Họ và tên : Lê Ngọc Anh Quân

MSSV: 20176852

Bài 4

Source code :

.data

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

.text

main:

jal WARP

print:

add $a1, $v0, $zero # $a1 = 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,$a0in 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) #restoreframe pointer

jr $ra #jump to calling

fact\_end:

Nhật xét câu lệnh

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 $a0,10 #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)

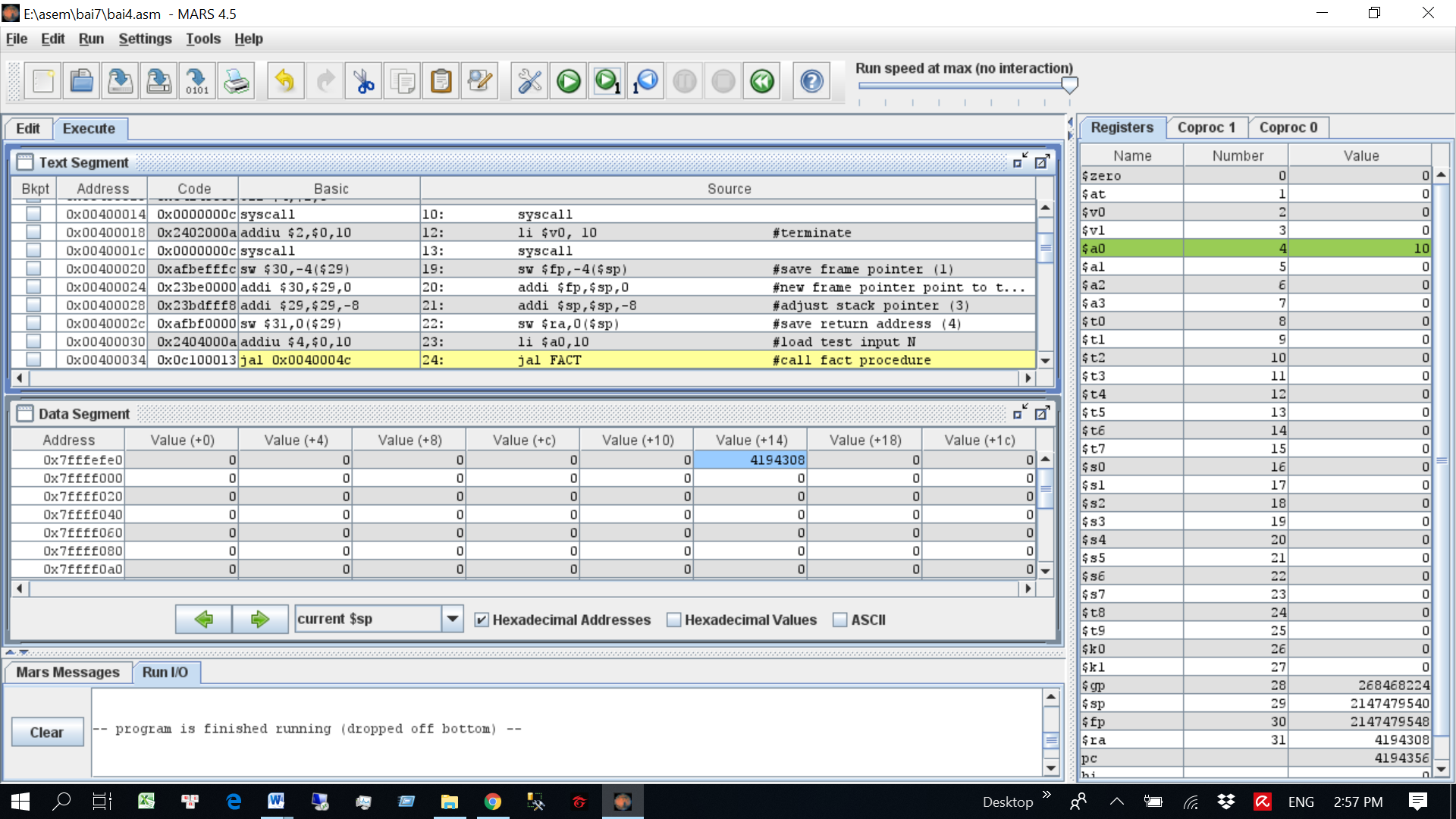
lw $fp,-4($sp) #return frame pointer (7)

jr $ra

wrap\_end:

* Hàm này để khởi tạo stack lưu giá trị thanh ghi $ra và frame pointer để lưu giá trị địa chỉ stack . Sử dụng frame pointer để tiện người dùng truy cập vào địa chỉ stack mình cần .
* Gán $a0 = 3 , ta phải đi tính 3! .

Kết quả :



Hàm FACT:

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,$a0in 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

