**BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH TUẦN 7**

**Assignment 1**

* Code

#Laboratory Exercise 7 Home Assignment 1

.text

main: li $a0, 45 #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] $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

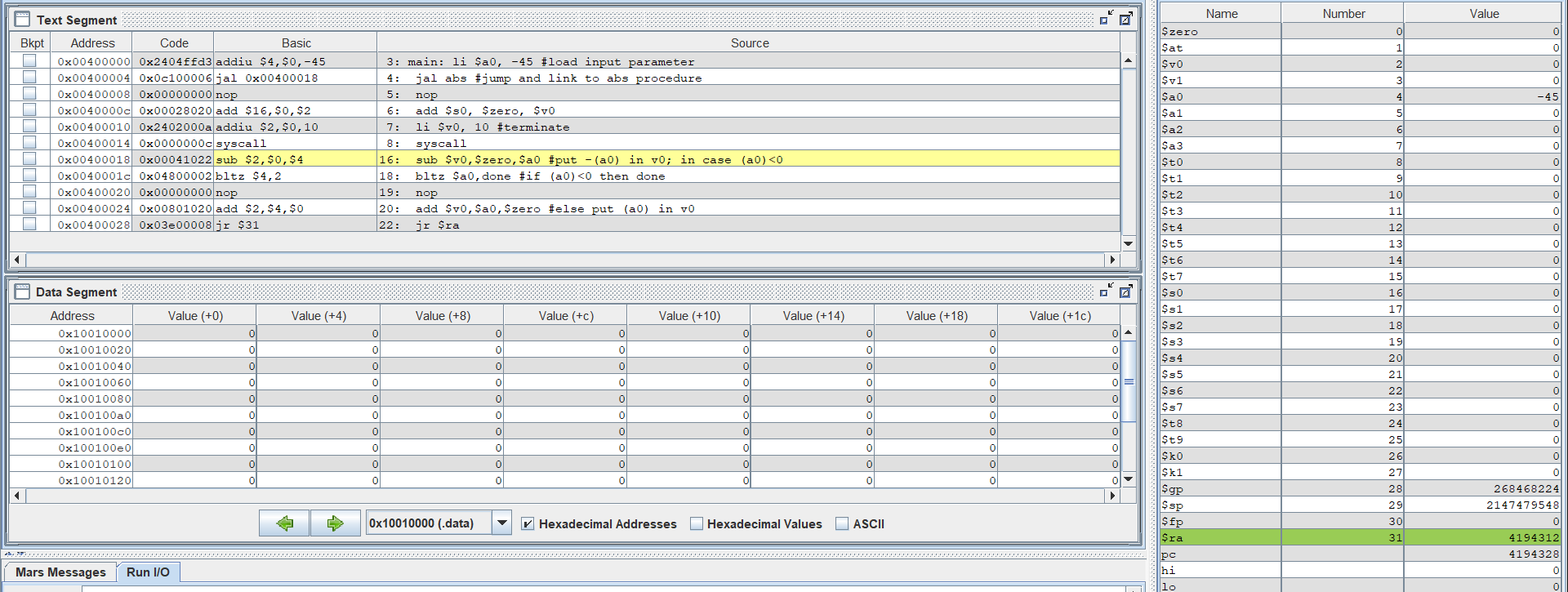
nop

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

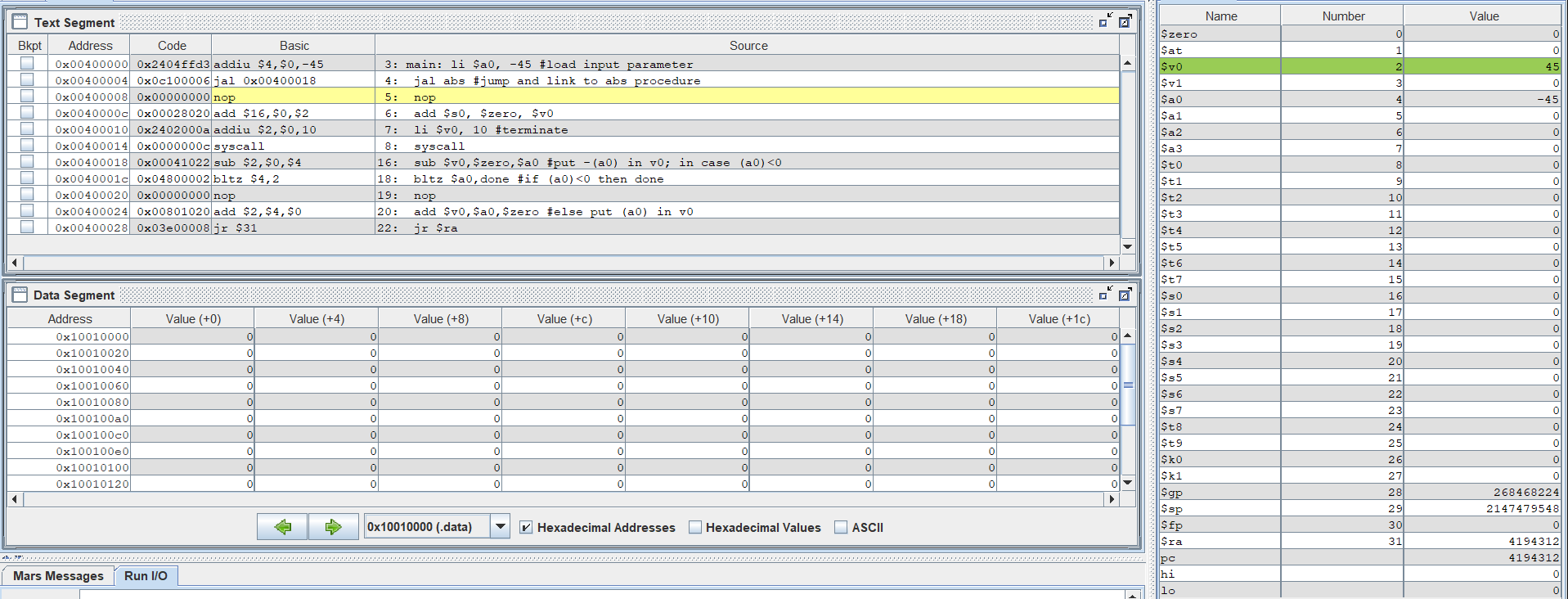
done:

