Thực hành KTMT Lab 7

Lê Quốc Đảng

20225801

**Assignment 1**

* **Code**

.text

main: li $a0,-15 #load input parameter

jal abs #jump and link to abs procedure

nop

add $s0, $zero, $v0

li $v0,10 #terminate

syscall

endmain:

#--------------------------------------------------------------------

# function abs

# param[in] $a1 the interger need to be gained the absolute value

# return $v0 absolute value

#--------------------------------------------------------------------

abs:

sub $v0,$zero,$a1 #put -(a0) in v0; in case (a0)<0

bltz $a1,done #if (a0)<0 then done

nop

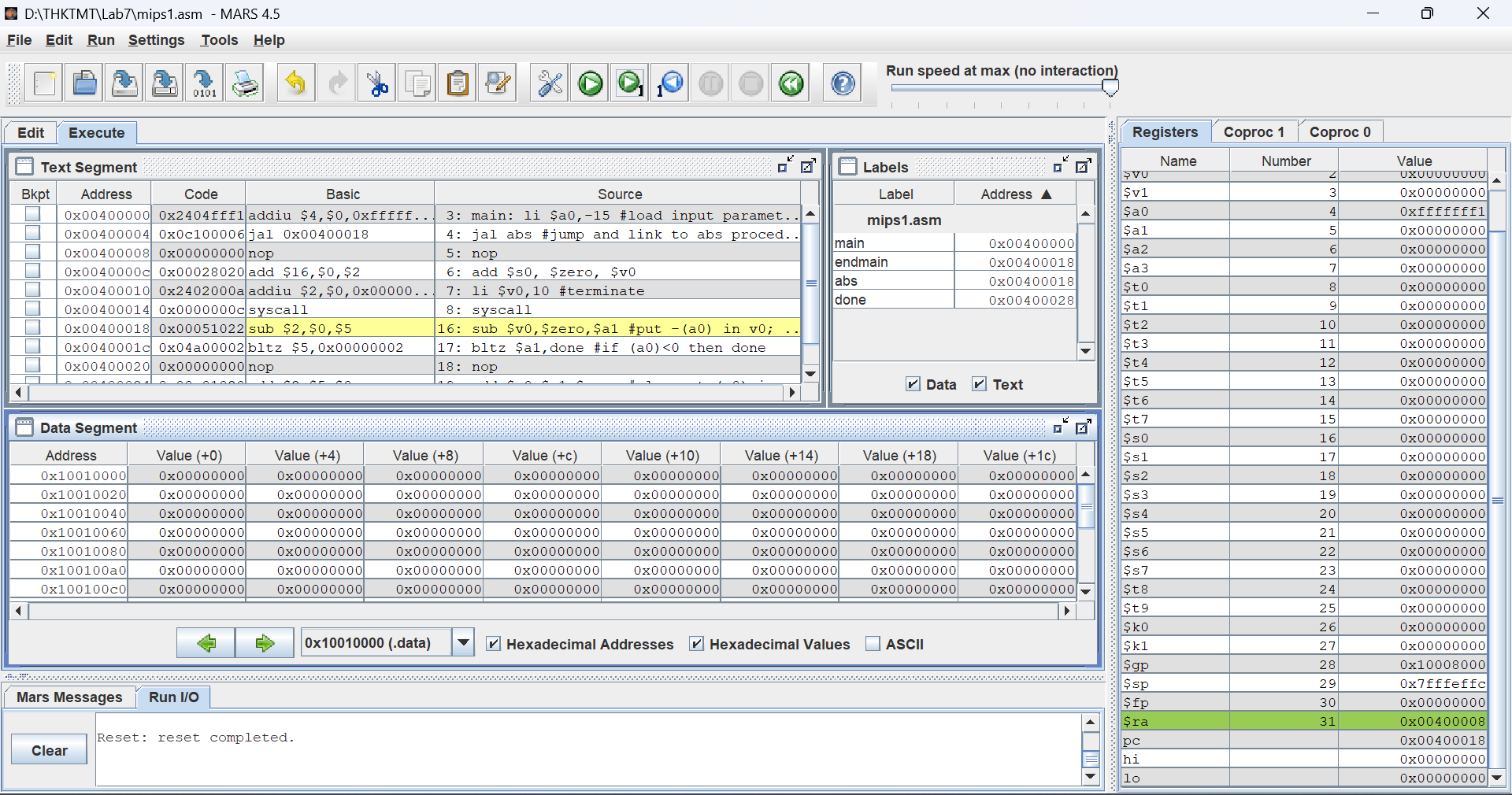
add $v0,$a1,$zero #else put (a0) in v0

done:

jr $ra

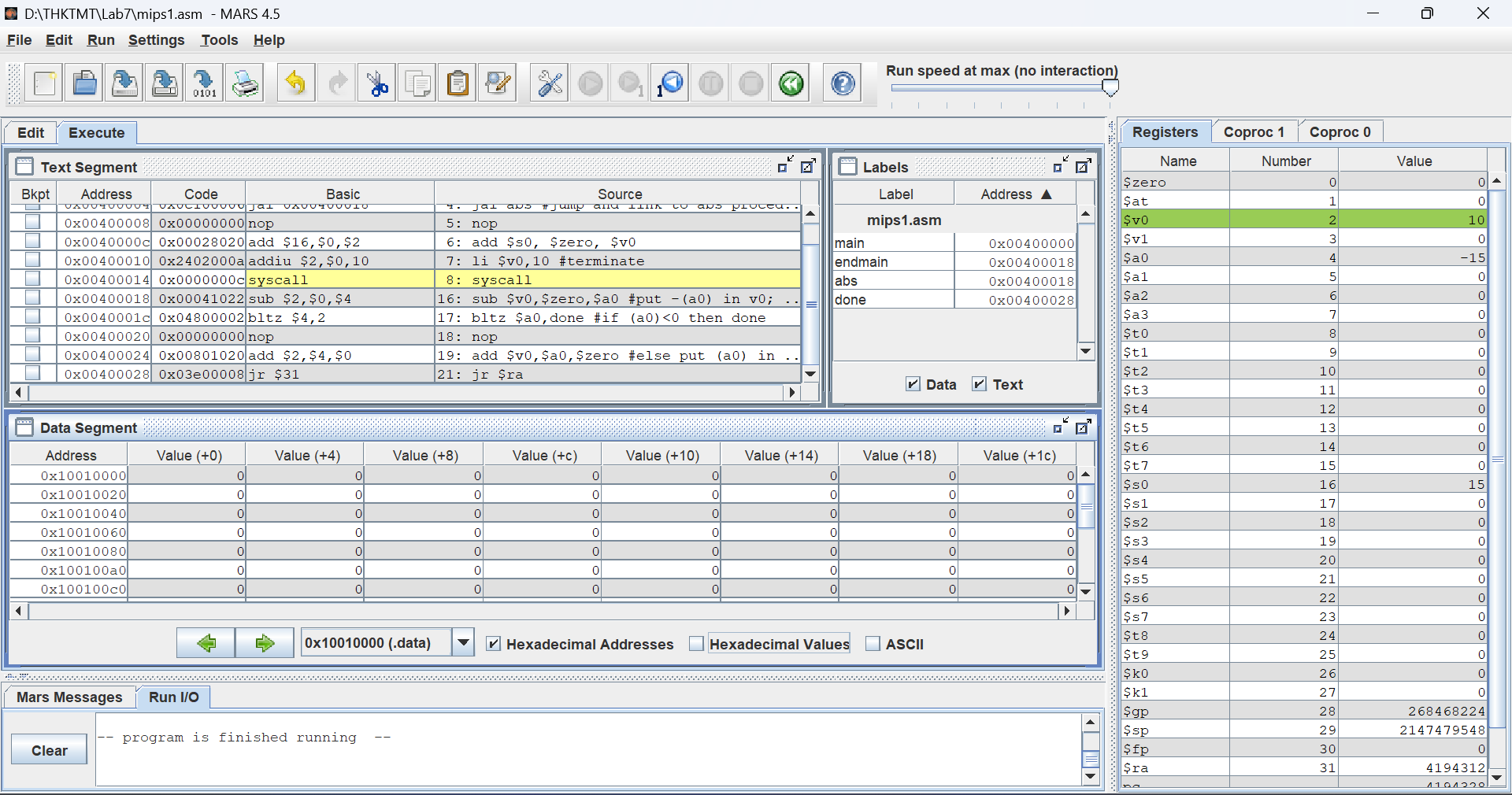
* **Nhận xét**

-Khi thực hiện lệnh jal abs (có địa chỉ lệnh là 0x00400004), thanh ghi $ra gán địa chỉ của câu lệnh tiếp theo là 0x00400008, thanh ghi pc gán địa chỉ của nhãn abs là 0x00400018.



-Khi thực hiện lệnh jr $ar, gán thanh ghi pc bằng giá trị của thanh ghi $ra (0x00400008) để thực hiện cậu lệnh tiếp theo sau khi kết thúc thủ tục abs.

-Kết quả sau chương trình



$s0 = 15 = abs($a0)

* Kết quả đúng

**Assignment 2**

* **Code**

.text

main: li $a0,5 #load test input

li $a1,2

li $a2,8

jal max #call max procedure

add $s0,$v0,$zero

li $v0,10

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

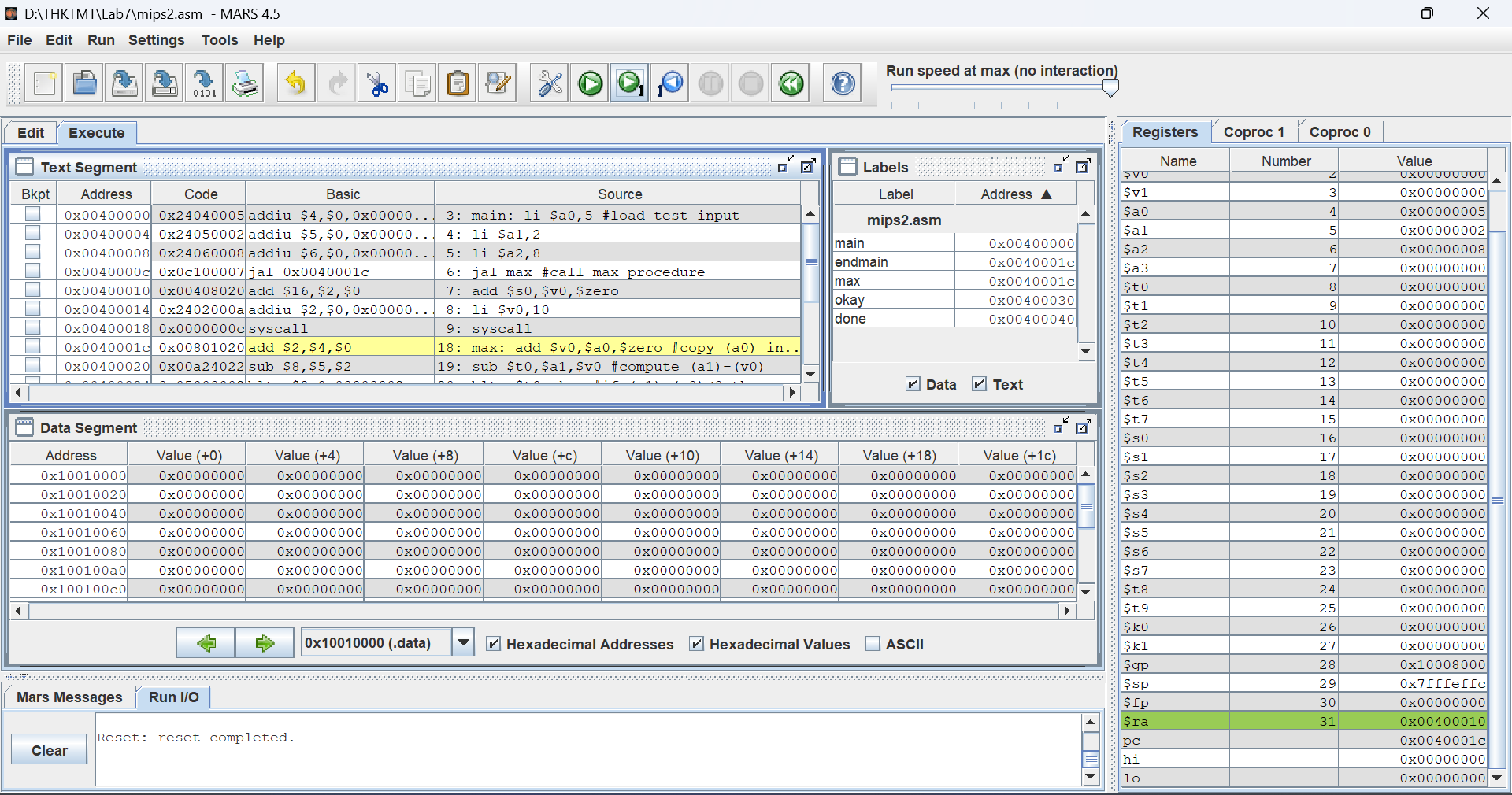
add $v0,$a2,$zero #else (a2) is largest overall

