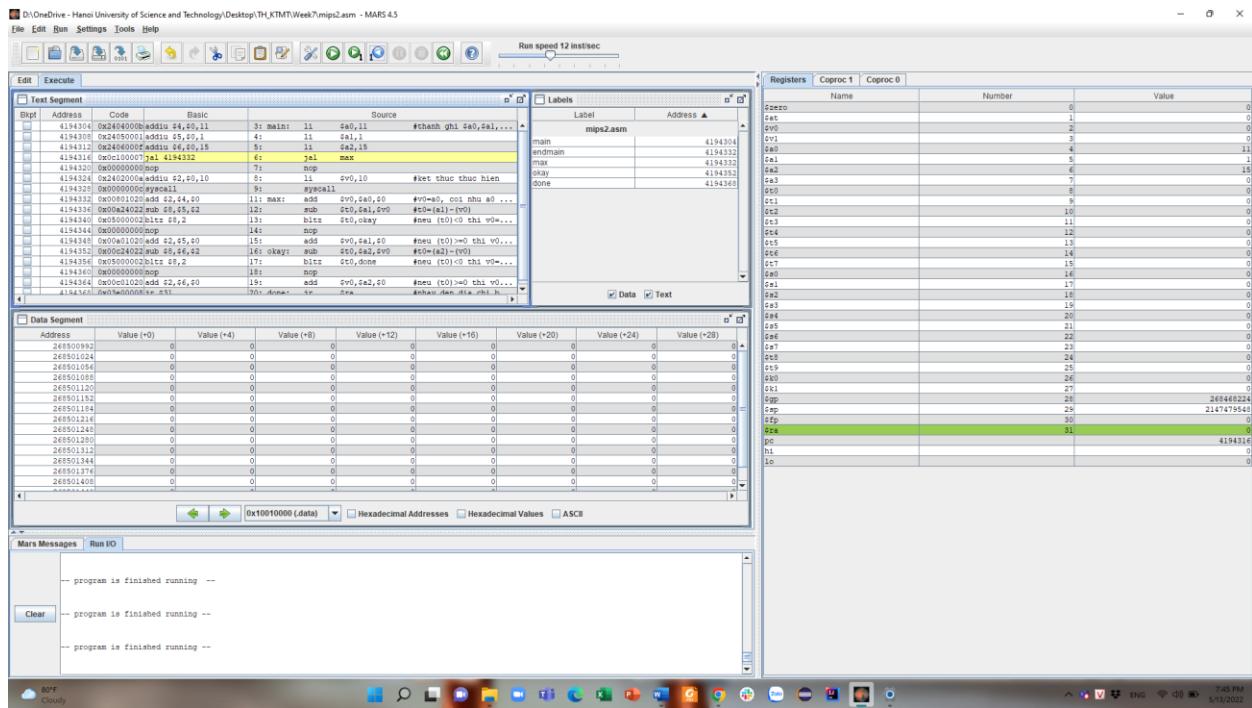


- Giải thích:

Trước khi chạy đến chương trình con max:

+ \$ra là 0. (\$ra = 0)

+ pc ghi địa chỉ của câu lệnh đang trả tới. (pc = 4194316)



Sau khi chạy đến chương trình con max:

+ \$ra ghi địa chỉ của câu lệnh liền sau câu lệnh jal là nop. (\$ra = 4194320)

+ pc ghi địa chỉ của câu lệnh đang trả tới. (pc = 4194332)

Sau khi kết thúc chương trình:

- + \$ra ghi địa chỉ của câu lệnh liền sau câu lệnh jal là nop. (\$ra = 4194320)
 - + pc ghi địa chỉ của câu lệnh đang trả tới. (pc = 4194332)

The screenshot shows the MARS 4.5 assembly editor interface. The main window displays the assembly code for 'mips2.asm' with various instructions like add, sub, and loop labels. The registers pane shows the state of the CPU registers. The memory dump pane shows the memory starting at address 0x10000000. The messages pane at the bottom left shows three entries: "program is finished running --". The status bar at the bottom right shows system information including battery level, signal strength, and network connection.