jr $ra

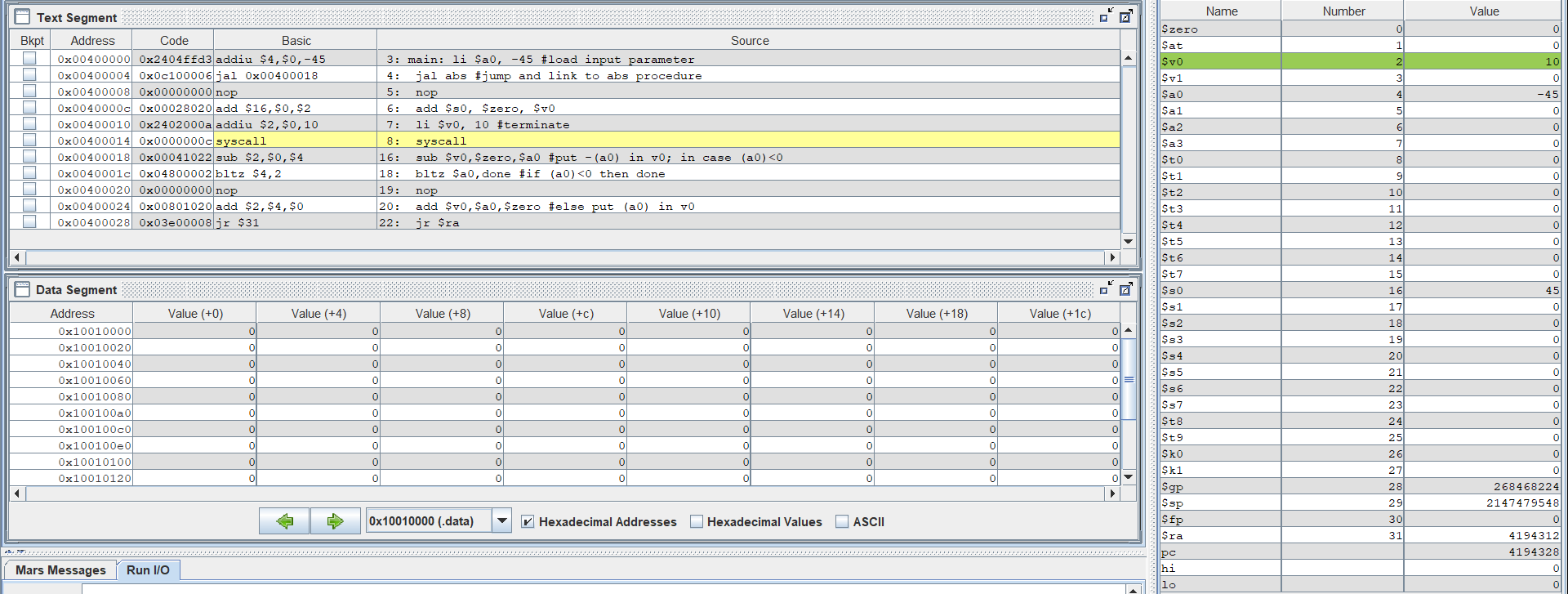
* **TH: $a0 = -45 <0**
* Lệnh: jal abs
* $pc = 4194328, $$ra = 4194312
* Gọi ra thủ tục abs

****

* Lệnh: jr $ra
* $pc = 4194312, $ra = 4194312
* Trả về hàm main từ thủ tục abs

****

* **Kết quả chạy chương trình: $a0 = 45**

****

* **Nhận xét:** Chương trình nhảy tới thủ tục abs từ lệnh jal abs. Trong thủ tục abs, chương trình so sánh $a0 với $zero, nếu $a0 >0 thì sẽ chạy lệnh add $v0,$a0,$zero , nếu $a0 < 0 thì sẽ chạy lệnh sub $v0,$zero,$a0 ( #put -(a0) in v0; in case (a0)<0 )