done: jr $ra #return to calling program

* **Nhận xét**

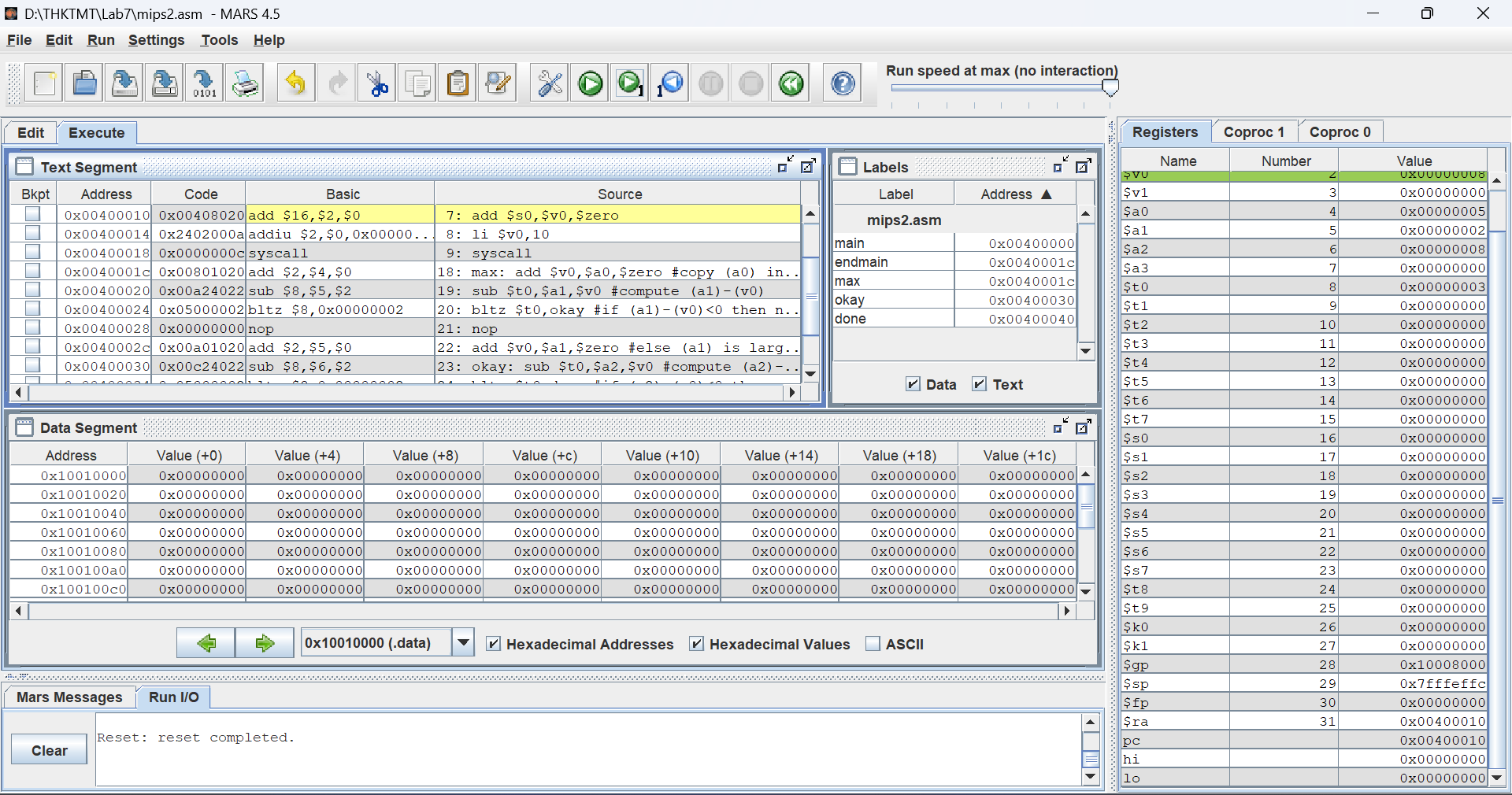
jal max #call max procedure

-Gọi ra thủ tục max. Khi thực hiện lệnh jal max (có địa chỉ lệnh là 0x0040000c), thanh ghi $ra gán địa chỉ của câu lệnh tiếp theo là 0x00400010, thanh ghi pc gán địa chỉ của nhãn abs là 0x0040001c.

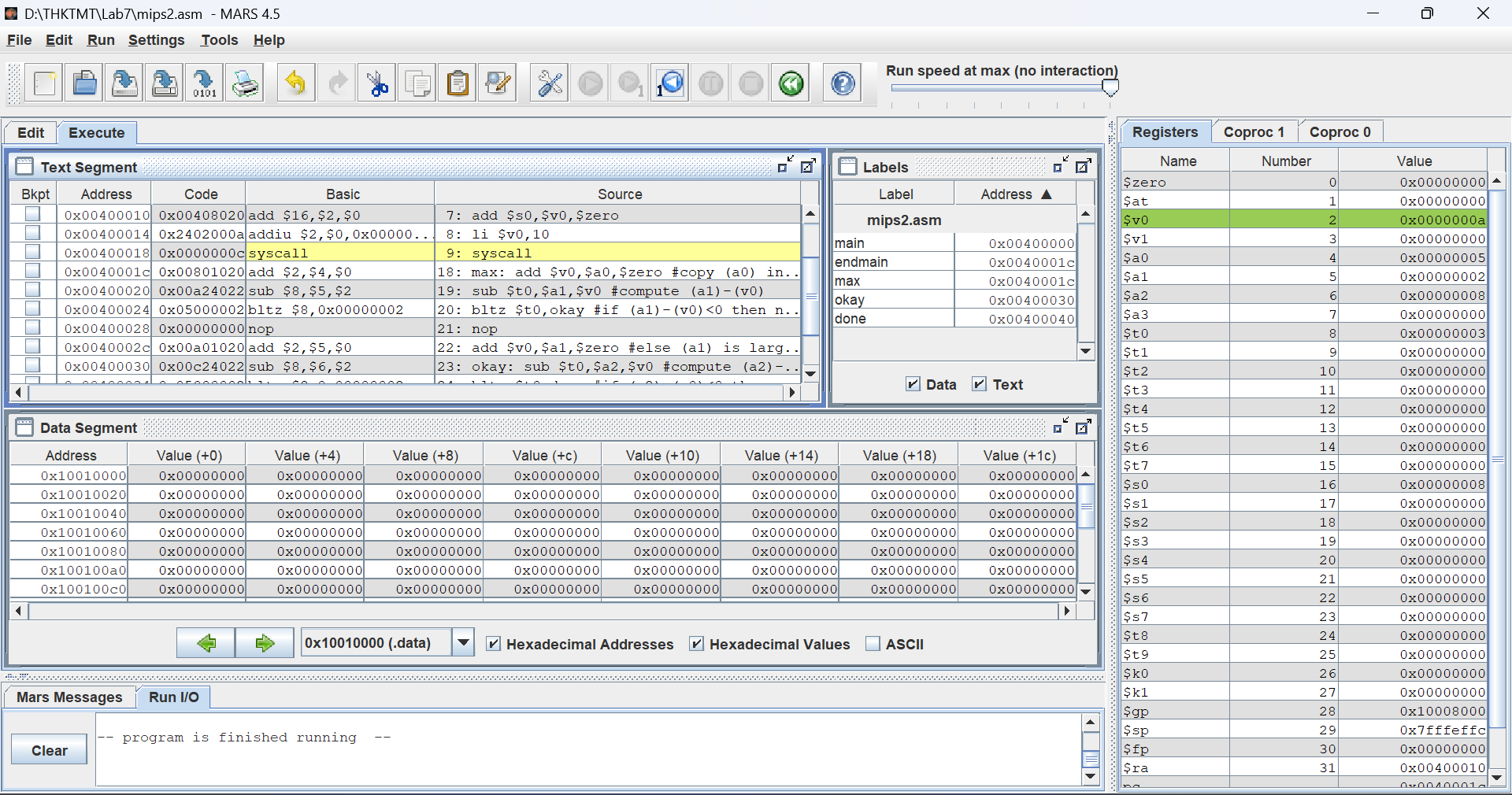


done: jr $ra #return to calling program

-Khi thực hiện lệnh jr $ar, gán thanh ghi pc bằng giá trị của thanh ghi $ra (0x00400010) để thực hiện cậu lệnh tiếp theo sau khi kết thúc thủ tục abs.