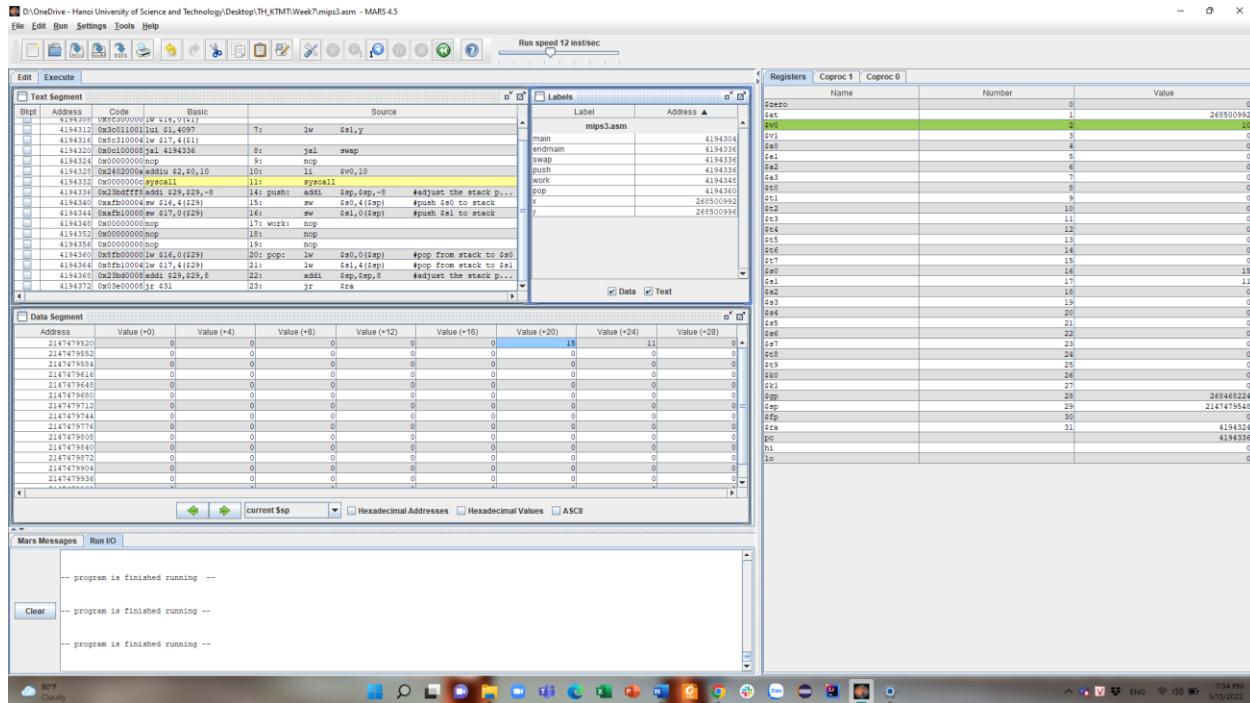
Exercise 3:

- Chương trình:

```
#Laboratory Exercise 7, Home Assignment 3
.data
x: .word 11          #khai bao x,y
y: .word 15

.text
main: lw    $s0,x
       lw    $s1,y
       jal   swap
       nop
       li    $v0,10
       syscall
endmain:
swap:
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
      jr   $ra
```

- Kết quả:

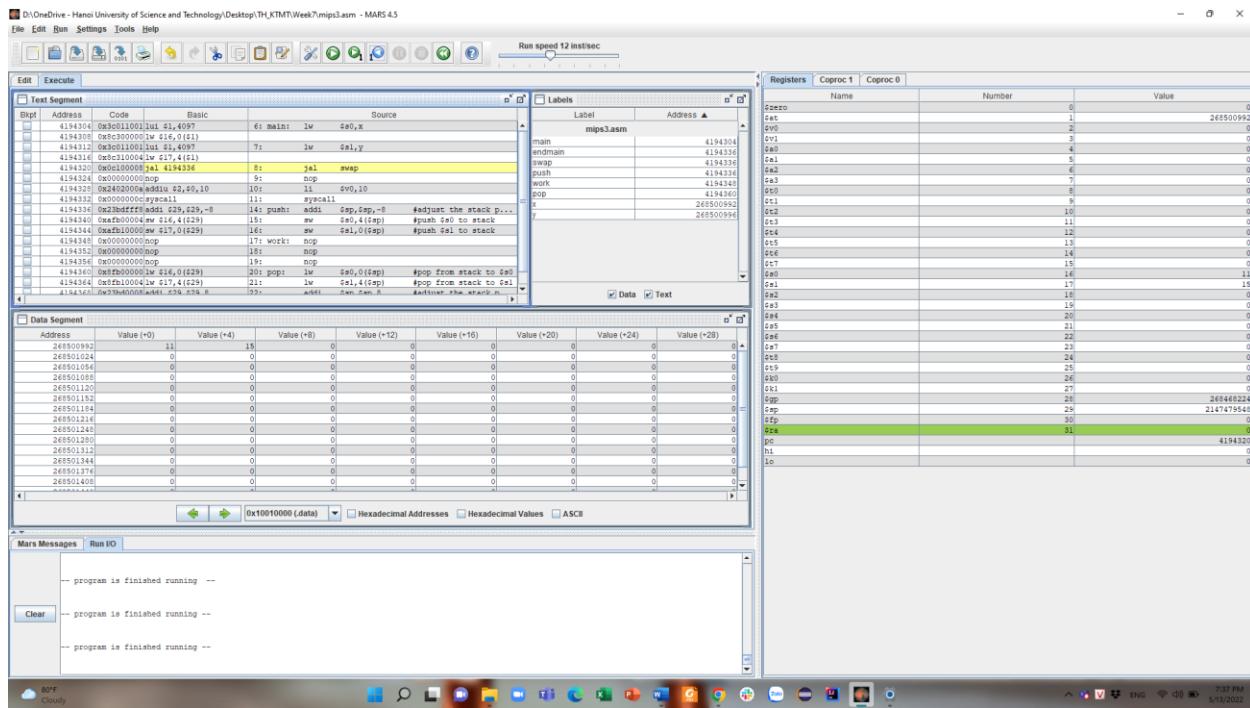


- Giải thích:

Trước khi chạy đến chương trình con swap:

+ \$ra là 0. (\$ra = 0)

+ pc ghi địa chỉ của câu lệnh đang trả tới. (pc = 4194320)



Sau khi chạy đến chương trình con swap:

+ \$ra ghi địa chỉ của câu lệnh liền sau câu lệnh jal là nop. (\$ra = 4194324)

+ pc ghi địa chỉ của câu lệnh đang trả tới. (pc = 4194336)

D:\OneDrive - Hanoi University of Science and Technology\Desktop\TH_KTMT\Week7\mips3.asm - MARS 4.5

Registers

Name	Number	Value
\$zero	0	0
\$at	1	268500992
\$v0	2	0
\$v1	3	0
\$t0	4	0
\$t1	5	0
\$t2	6	0
\$t3	7	0
\$t4	8	0
\$t5	9	0
\$t6	10	0
\$t7	11	0
\$t8	12	0
\$t9	13	0
\$t10	14	0
\$t11	15	0
\$t12	16	11
\$t13	17	15
\$t14	18	0
\$t15	19	0
\$t16	20	0
\$t17	21	0
\$t18	22	0
\$t19	23	0
\$t20	24	0
\$t21	25	0
\$t22	26	0
\$t23	27	0
\$t24	28	268466224
\$t25	29	214747944
\$t26	30	0
\$ra	31	4194324
pc		4194336
hi		0
lo		0

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	11	15	0	0	0	0	0	0
26850124	0	0	0	0	0	0	0	0
268501594	0	0	0	0	0	0	0	0
268501852	0	0	0	0	0	0	0	0
268501120	0	0	0	0	0	0	0	0
268501152	0	0	0	0	0	0	0	0
268501184	0	0	0	0	0	0	0	0
268501216	0	0	0	0	0	0	0	0
268501240	0	0	0	0	0	0	0	0
268501250	0	0	0	0	0	0	0	0
268501272	0	0	0	0	0	0	0	0
268501344	0	0	0	0	0	0	0	0
268501376	0	0	0	0	0	0	0	0
268501408	0	0	0	0	0	0	0	0

Mars Messages

```
-- program is finished running --
-- program is finished running --
-- program is finished running --
```

Sau khi kết thúc chương trình:

- + \$ra ghi địa chỉ của câu lệnh liền sau câu lệnh jal là nop. (\$ra = 4194324)
- + pc ghi địa chỉ của câu lệnh đang trả tối. (pc = 4194336)

D:\OneDrive - Hanoi University of Science and Technology\Desktop\TH_KTMT\Week7\mips3.asm - MARS 4.5