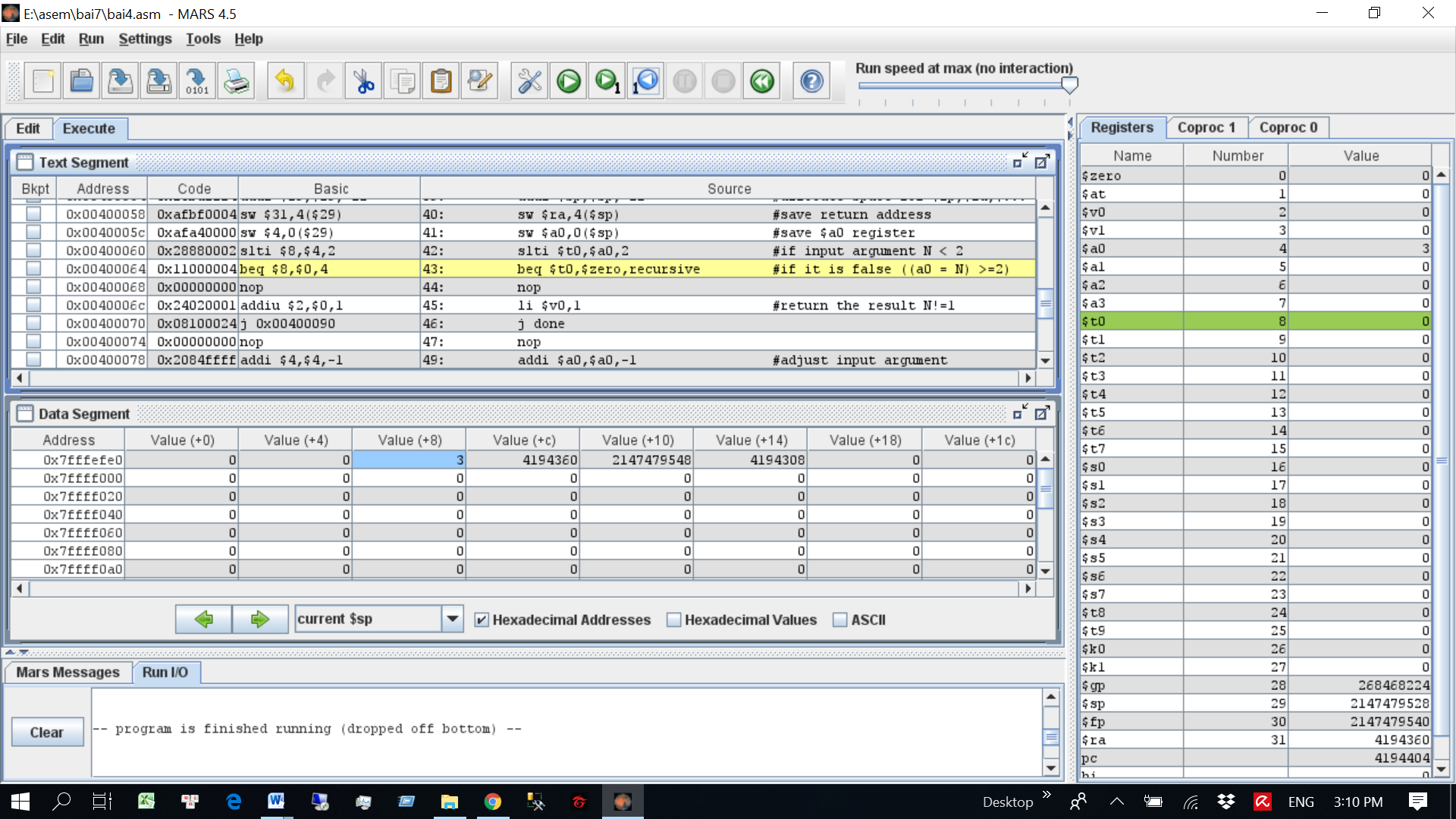
lw $fp,-4($sp) #restoreframe pointer

jr $ra #jump to calling

fact\_end:

* ở stack , tạo ra 3 ô , 1 ô để chứa địa chỉ của thanh ghi $ra , 1 ô chứa địa chỉ thanh gi $fp , 1 ô chứ giá trị của $a0 .
* Mỗi lần recursive , $a0 giảm 1 .

Khi $a0=3 , ta có kết quả sau :



* Nhận thấy ở Value(+14) lưu giá trị thanh ghi $ra ở hàm WARP .Value(+10) lưu giá trị của $fp là con trỏ chỉ đến đỉnh stack . Value(+8) chỉ giá trị của $a0.
* Vì N=3 nên chạy hàm recursive.

Câu lệnh tiếp theo :

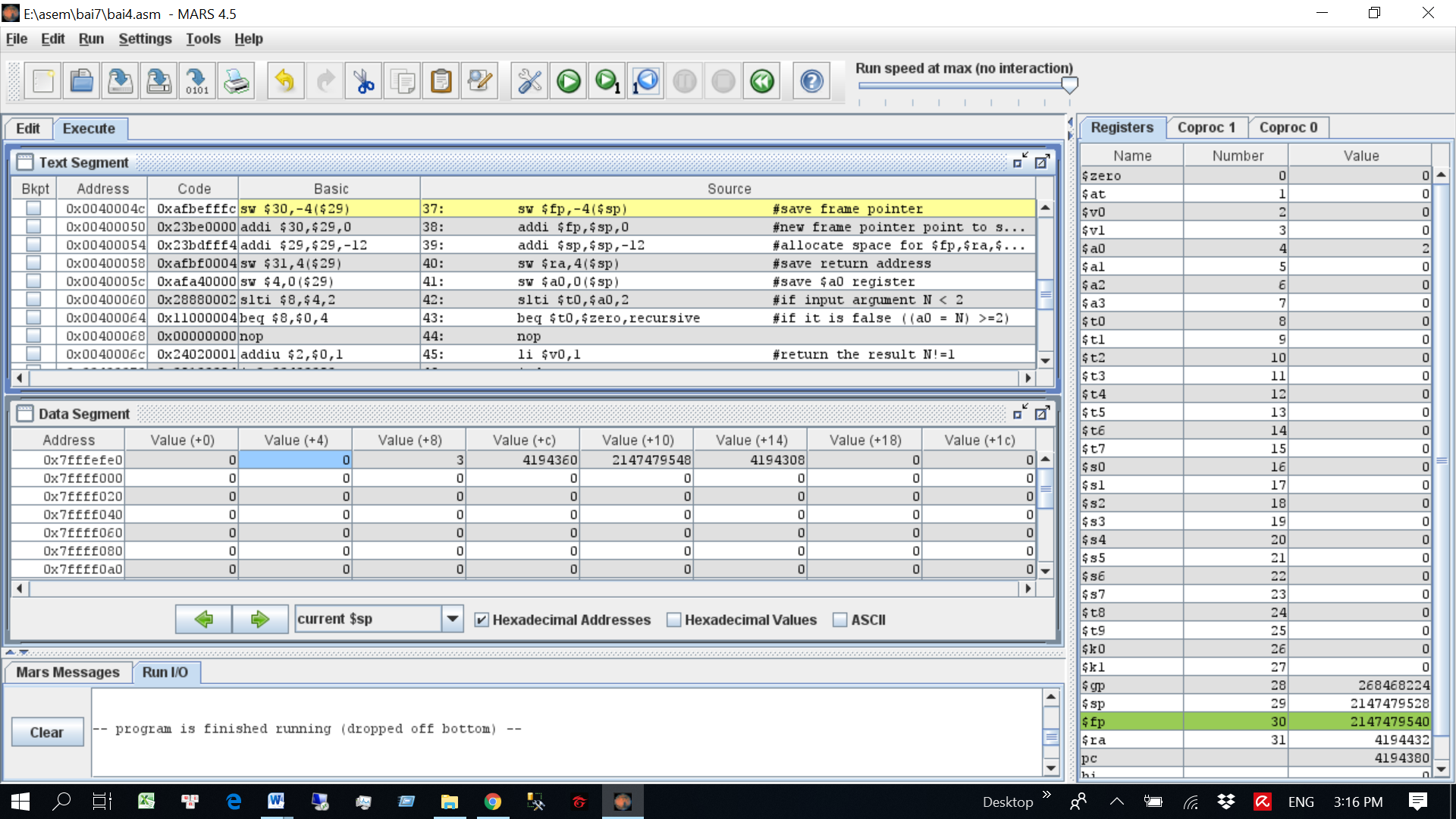
recursive:

addi $a0,$a0,-1 #adjust input argument

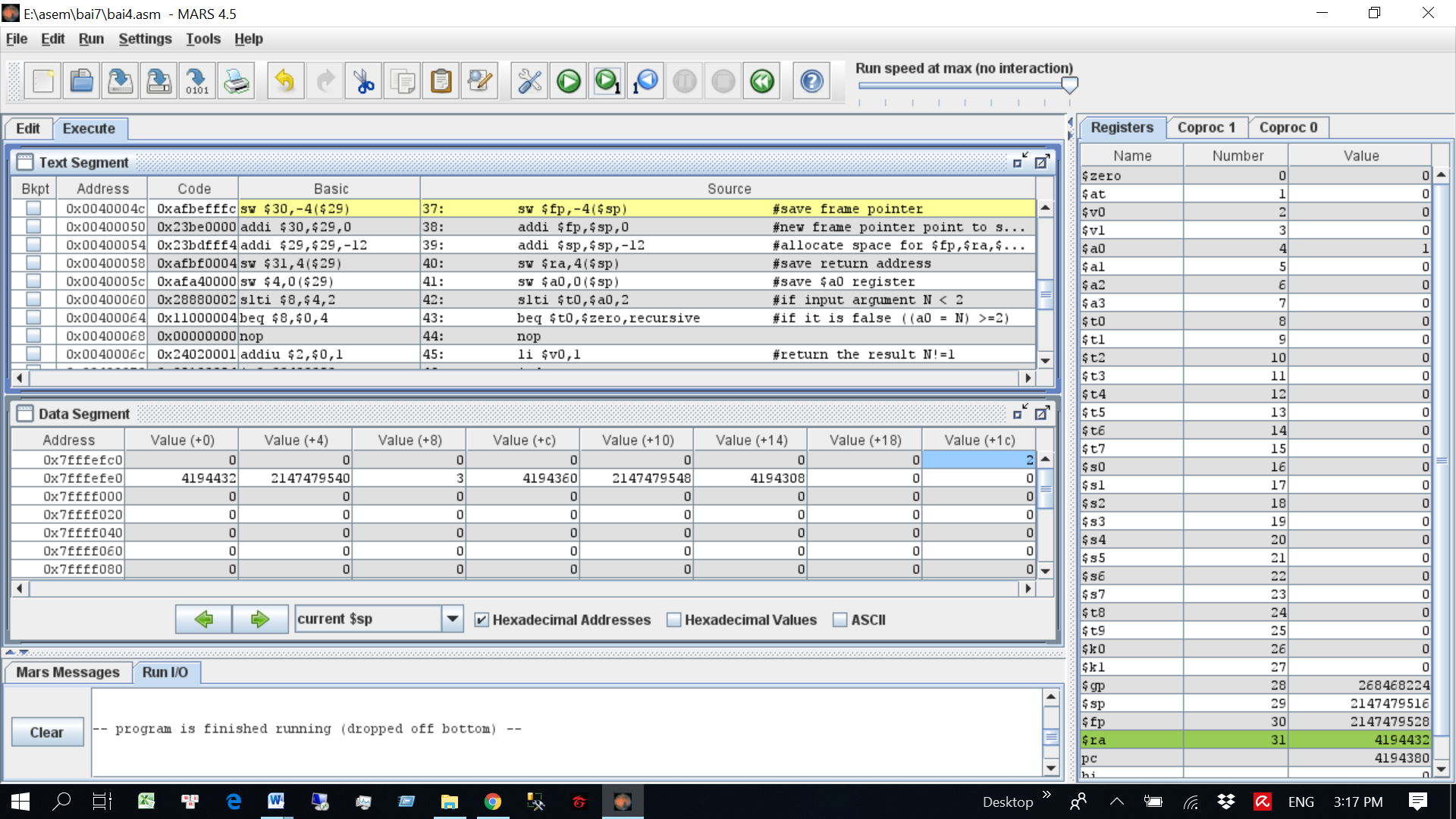
jal FACT #recursive call

* Gán $a0=$a0-1 = 3-1=2 , sau đó chạy lại hàm FACT để thực hiện đệ quy .