**Assignment 2**

* Code :

.text

main: li $a0,2 #load test input

li $a1,6

li $a2,9

jal max #call max procedure

nop

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)

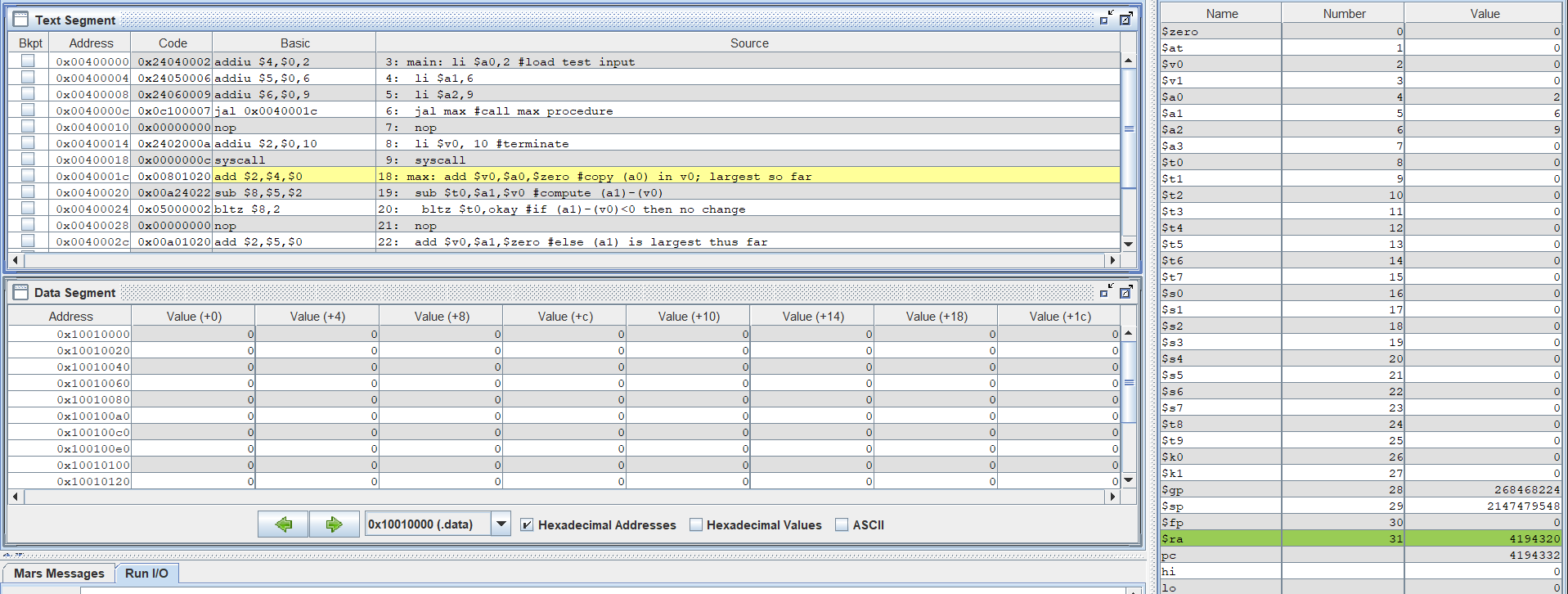
bltz $t0,done #if (a2)-(v0)<0 then no change