Registers

Name	Number	Value
\$zero	0	0
\$at	1	268500993
\$v0	2	19
\$v1	3	0
\$t0	4	0
\$t1	5	0
\$t2	6	0
\$t3	7	0
\$t4	8	0
\$t5	9	0
\$t6	10	0
\$t7	11	0
\$t8	12	0
\$t9	13	0
\$t10	14	0
\$t11	15	0
\$t12	16	19
\$t13	17	11
\$t14	18	0
\$t15	19	0
\$t16	20	0
\$t17	21	0
\$t18	22	0
\$t19	23	0
\$t20	24	0
\$t21	25	0
\$t22	26	0
\$t23	27	268466224
\$t24	28	214747944
\$t25	29	0
\$t26	30	0
\$ra	31	4194324
pc		4194336
hi		0
lo		0

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
214747932	0	0	0	0	0	19	11	0
214747952	0	0	0	0	0	0	0	0
214747964	0	0	0	0	0	0	0	0
214747984	0	0	0	0	0	0	0	0
214747996	0	0	0	0	0	0	0	0
214747998	0	0	0	0	0	0	0	0
214747999	0	0	0	0	0	0	0	0
214747950	0	0	0	0	0	0	0	0
214747951	0	0	0	0	0	0	0	0
214747952	0	0	0	0	0	0	0	0
214747954	0	0	0	0	0	0	0	0
214747956	0	0	0	0	0	0	0	0
214747957	0	0	0	0	0	0	0	0
214747958	0	0	0	0	0	0	0	0
214747959	0	0	0	0	0	0	0	0
214747960	0	0	0	0	0	0	0	0
214747961	0	0	0	0	0	0	0	0
214747962	0	0	0	0	0	0	0	0
214747963	0	0	0	0	0	0	0	0

Mars Messages

```
-- program is finished running --
-- program is finished running --
-- program is finished running --
```

Exercise 4:

a, Khi chương trình sử dụng thanh ghi \$fp

- *Chương trình:*

```
#Laboratory Exercise 7, Home Assignment 4
.data
Message: .ascii "Ket qua tinh giai thua la: "
.text
main: jal WARP

print: add $a1,$v0,$0,      #$a0 = result from N!
       li $v0,56
       la $a0,Message
       syscall
quit:  li $v0,10           #ket thuc thuc hien
       syscall
endmain:
WARP:  sv $fp,-4($sp)    #save frame pointer
       addi $sp,$sp,0      #new frame pointer point to the top
       addi $sp,$sp,-8      #adjust stack pointer
       sw $ra,0($sp)       #save return address

       li $a0,3            #load test input N
       jal FACT             #call fact procedure
       nop

       lw $ra,0($sp)       #restore return address
       addi $sp,$sp,0      #return stack pointer
       lw $fp,-4($sp)     #return frame pointer
       jr $ra

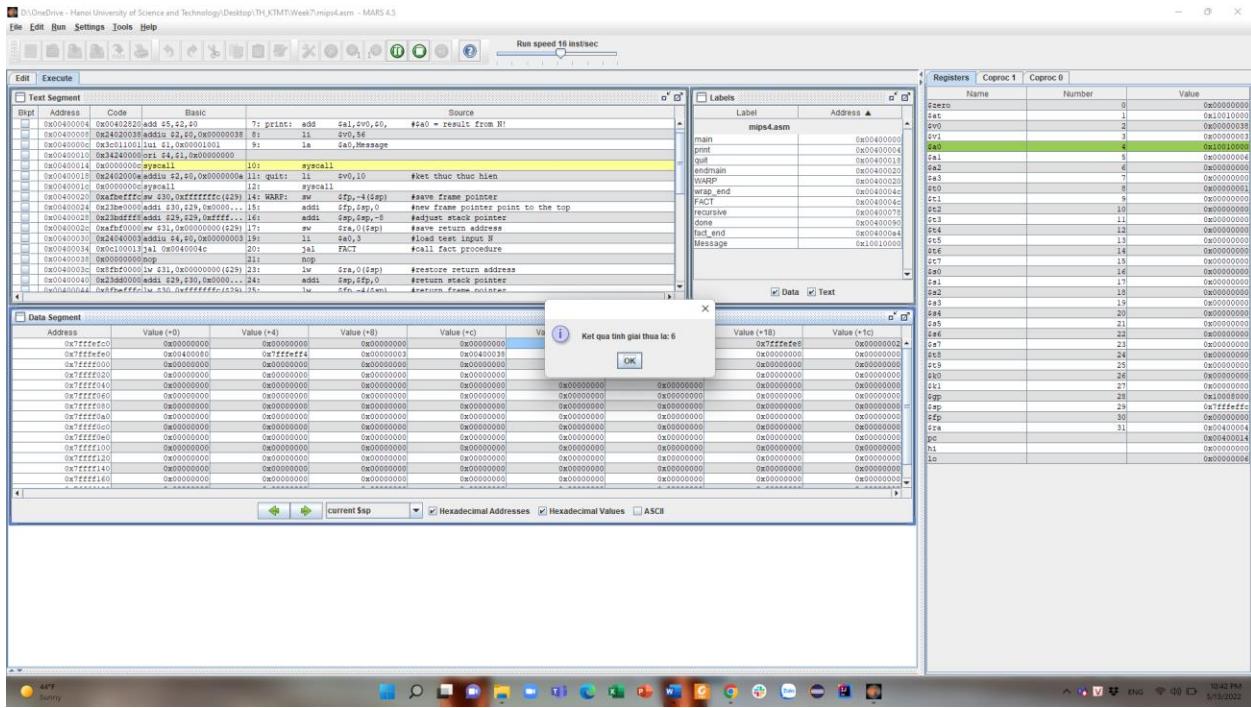
warp_end:
FACT:  sv $fp,-4($sp)    #save frame pointer
       addi $sp,$sp,0      #new frame pointer point to stack's top
       addi $sp,$sp,-12    #allocate space for $fp,$ra,$a0 in stack
       sv $ra,4($sp)       #save return address
       sv $a0,0($sp)       #save $a0 register
       slti $t0,$a0,2       #if input argument N < 2
       beq $t0,$zero,recursive  #if it is false ((a0 = N) >=2)
       nop
       li $v0,1            #return the result N=i
       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:
```