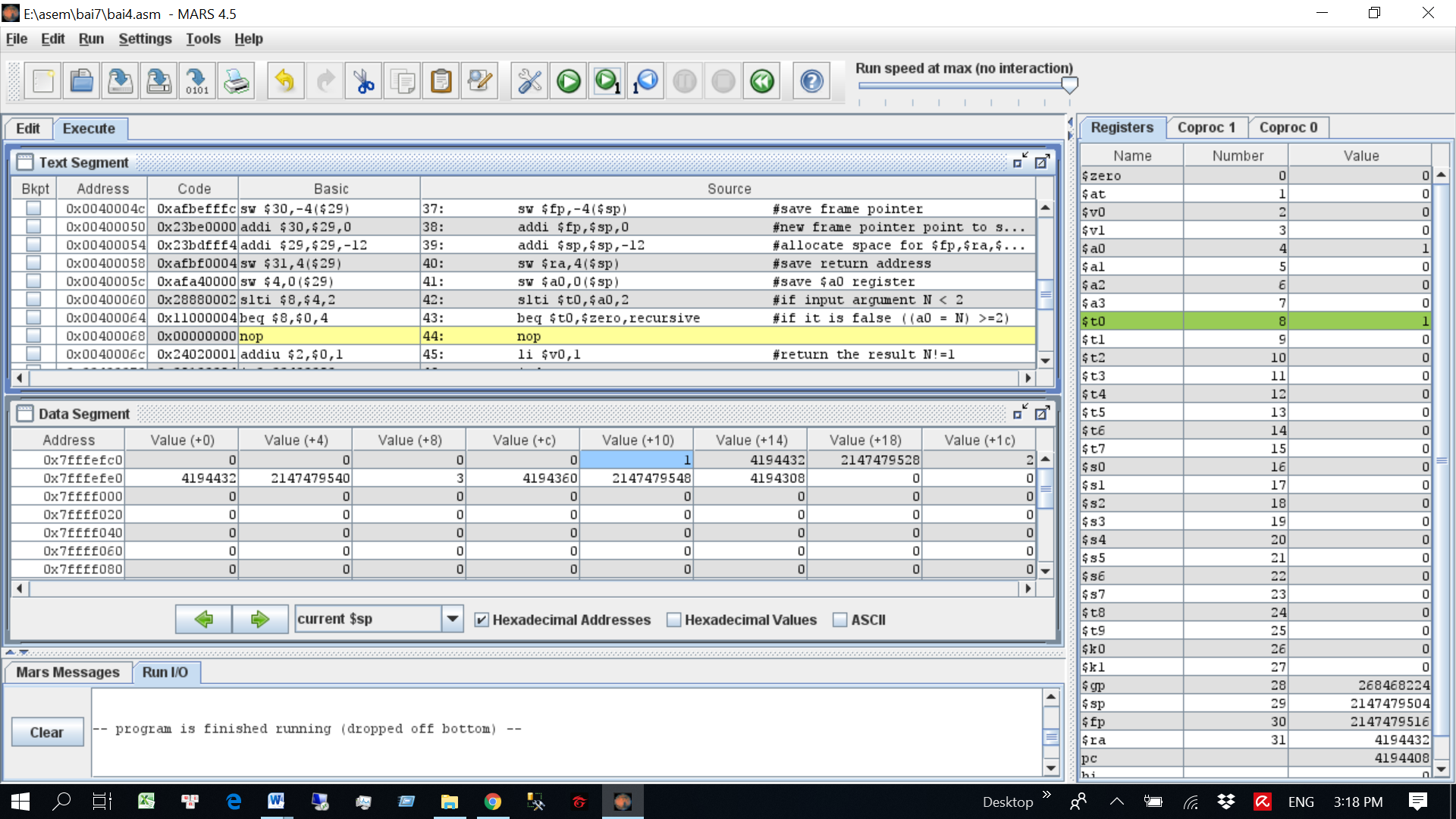
Kết quả Sử dụng lệnh jal nên thanh ghi $ra sẽ chỉ địa chỉ của câu lệnh tiếp theo



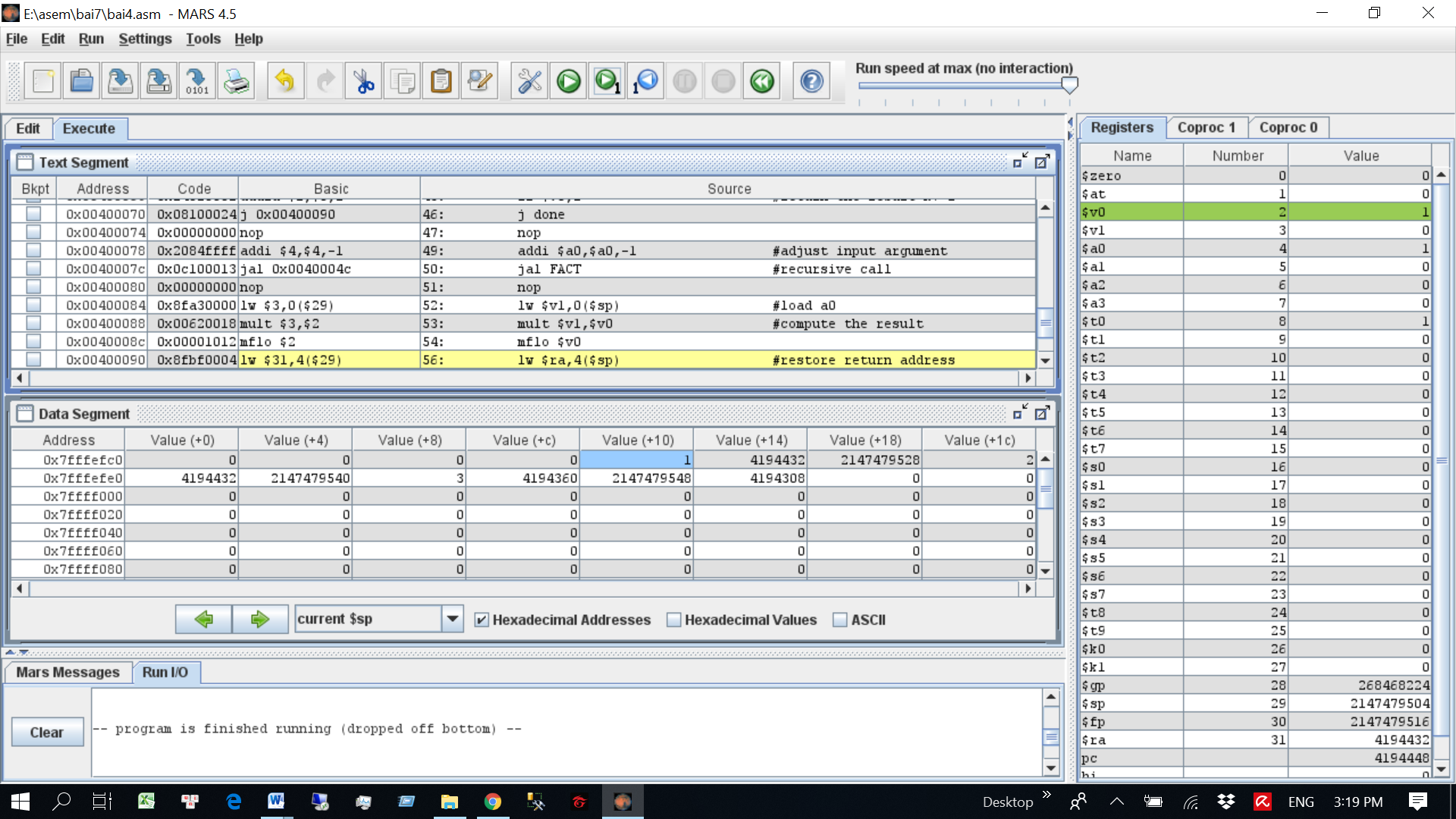
Tương tự ta có sau lần đệ quy khi $a0=2



Tiếp tục giảm $a0=1 , thực hiện lần đệ quy :