nop

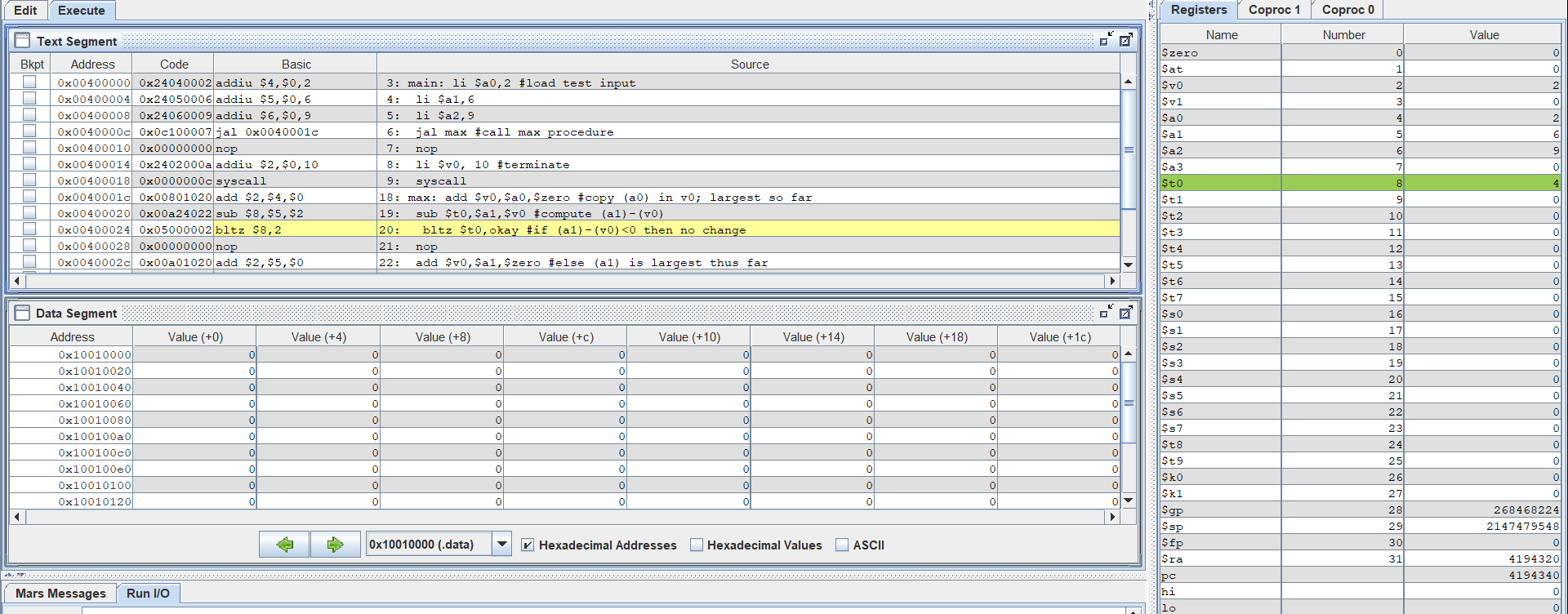
add $v0,$a2,$zero #else (a2) is largest overall