- *Kết quả:*



- Giải thích:

+ Trước khi vào chương trình con WARP:

\$sp	29	0x7ffffeffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400000

+ Trong hàm WARP và chạy xong sw

\$sp	29	0x7ffffeff4
\$fp	30	0x7ffffeffc
\$ra	31	0x00400004
pc		0x00400030

Stack:

Value (+14)	Value (+18)	Value (+1c)
0x00400004	0x00000000	0x00000000

$$\$ra(\text{main}) = 0x00400004$$

$$0(\$sp) = 0x00400004$$

$$\$fp(\text{main}) = 0x00000000$$

$$0(\$fp) = 0x00000000$$

+ Trong hàm FACT (\$a0=3) và chạy xong sw

\$sp	29	0x7fffffe8
\$fp	30	0x7ffffeff4
\$ra	31	0x00400038
pc		0x00400060

Stack:

Value (+8)	Value (+c)	Value (+10)	Value (+14)
0x00000003	0x00400038	0x7ffffeffc	0x00400004

\$a0=3

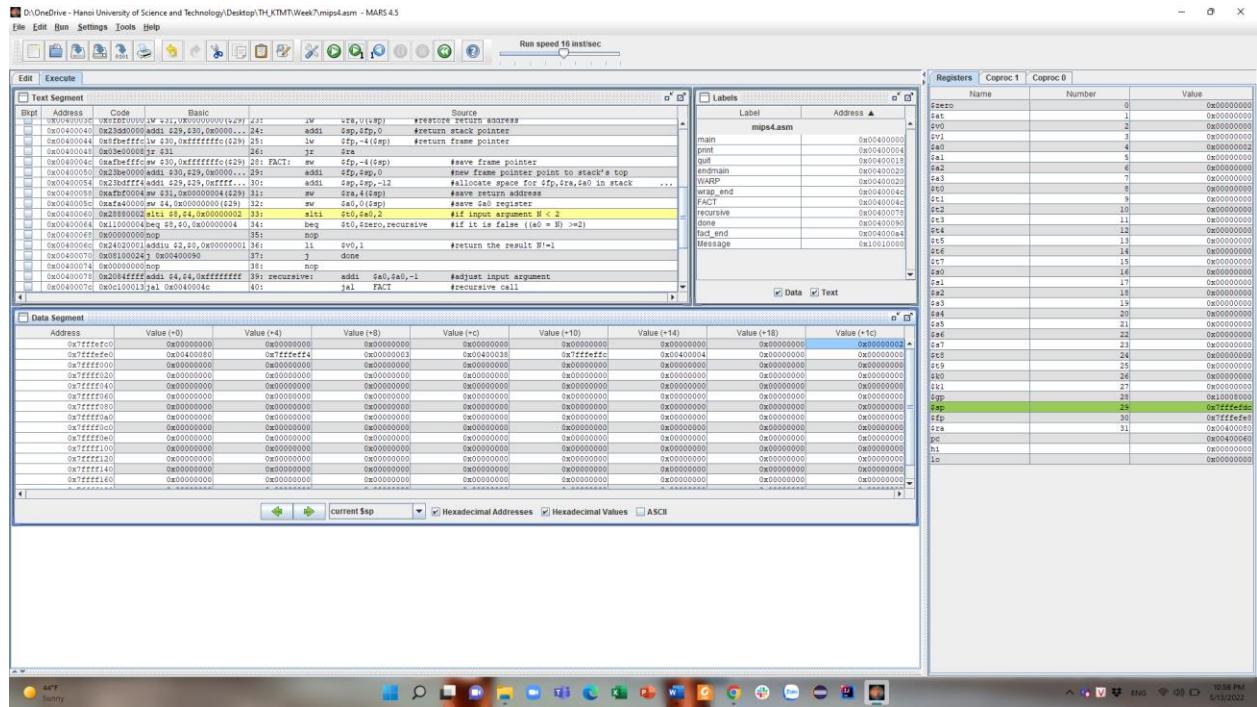
\$ra(WARP)= 0x00400038

\$fp(WARP)= 0x7ffffeffc

0(\$sp)= 0x00000003

0(\$fp) = 0x00400004

+ Trong hàm FACT(\$a0=2) và chạy xong sw



+ Trong hàm FACT(\$a0=1) và chạy xong sw

The screenshot displays the MARS 4.5 assembly editor interface with the following windows open:

- Registers**: Shows the state of various CPU registers:
 - \$zero: 0x00000000
 - \$at: 0x00000000
 - \$v0: 0x00000000
 - \$v1: 0x00000000
 - \$a0: 0x00000000
 - \$a1: 0x00000000
 - \$a2: 0x00000000
 - \$a3: 0x00000000
 - \$t0: 0x00000000
 - \$t1: 0x00000000
 - \$t2: 0x00000000
 - \$t3: 0x00000000
 - \$t4: 0x00000000
 - \$t5: 0x00000000
 - \$t6: 0x00000000
 - \$t7: 0x00000000
 - \$t8: 0x00000000
 - \$t9: 0x00000000
 - \$t10: 0x00000000
 - \$t11: 0x00000000
 - \$t12: 0x00000000
 - \$t13: 0x00000000
 - \$t14: 0x00000000
 - \$t15: 0x00000000
 - \$t16: 0x00000000
 - \$t17: 0x00000000
 - \$t18: 0x00000000
 - \$t19: 0x00000000
 - \$t20: 0x00000000
 - \$t21: 0x00000000
 - \$t22: 0x00000000
 - \$t23: 0x00000000
 - \$t24: 0x00000000
 - \$t25: 0x00000000
 - \$t26: 0x00000000
 - \$t27: 0x00000000
 - \$t28: 0x00000000
 - \$t29: 0x00000000
 - \$t30: 0x00000000
 - \$t31: 0x00000000
 - \$gp: 0x00000000
 - \$sp: 0x00000000
 - \$fp: 0xffffffff
 - \$ra: 0x00000000
- Labels**: Lists labels and their addresses:
 - main: 0x00000004
 - print: 0x00000004
 - call: 0x00000004
 - endmain: 0x00000020
 - WARP: 0x00000020
 - wrap_end: 0x00000020
 - fact: 0x00000040
 - recursive: 0x00000070
 - done: 0x00000090
 - fact_end: 0x000000e4
 - Message: 0x10100100
- Data Segment**: Shows memory starting at address 0x00000000:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+tc)
0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000004	0x00000000	0x00000000	0xffffffff	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000008	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x0000000c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000010	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000014	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000018	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000024	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000028	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000032	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000036	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000044	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000048	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000052	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000056	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000064	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000068	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000072	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000076	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000084	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000088	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000092	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000096	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000a4	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000a8	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000b2	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000b6	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000b8	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000bc	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000cc	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000dc	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000ec	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000fc	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
- Stack**: Shows the stack pointer (\$sp) at address 0x00000000 with a value of 0x00000000.
- Registers**: Shows the stack pointer (\$sp) at address 0x00000000 with a value of 0x00000000.
- Labels**: Lists labels and their addresses:
 - main: 0x00000004
 - print: 0x00000004
 - call: 0x00000004
 - endmain: 0x00000020
 - WARP: 0x00000020
 - wrap_end: 0x00000020
 - fact: 0x00000040
 - recursive: 0x00000070
 - done: 0x00000090
 - fact_end: 0x000000e4
 - Message: 0x10100100
- Data Segment**: Shows memory starting at address 0x00000000 with various data values.
- Stack**: Shows the stack pointer (\$sp) at address 0x00000000 with a value of 0x00000000.