-Kết quả của chương trình



Với input $a0 = 5, $a1 = 2, $a2 = 8. Kết quả trả ra được lưu tại $s0 = 8.

* Kết quả đúng

**Assignment 3**

* **Code**

.text

li $s0,7

li $s1,-3

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

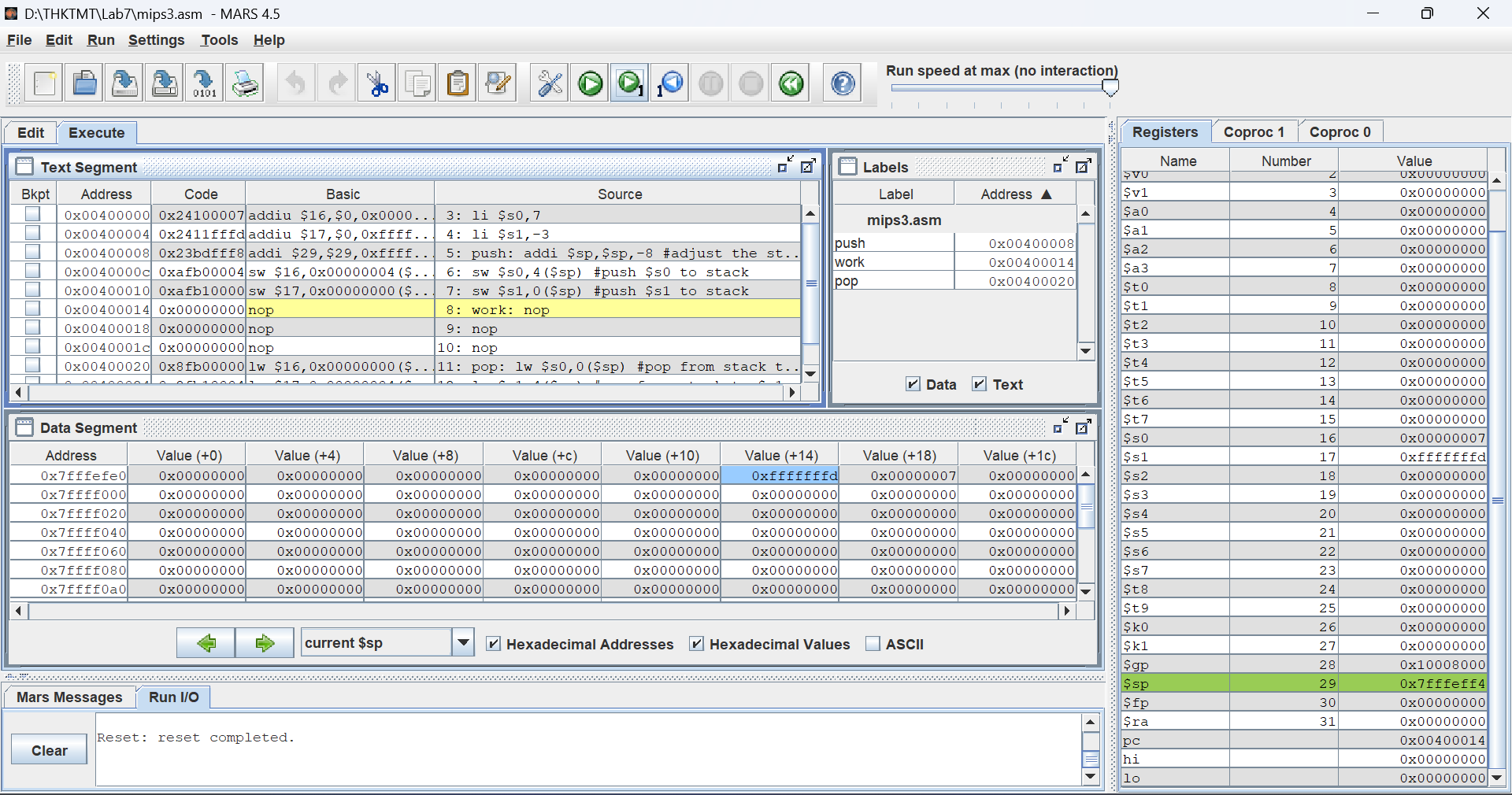
pop: lw $s0,0($sp) #pop from stack to $s0