done: jr $ra #return to calling program

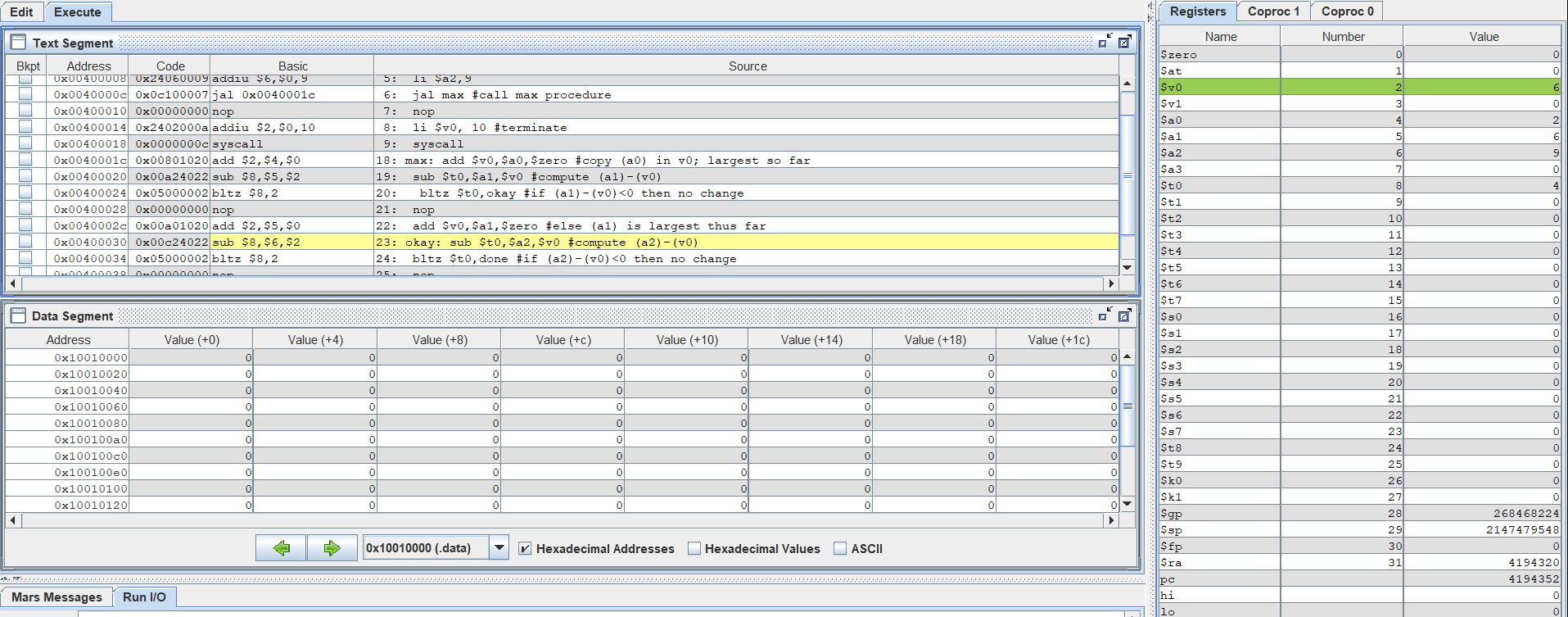
* Lệnh : jal max #call max procedure



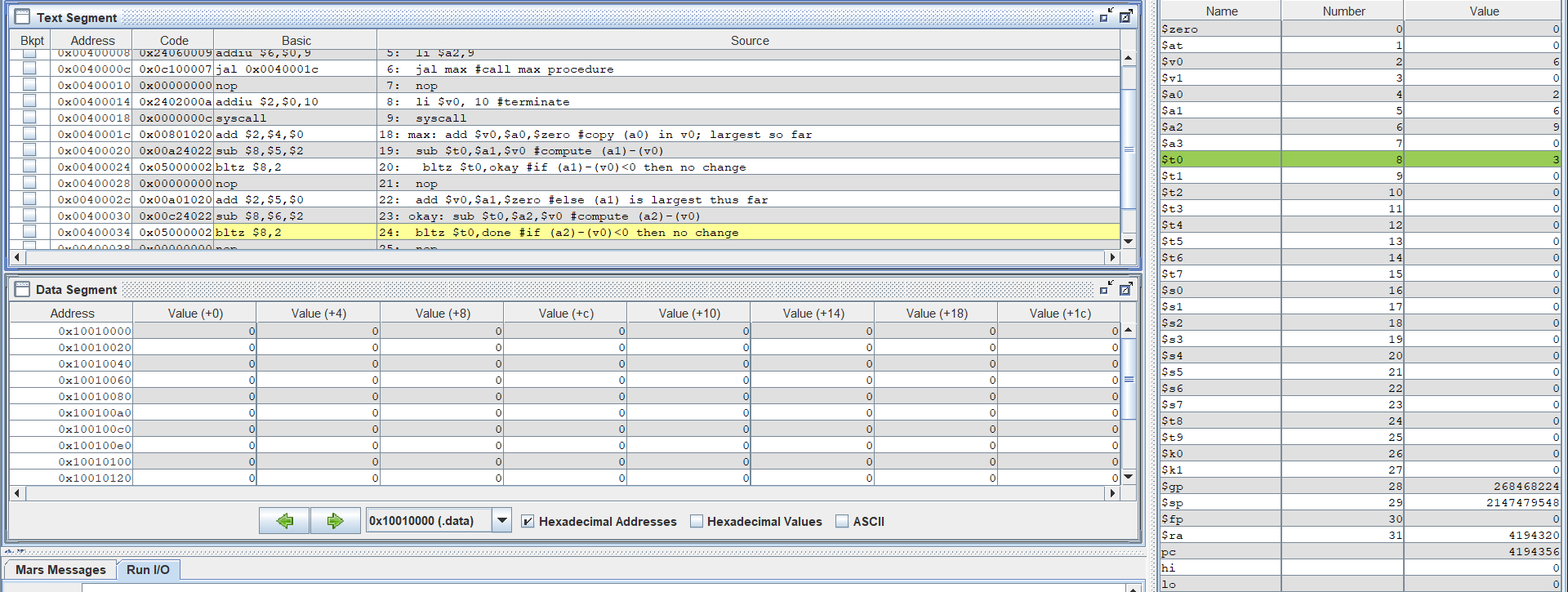
* Nhảy đến địa chỉ thủ tục **max** và địa chỉ trả về lưu tại $ra = 3194320
* **Trong thủ tục max**
* Lệnh add $v0,$a0,$zero :
* $v0 = $a0 = 2 ( giá trị lớn nhất tạm thời là $a0 được gán cho $v0)
* Lệnh sub $t0,$a1,$v0 #compute (a1)-(v0)