b, Khi chương trình không sử dụng thanh ghi \$fp:

- *Chương trình:*

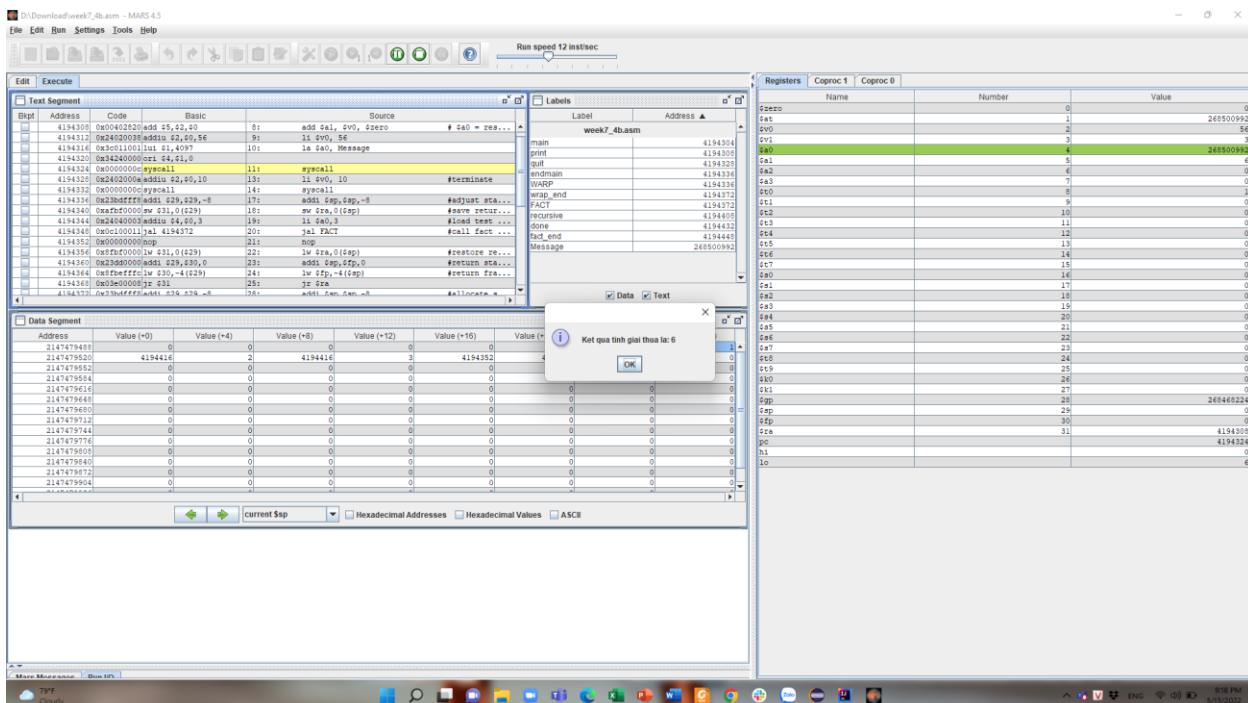
```

#Laboratory Exercise 7, Home Assignment 4
.data
Message: .ascii "Ket qua tinh giai thua la: "
.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:
WARP:
    addi $sp,$sp,-8            #adjust stack pointer (3)
    sw $ra,0($sp)             #save return address (4)
    li $a0,3
    jal FACT
    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:
FACT:
    addi $sp,$sp,-8            #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
    j done
    nop
recursive:
    addi $a0,$a0,-1           #adjust input argument
    jal FACT
    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 $sp,$sp,8              #restore frame pointer
    jr $ra
fact_end:

```

- Kết quả:



Exercise 5:

- Chương trình:

```
#Laboratory Exercise 7, Assignment 5
.data
.Mess1: .asciiz "Largest: "
.Mess2: .asciiz "\nSmallest: "
.Comma: .asciiz ","
.text
main: li      $s0, 11          # Load input
      li      $s1, 1
      li      $s2, 0
      li      $s3, -9
      li      $s4, 15
      li      $s5, 11
      li      $s6, 14
      li      $s7, -20
      jal    init
      nop
      li      $v0, 4
      la      $a0, Mess1      #print mess1
      syscall
      li      $v0, 1
      add   $a0,$t0,$zero
      syscall      #print max value
      li      $v0, 4
      la      $a0, Comma      #print ","
      syscall
      li      $v0, 1
      add   $a0,$t5,$zero      # print max value's position
      syscall
      li      $v0, 4
      la      $a0, Mess2      #print mess2
      syscall
      li      $v0,1
      add   $a0,$t1,$zero
      syscall      #print min value
      li      $v0, 4
      la      $a0, Comma      #print ","
      syscall
      li      $v0, 1
      add   $a0,$t6,$zero
      syscall      # print min value's position
      li      $v0, 10

      syscall      # exit
.endmain:
swapMax: add   $t0,$t3,$zero      # set Max = $t3
      add   $t5,$t2,$zero      # set i of max = $t2
      jr   $ra
swapMin: add   $t1,$t3,$zero      # set Min = $t3
      add   $t6,$t2,$zero      # set i of min = $t2
      jr   $ra
init: add   $fp,$sp,$zero      # save address of origin sp
      addi $sp,$sp, -36      # create space for stack
      sw   $0, 0($sp)
      sw   $s1, 4($sp)
      sw   $s2, 8($sp)
      sw   $s3, 12($sp)
      sw   $s4, 16($sp)
      sw   $s5, 20($sp)
      sw   $s6, 24($sp)
      sw   $s7, 28($sp)
      sw   $ra, 32($sp)      #save $ra for main
      add   $t0,$s0,$zero      # set Max = $s0
      add   $t1,$s0,$zero      # set Min = $s0
      li   $t5, 0            # set $t5 to 0
      li   $t6, 0            # set $t6 to 0
      li   $t2, 0            # set $t2 to 0 , i = 0
max_min: addi $sp,$sp,4
      lw   $t3,-4($sp)
      sub   $t4, $sp, $fp      # check if meet $ra
      beq   $t4,$zero, done      # if true then done
      addi $t2,$t2,1          # i++
      sub   $t4,$t0,$t3      # Max - $t3
      bltzal $t4, swapMax      #if Max - $t3 < 0 , swap Max
      sub   $t4,$t3,$t1      # $t3 - Min
      bltzal $t4, swapMin      # if $t3 < Min < 0 , swap Min
      j     max_min          # repeat
done: lw   $ra, -4($sp)      # load #$ra
      jr   $ra               # return
```

- Kết quả:

Name	Number	Value
\$zero	0	0
\$at	1	246500992
\$v0	2	16
\$t0	3	0
\$a0	4	8
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t1	8	-15
\$t2	9	-20
\$s0	10	0
\$s1	11	4194340
\$s2	12	0
\$s3	13	5
\$s4	14	0
\$s5	15	0
\$s6	16	11
\$s7	17	1
\$s8	18	0
\$s9	19	-9
\$s10	20	15
\$s11	21	11
\$s12	22	14
\$s13	23	-20
\$s14	24	0
\$s15	25	0
\$s16	26	0
\$s17	27	0
\$sp	28	246500224
\$gp	29	246500144
\$tp	30	214745540
\$ra	31	4194340
pc		4194464
hi		0
lo		0

- Giải thích:

Thanh ghi \$t0 lưu giá trị lớn nhất: (t0) = 15

Thanh ghi \$t1 lưu giá trị nhỏ nhất: (t1) = -20