lw $s1,4($sp) #pop from stack to $s1

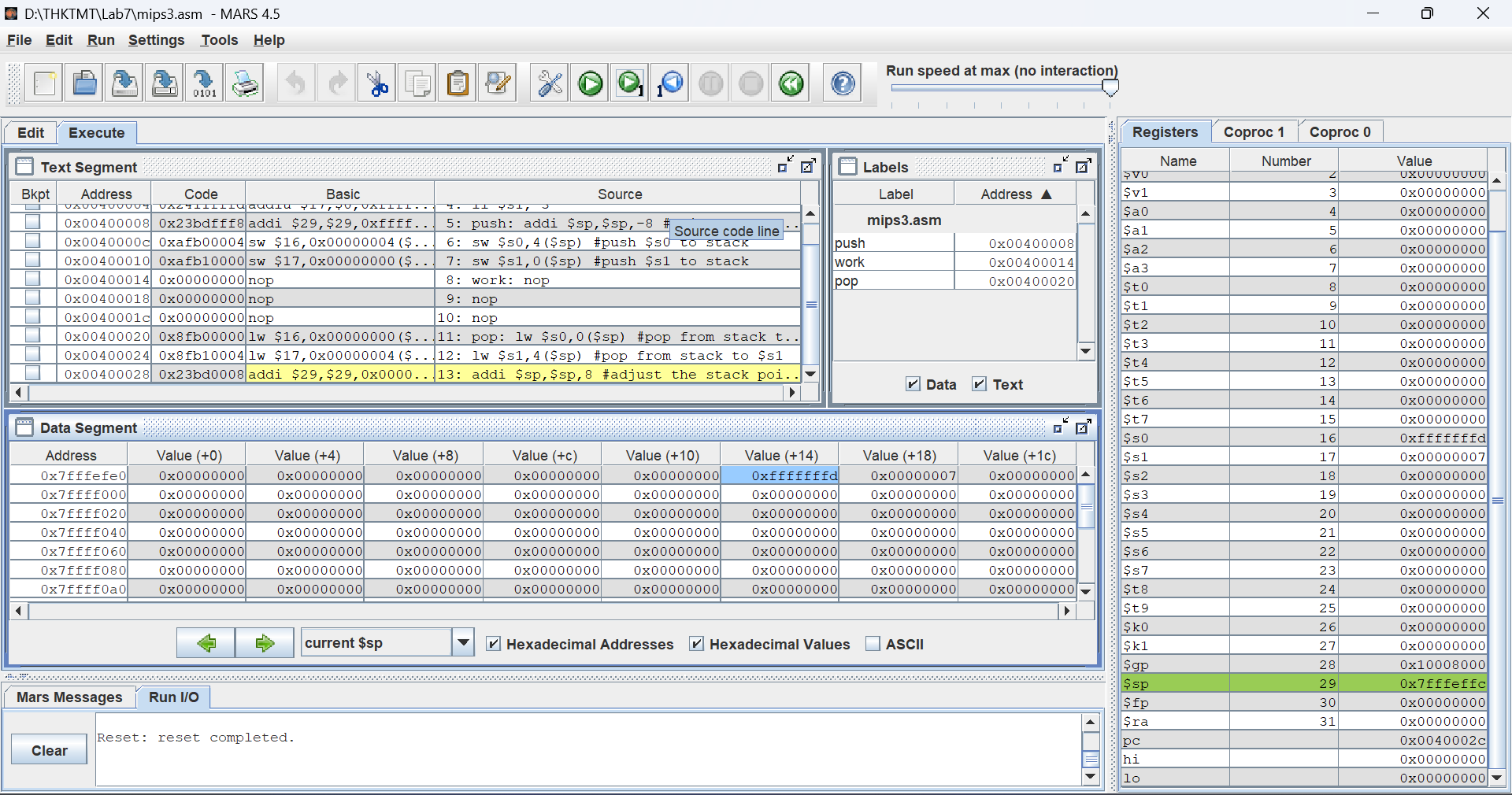
addi $sp,$sp,8 #adjust the stack pointer

* **Nhận xét**

**-**Sau lệnh addi ở nhãn push thanh ghi $sp giảm đi 8 byte tức là có sự cấp phát bộ nhớ stack. Sau đó lưu các giá trị $s0 vào $sp + 4, và $s1 vào $sp

****

**-**Ở nhãn pop, gán giá trị của bộ nhớ có địa chỉ bằng $sp vào $s0, gán giá trị của bộ nhớ có địa chỉ bằng $sp + 4 vào $s1. Lệnh addi $pc + 8 giúp giải phóng stack, trả về đỉnh stack.



-Kết quả: giá trị 2 thanh ghi $s0, $s1 đổi chỗ cho nhau thể hiện FILO của stack.

**Assignment 4**

* **Code**

.data

Message: .asciiz "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:

#----------------------------------------------------------------------

#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 $a0,3 #load test input N

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:

* **Nhận xét**

**-** Khi thực hiện lệnh jal WARP (có địa chỉ lệnh là 0x00400000), thanh ghi $ra gán địa chỉ của câu lệnh tiếp theo là 0x00400004, thanh ghi pc gán địa chỉ của nhãn WARP là 0x00400020.

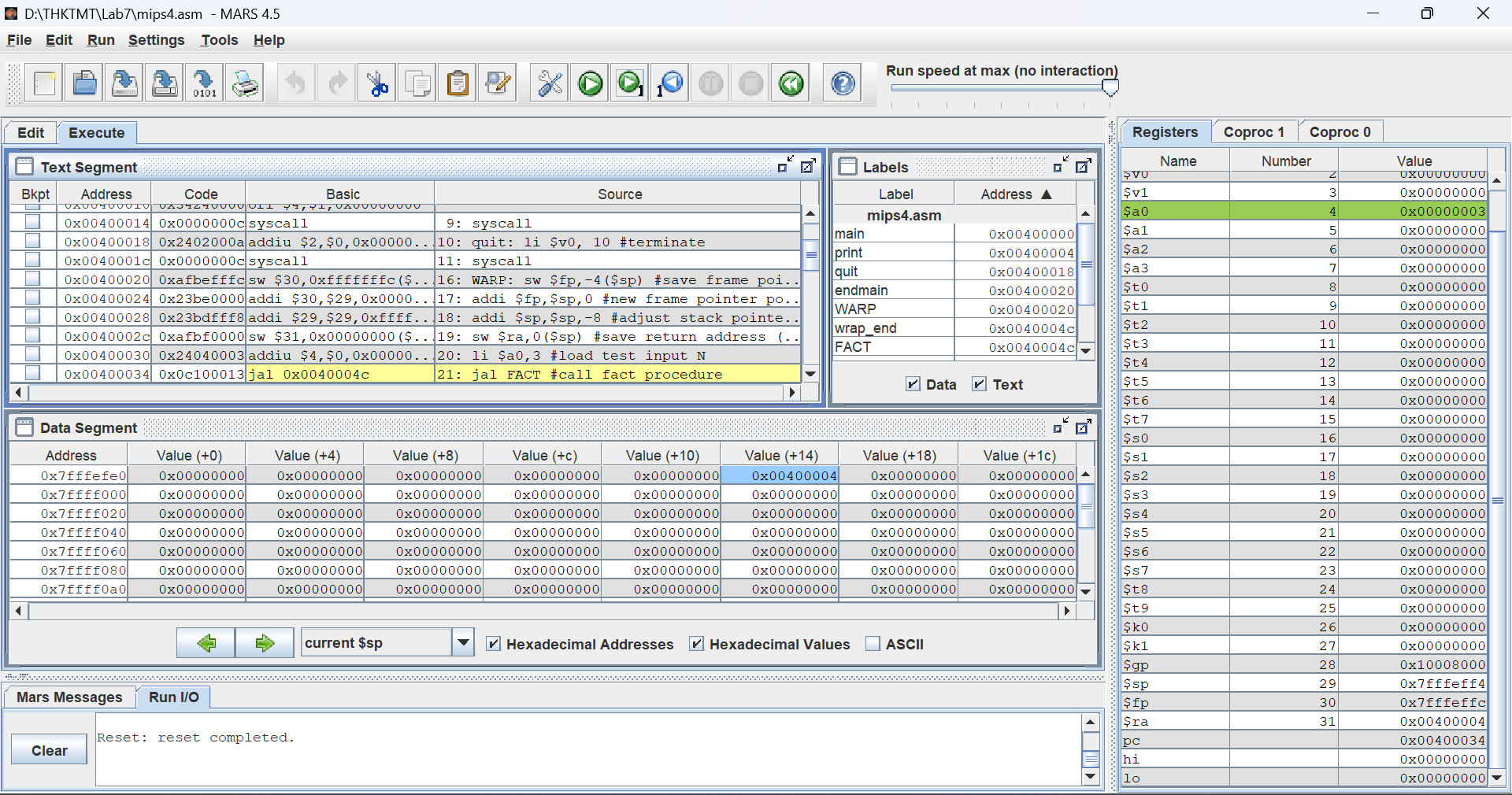
WARP: sw $fp,-4($sp) #save frame pointer (1)