****

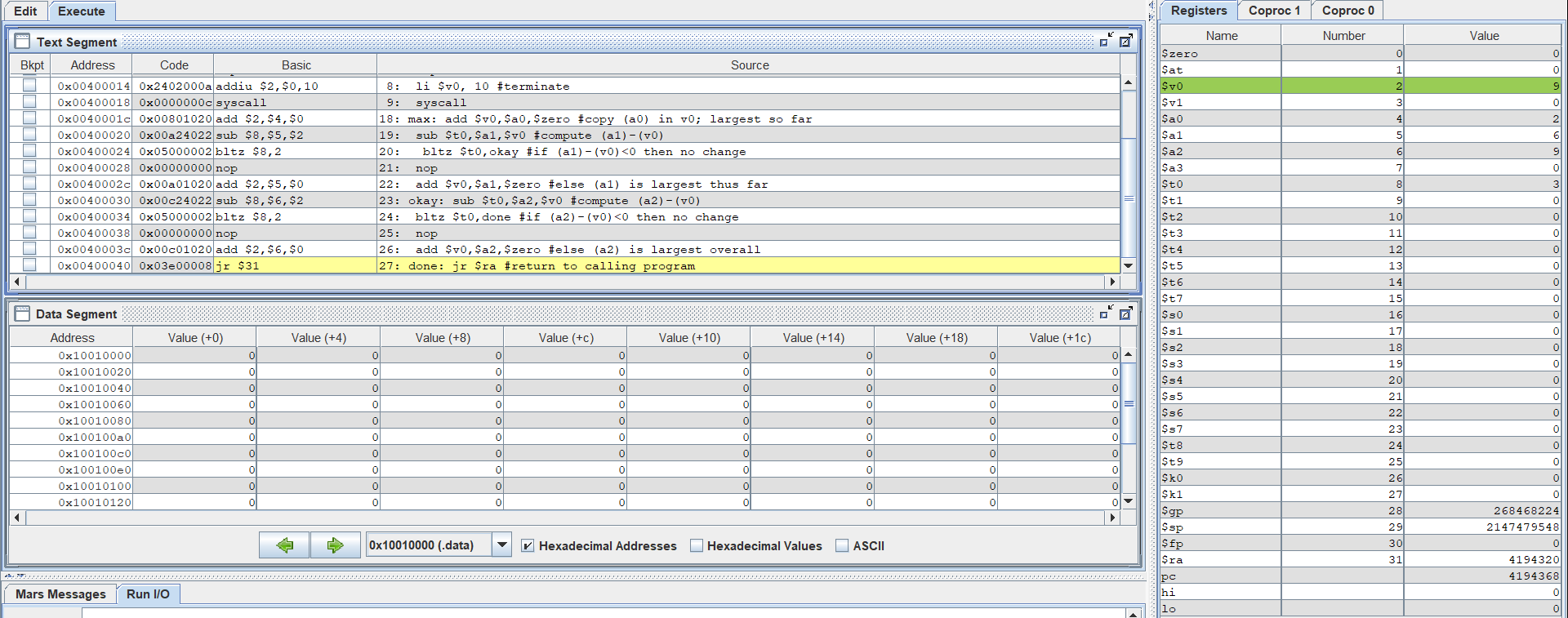
* $t0 = $a1 - $v0 = 4
* Lệnh bltz $t0,okay #if (a1)-(v0)<0 then no change
* Nếu $t0 < 0( a1<v0) => Nhảy tới okay
* Vì trong th này $t0 = 4=> chương trình không nhảy tới **okay**  mà tiếp tục chạy
* Lệnh add $v0,$a1,$zero

****

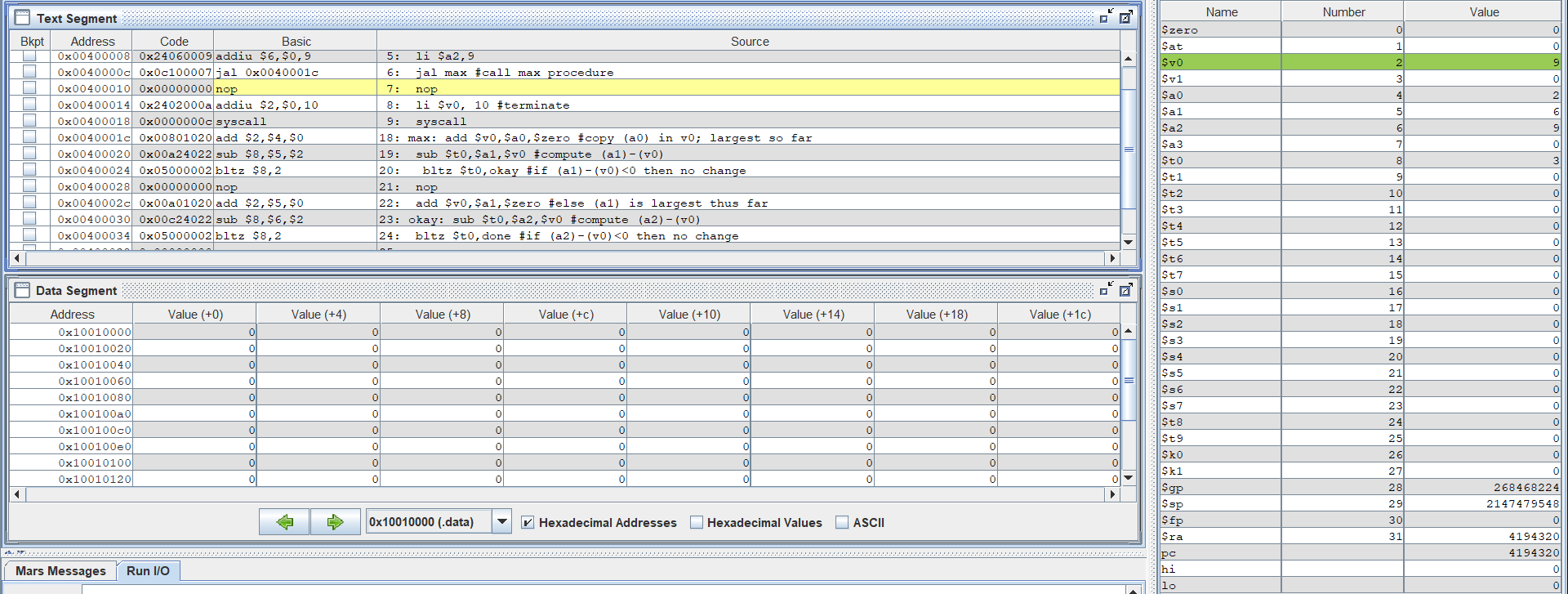
* Gán lại giá trị lớn nhất $v0 là $a1
* Lệnh okay: sub $t0,$a2,$v0 #compute (a2)-(v0)