* Vì lúc này $a0=1 < 2 nên ta gián $v0=1 rồi nhảy đến hàm done

Kết quả 

Câu lệnh tiếp theo :

done:

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

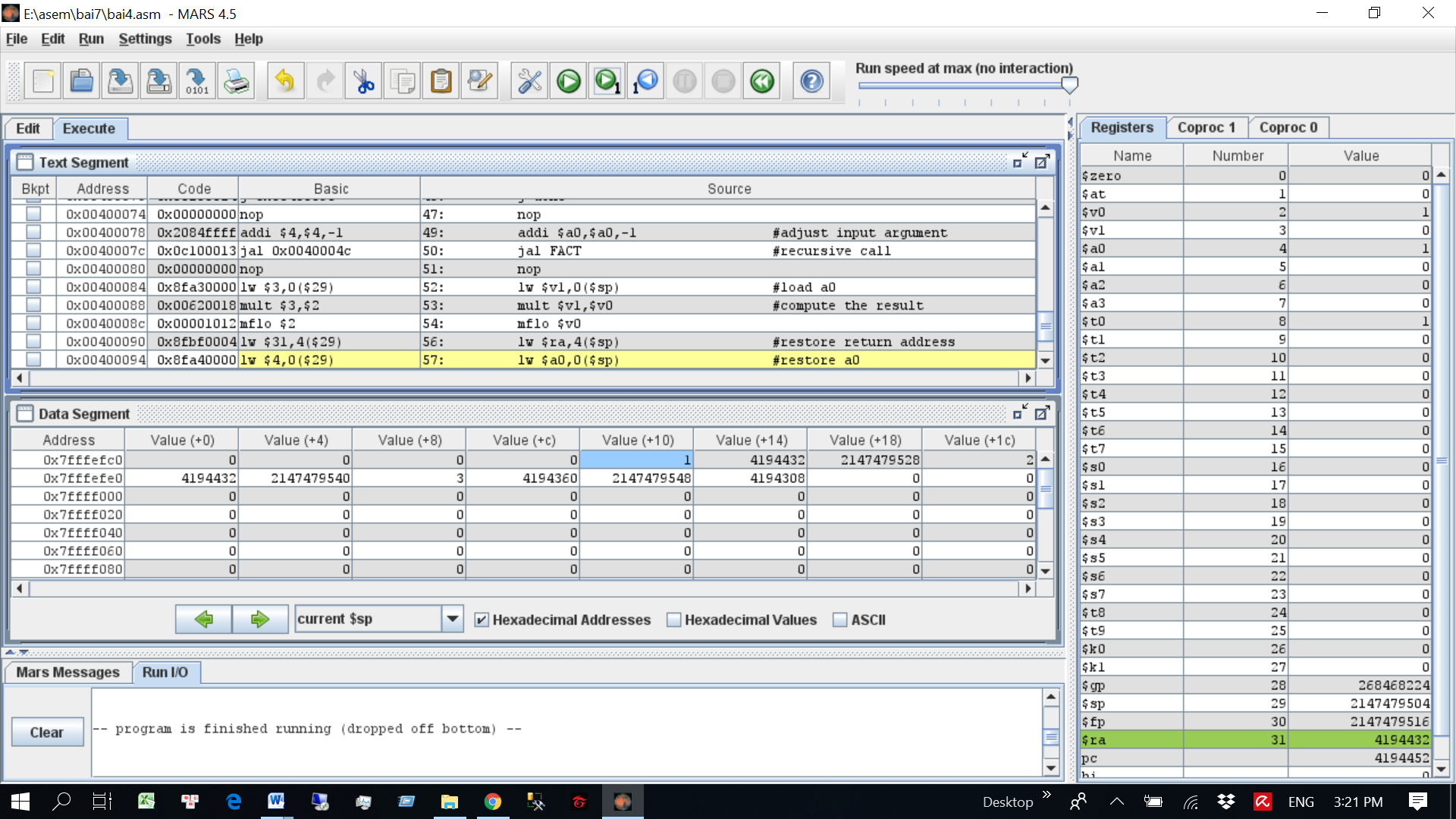
lw $a0,0($sp) #restore a0

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

lw $fp,-4($sp) #restoreframe pointer

jr $ra #jump to calling

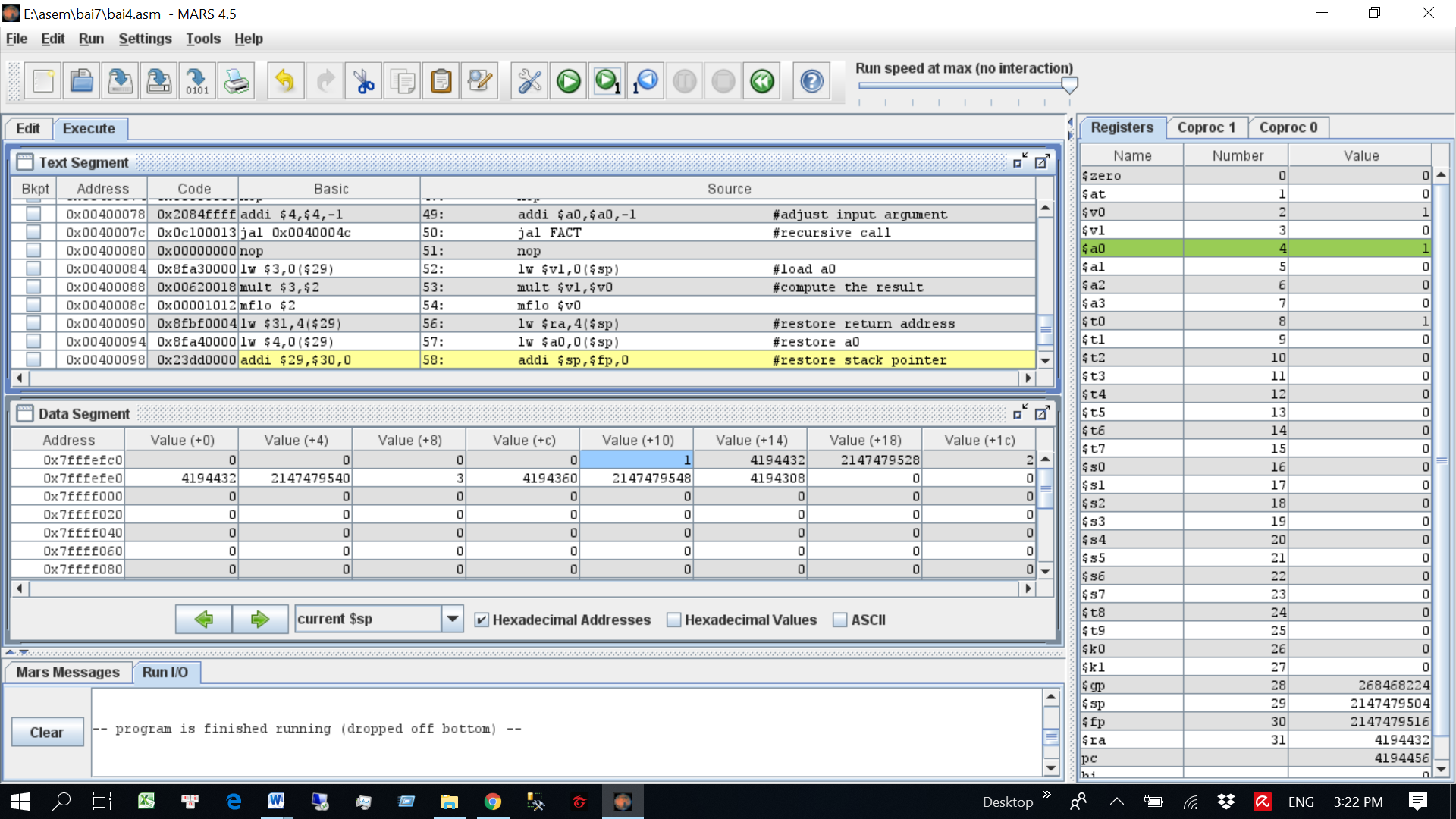
* Câu lệnh lw $ra,4($sp) để lấy địa chỉ $ra được lưu trong stack ngăn thứ 2 .
* Kết quả