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

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

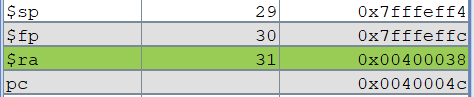
sw $ra,0($sp) #save return address (4)

li $a0,3 #load test input N

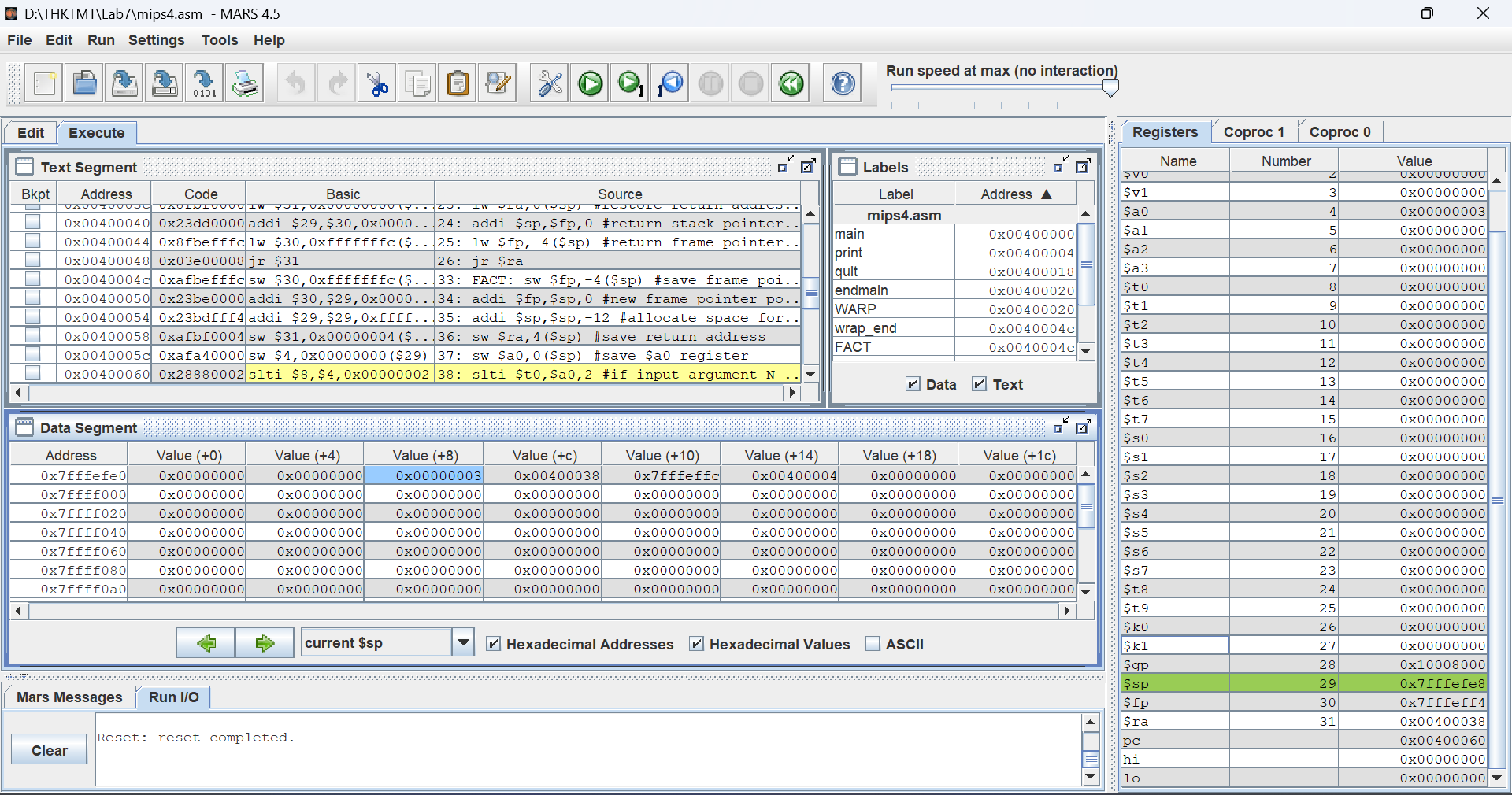
-Tại nhãn WARP, lưu giá trị ban đầu của $fp tại $sp – 4, $fp = $sp = 0x7fffeffc để lưu lại giá trị đỉnh stack. Giảm $sp 8 byte để cấp phát bộ nhớ. Lưu $ra để lưu lại địa chỉ của câu lệnh tiếp theo khi kết thúc WARP, và tránh làm mất địa chỉ này khi thực hiện thủ tục con trong thủ tục WARP. 

jal FACT #call fact procedure

-Gọi thủ tục FACT. Khi thực hiện lệnh jal FACT (có địa chỉ lệnh là 0x00400034), thanh ghi $ra gán địa chỉ của câu lệnh tiếp theo là 0x00400038, thanh ghi pc gán địa chỉ của nhãn FACT là 0x0040004c.

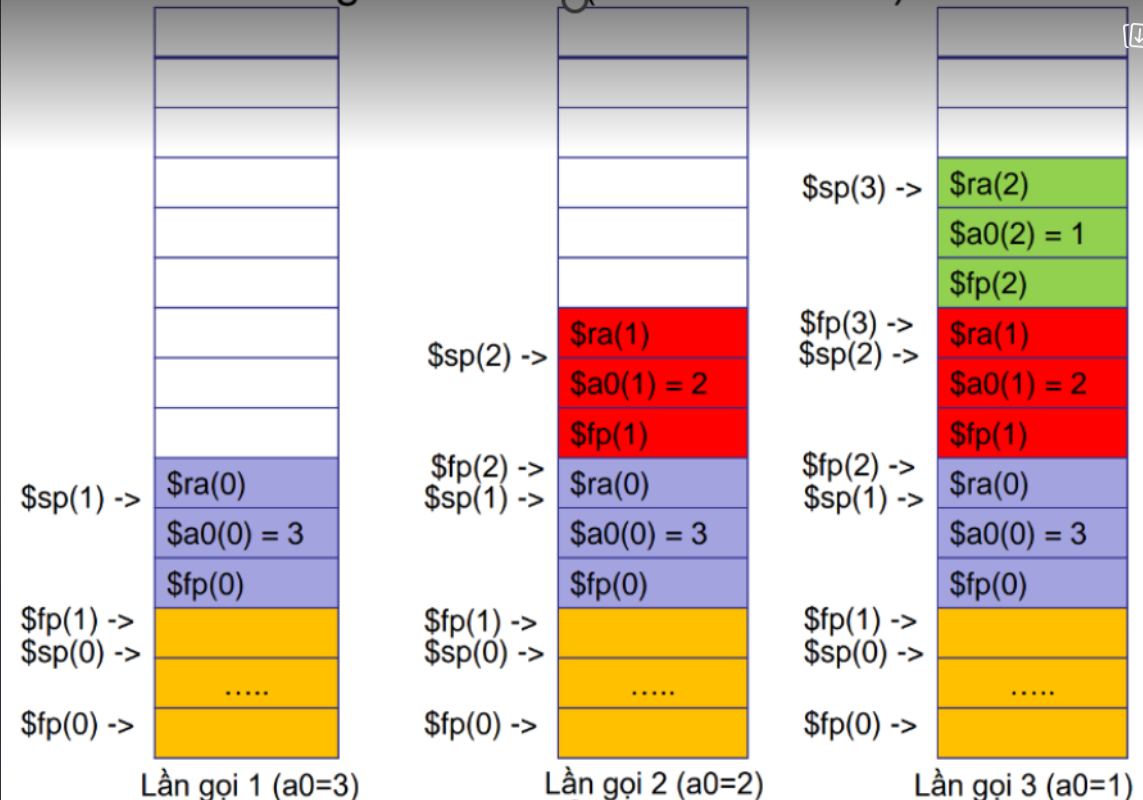


-Tại nhãn FACT, lưu giá trị ban đầu của $fp tại $sp – 4, $fp = $sp = 0x7fffeff4 để lưu lại giá trị stack cũ. Giảm $sp 12 byte để cấp phát bộ nhớ. Lưu $ra tại $sp + 4, $a0 giá trị đầu vào tại $sp.



-Kiểm tra nếu đầu vào >= 2 thì thực hiện đệ quy gọi thủ tục con FACT. Sau khi điều kiện kiểm tra sai thi kết thúc gọi đệ quy, gán $v0 = 1, nhảy đến nhãn done. Ta được bảng giá trị của ngăn xếp như sau:

|  |  |
| --- | --- |
| 0x7fffeffc |  |
| 0x7fffeff8 | $fp = 0x00000000 |
| 0x7fffeff4 | $ra = 0x00400004 |
| 0x7fffeff0 | $fp = 0x7fffeffc |
| 0x7fffefec | $ra = 0x00400038 |
| 0x7fffefe8 | $a0 = 0x00000003 |
| 0x7fffefe4 | $fp = 0x7fffeff4 |
| 0x7fffefe0 | $ra = 0x00400080 |
| 0x7fffefdc | $a0 = 0x00000002 |
| 0x7fffefd8 | $fp = 0x7fffefe8 |
| 0x7fffeffd4 | $ra = 0x00400080 |
| 0x7fffefd0 | $a0 = 0x00000001 |



-Tại done

lw $ra,4($sp) #restore return address

lw $a0,0($sp) #restore a0

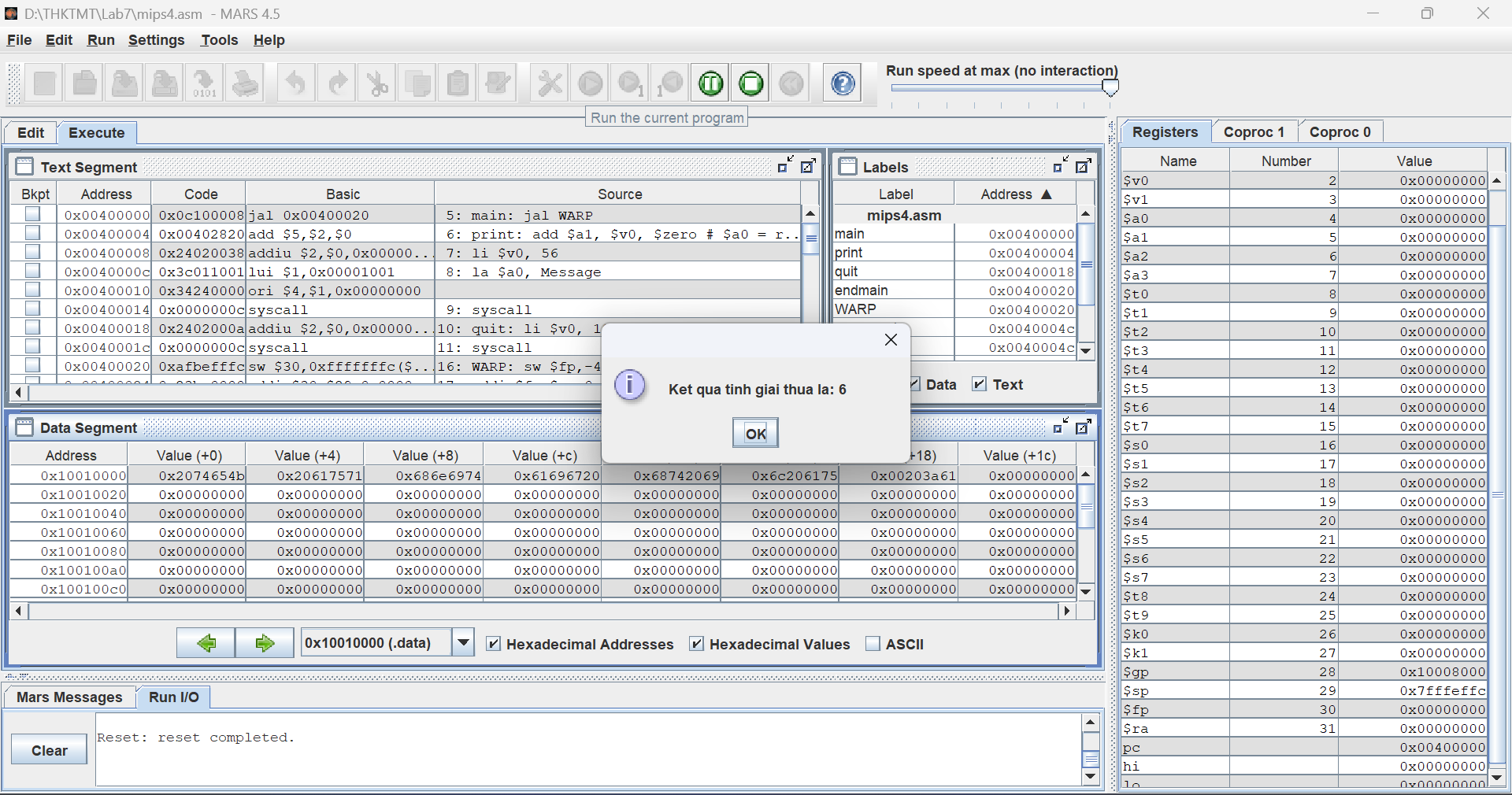
addi $sp,$fp,0 #restore stack pointer

lw $fp,-4($sp) #restore frame pointer

jr $ra #jump to calling

lấy ra giá trị $ra, $a0 tại stack, gán $sp = $fp để quay trở lại stack trước đó, lấy giá trị $fp tại $sp – 4 để lấy ra stack tiếp theo. Và thực hiện lệnh jr $ra (0x00400080)

-Load $v1 nhân $v0 với $v1. Rồi thực hiện như trên cho đến khi kết thúc thủ tục FACT ngoài cùng của đệ quy.