Câu lệnh lw $a0,0($sp) #restore a0

* Lấy giá trị $a0 = 1 được lưu trong stack ngăn thứ nhất

Kết quả :

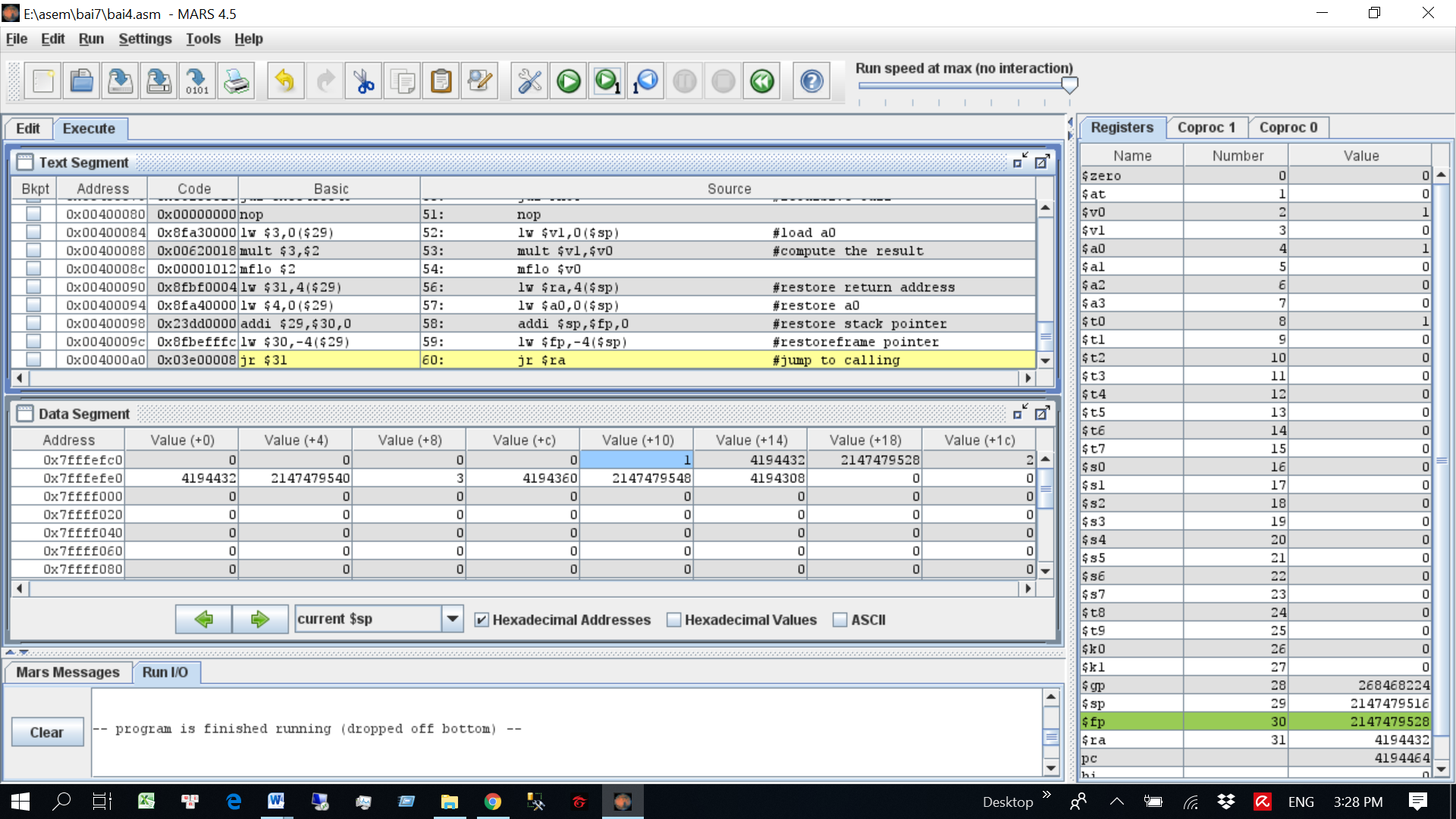


Câu lệnh

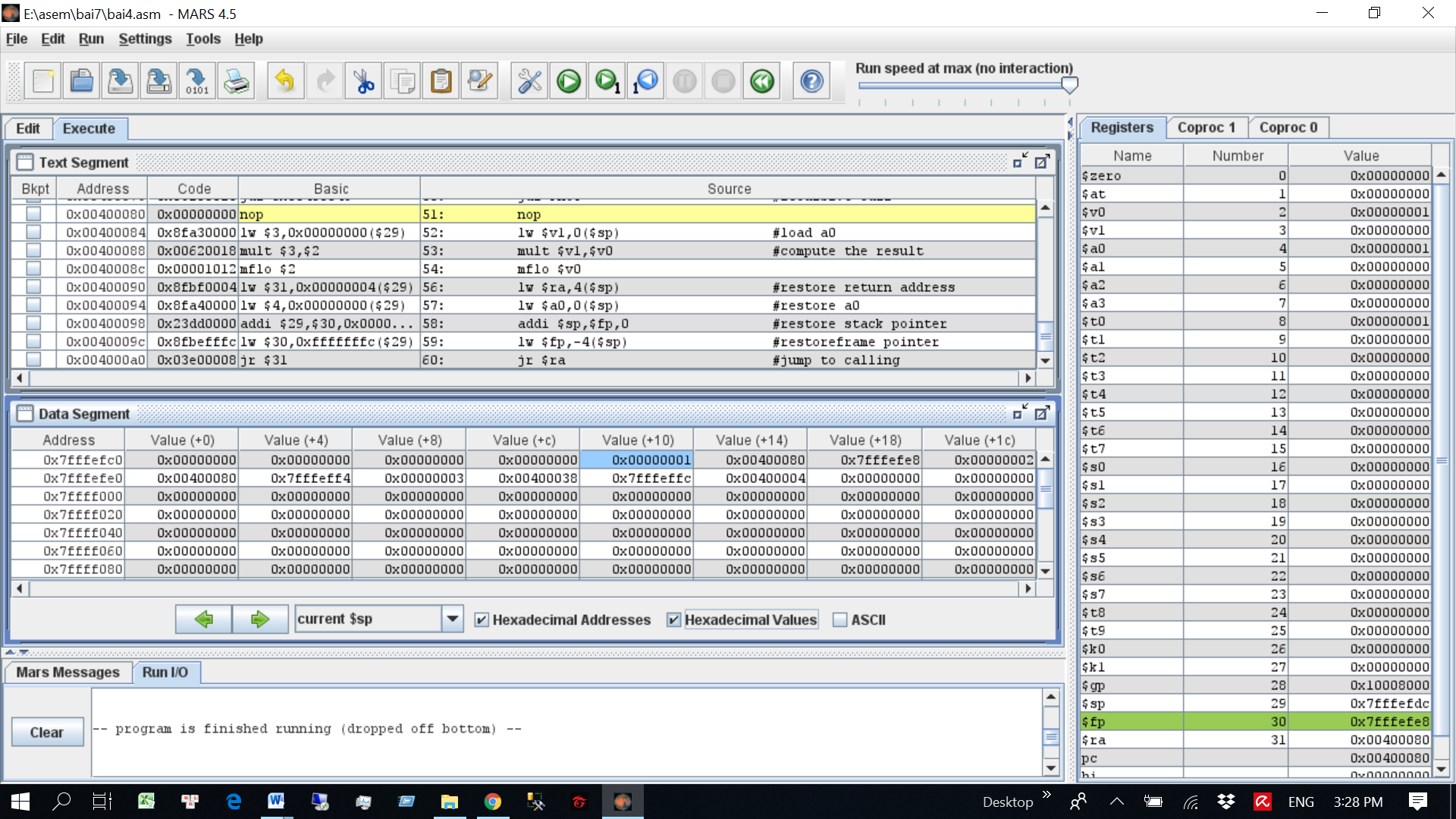
addi $sp,$fp,0 #restore stack pointer

lw $fp,-4($sp) #restoreframe pointer

* Lưu lại con trỏ ngăn xếp bawnngf con trỏ framepointer. Rồi lưu giá trị framepointer mới .
* Kết quả :



Sau khi chạy câu lệnh jal $ra



Nhảy về câu lệnh nop

Tiếp theo :