-Kết quả 

* Đúng.

**Exercise 5**

* **Code**

.data

msg1: .asciiz "Largest:"

msg2: .asciiz "\nSmallest:"

.text

main:

# Khởi tạo giá trị từ thanh ghi $s0 đến thanh ghi $s7

li $s0, 5

li $s1, 2

li $s2, -2

li $s3, 9

li $s4, -12

li $s5, 20

li $s6, -3

li $s7, -15

jal produce

nop

li $v0, 4

la $a0, msg1

syscall

add $a0, $t0, $zero

li $v0,1

syscall

li $v0,11

li $a0, ','

syscall

add $a0, $a1, $zero

li $v0,1

syscall

li $v0, 4

la $a0, msg2

syscall

add $a0, $t1, $zero

li $v0,1

syscall

li $v0,11

li $a0, ','

syscall

add $a0, $a2, $zero

li $v0,1

syscall

li $v0, 10

syscall

endmain:

swapMax: add $t0,$t3,$zero

add $a1,$t2,$zero

jr $ra

swapMin: add $t1,$t3,$zero

add $a2,$t2,$zero

jr $ra

produce: #find largest vs smallest

add $a3,$sp,$zero # Save address of origin $sp

addi $sp, $sp, -32 # integerS in stack

sw $s1, 0($sp)

sw $s2, 4($sp)

sw $s3, 8($sp)

sw $s4, 12($sp)

sw $s5, 16($sp)

sw $s6, 20($sp)

sw $s7, 24($sp)

sw $ra, 28($sp)

add $t0,$s0,$zero # Max = $s0

add $t1,$s0,$zero # Min = $s0

li $a1, 0 # Index of Max

li $a2, 0 # Index of Min

li $t2, 0 # i = 0

loop:

addi $sp, $sp, 4

lw $t3, -4($sp)

sub $t6, $sp, $a3

beq $t6,$zero, done # If $sp = $fp branch to the 'done'

nop

addi $t2,$t2,1 # i++

sub $t6,$t0,$t3

bltzal $t6, swapMax # If $t3 > Max branch to the swapMax

nop

sub $t6,$t3,$t1

bltzal $t6, swapMin # If $t3 < Min branch to the swapMin

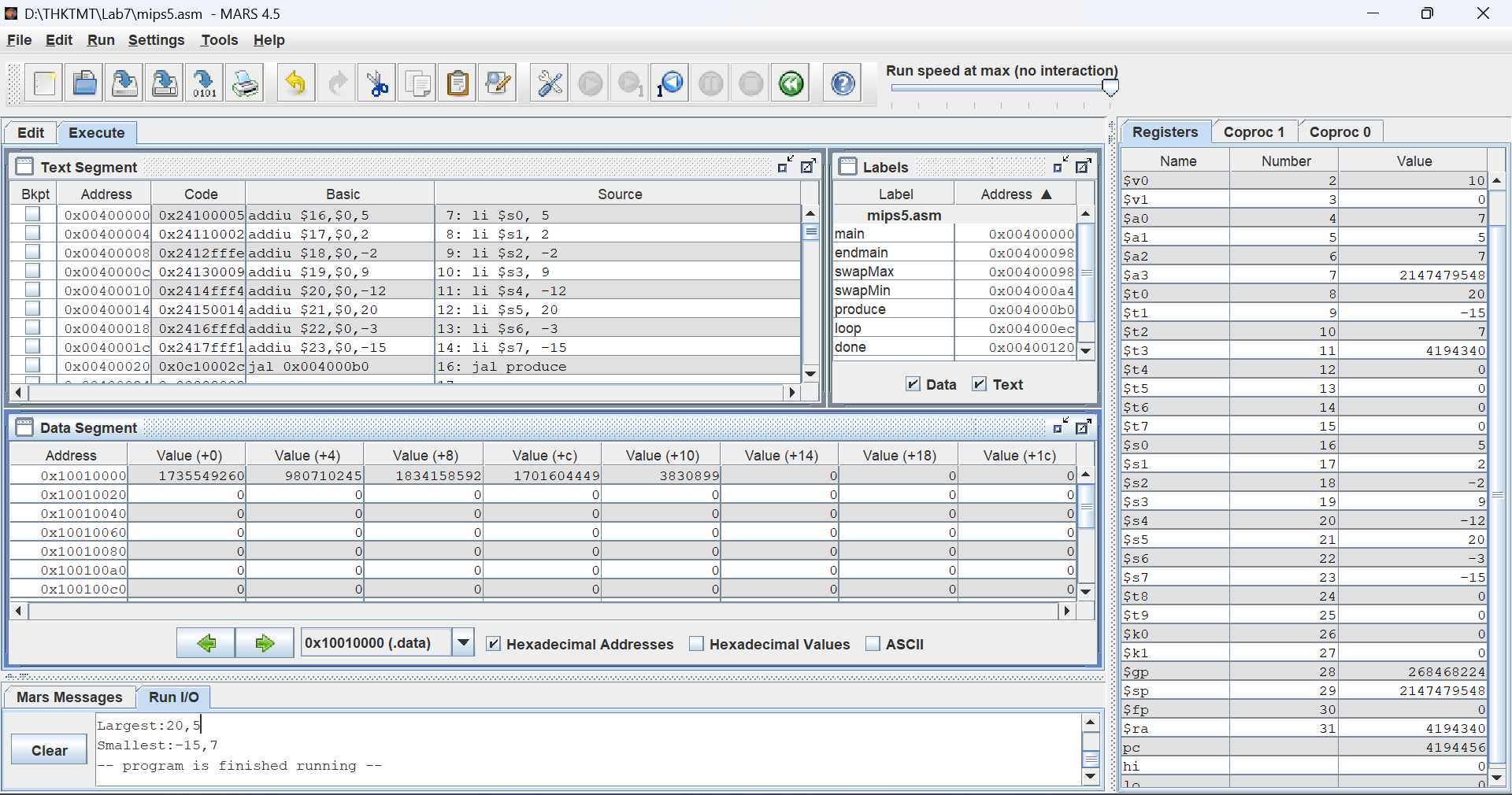
nop

j loop

done:

lw $ra, -4($sp)

jr $ra # Return to calling program

* **Kết quả**
* Kết quả đúng.