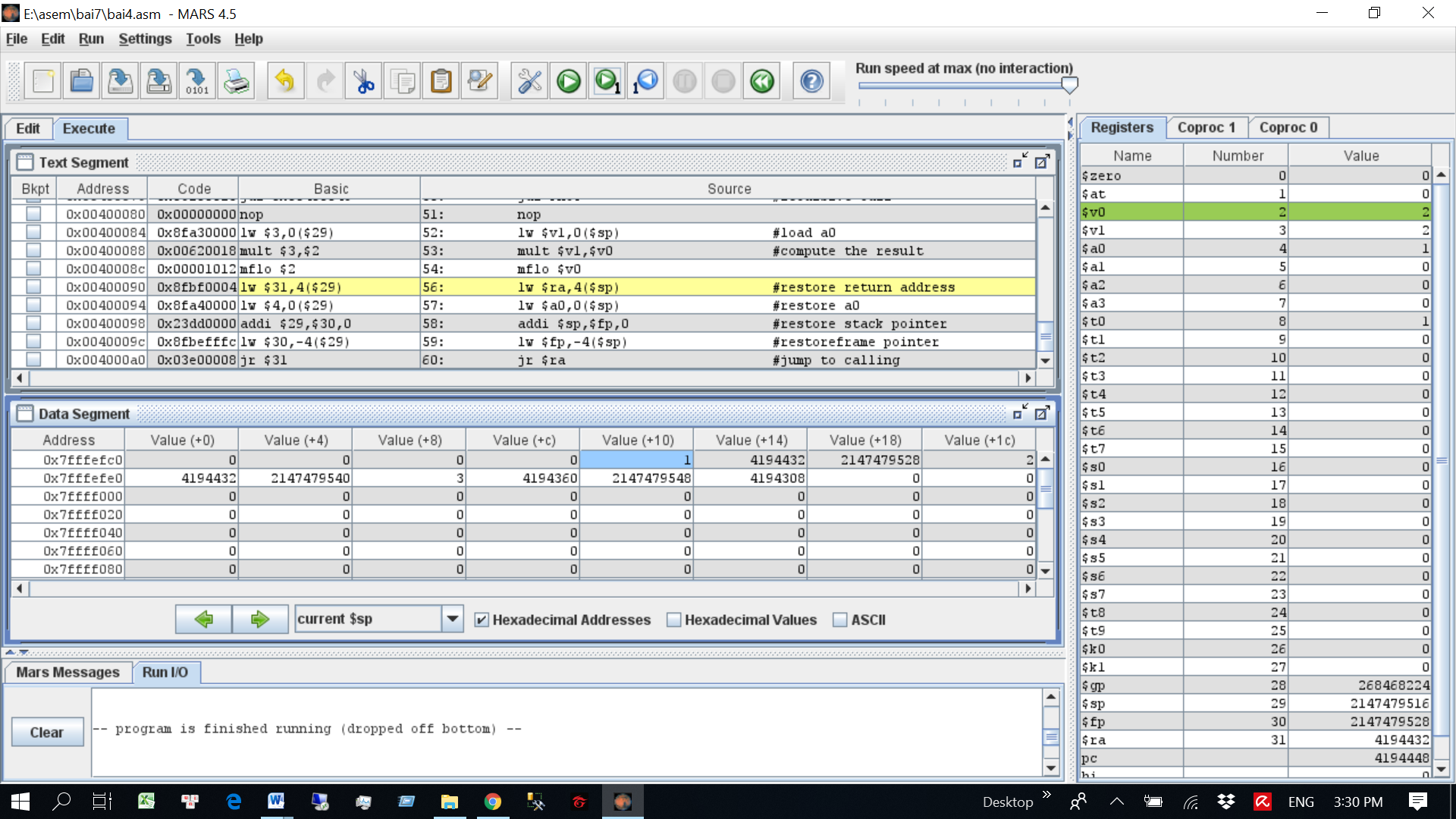
lw $v1,0($sp) #load a0

mult $v1,$v0 #compute the result

mflo $v0

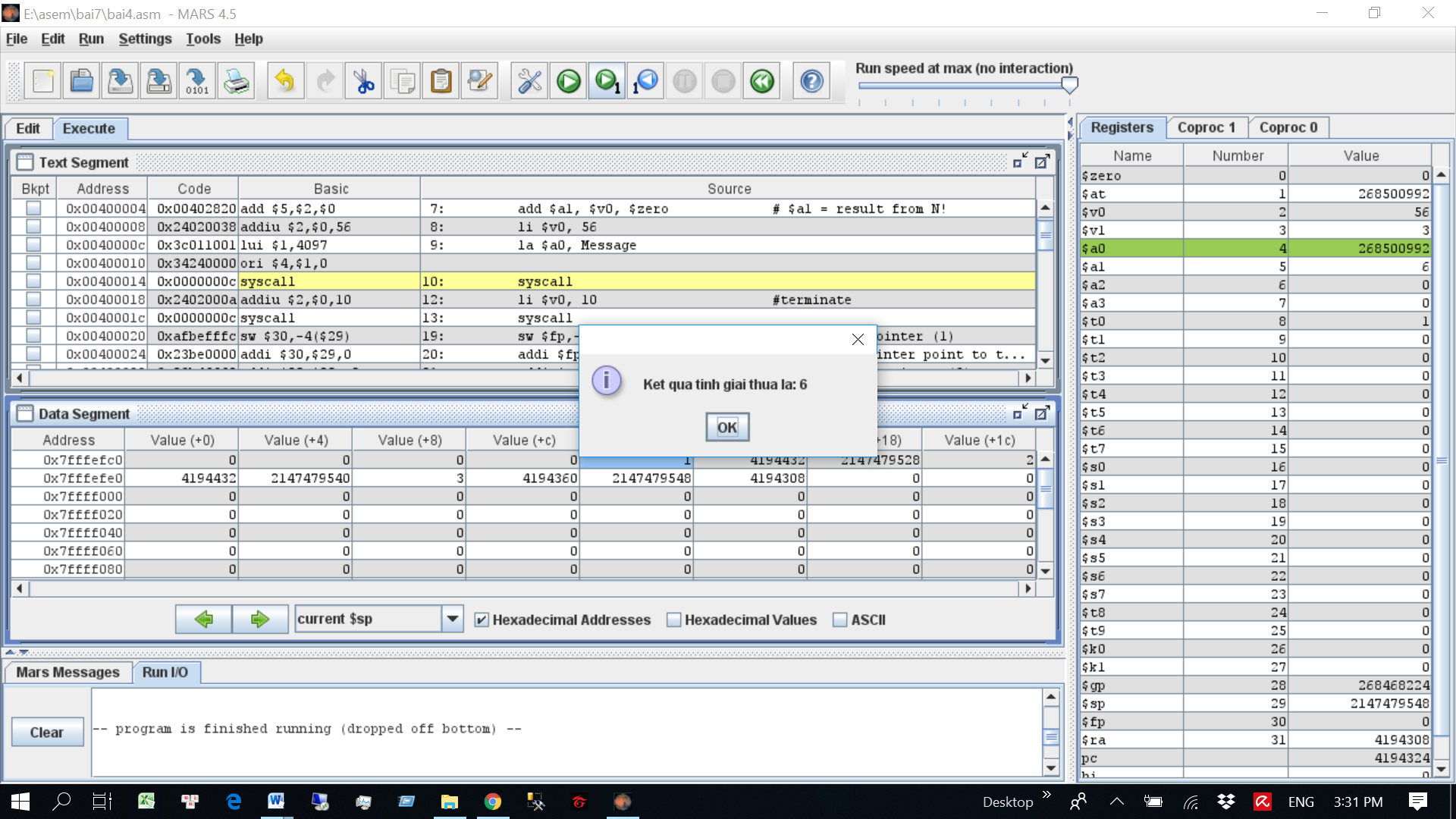
* Lấy $v1 = giá trị của ô đầu trong stack
* Lưu $v0=$v0 x $v1=1x2=2

Kết quả :



Tiếp tục chạy hàm done như bên trên

Ta được kết quả cuối cùng như sau



* Ta dùng stack để lưu nhớ $ra để sử dụng nhiều hàm .

Ngăn xếp trong trường hợp n=3

|  |  |
| --- | --- |
| 0x00000001 | $a0 |
| 0x00400080 | $ra |
| 0x7fffefe8 | $fp |
| 0x00000002 | $a0 |
| 0x00400080 | $ra |
| 0x7fffeff4 | $fp |
| 0x00000003 | $a0 |
| 0x00400038 | $ra |
| 0x7fffeffc | $fp |
| 0x00400004 | $ra |
| 0x00000000 | $fp |