* $t0 = $a2 - $v0 = 9-6 = 3
* Lệnh bltz $t0,done #if (a2)-(v0)<0 then no change
* Vì $t0 = 3 > 0 ($a2 > $v0 ) => chương trình chạy tiếp mà không nhảy vào **done**
* Lệnh add $v0,$a2,$zero #else (a2) is largest overall



* Gán lại giá trị lớn nhất $v0 = $a2
* Lệnh done: jr $ra



* Nhảy về **main** qua địa chỉ trả về $ra = 4194320

**Assignment 3**

* **Code:**

#Laboratory Exercise 7, Home Assignment 3

.text

li $s0,-1

li $s1,1

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

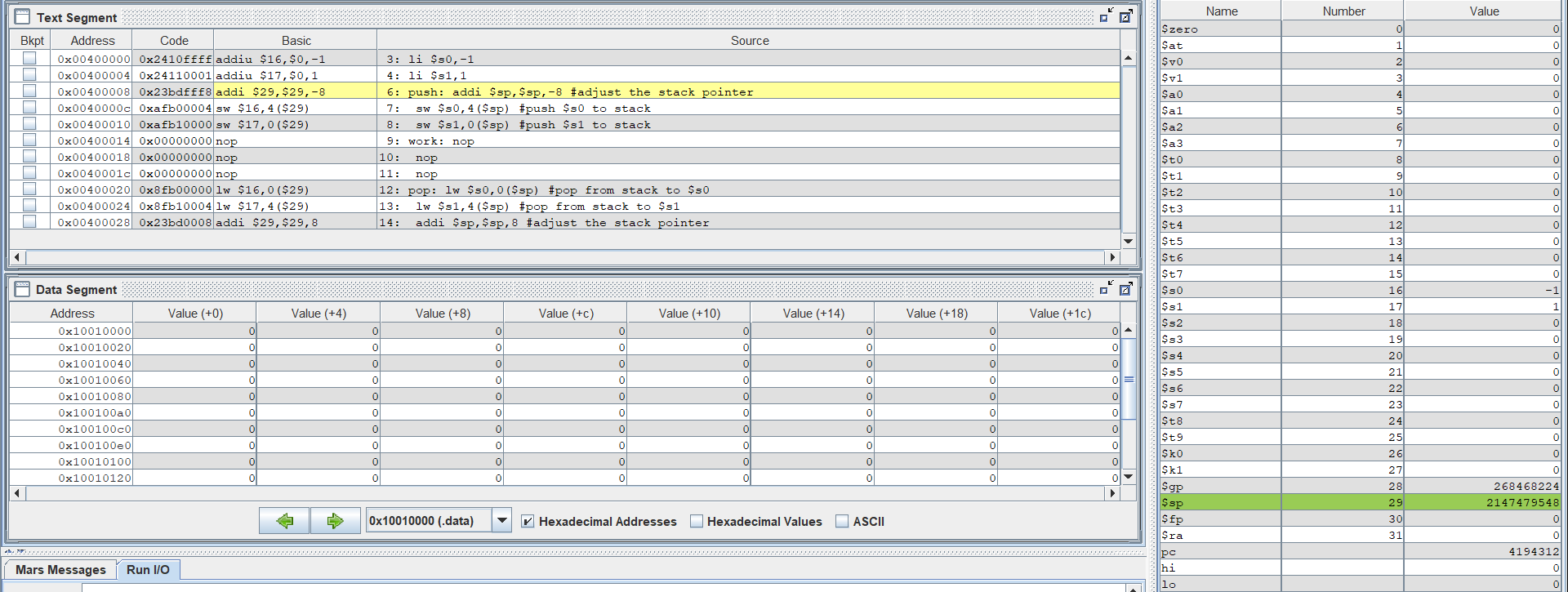
nop

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

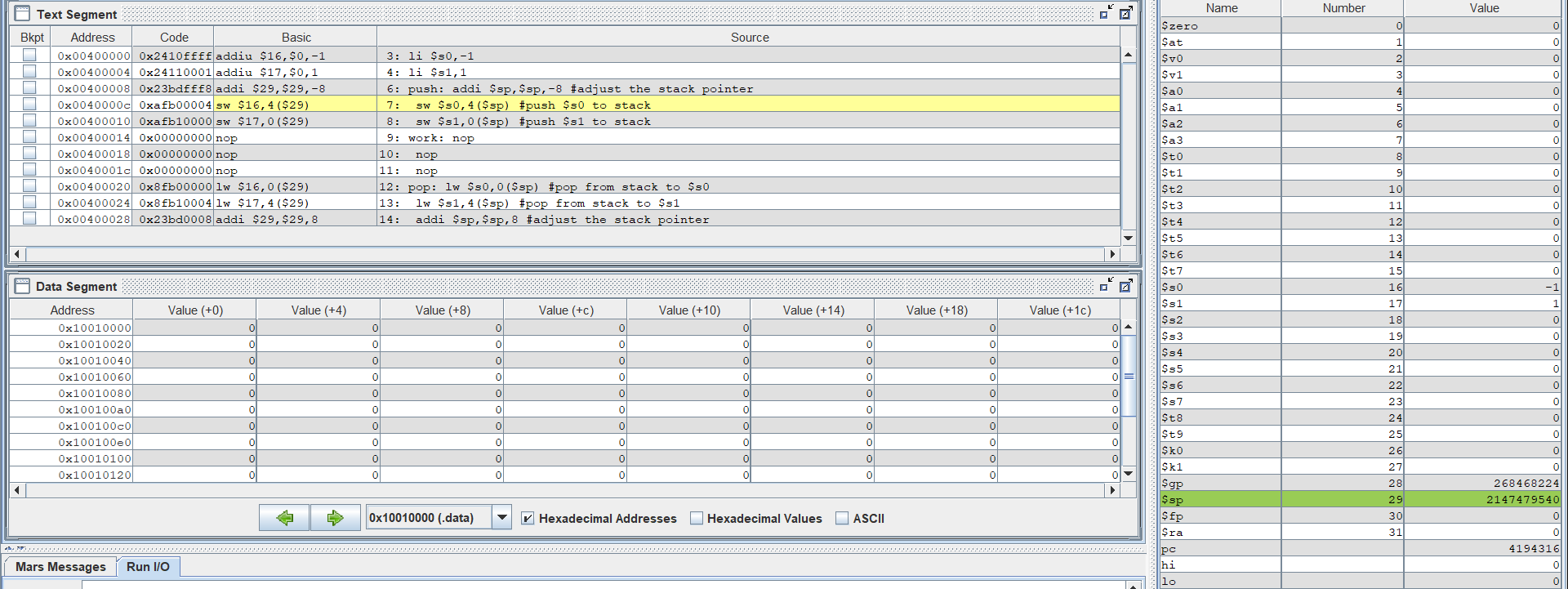
lw $s1,4($sp) #pop from stack to $s1

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

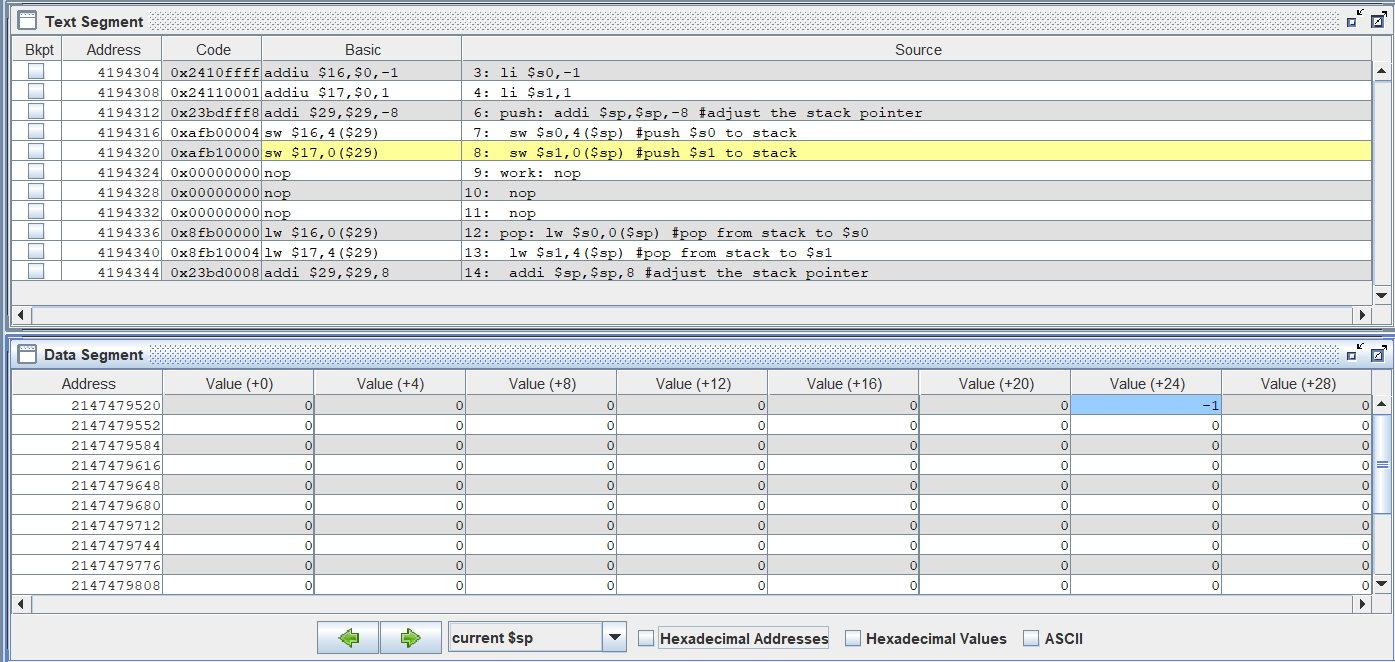
* Lúc đầu $sp = 2147479548



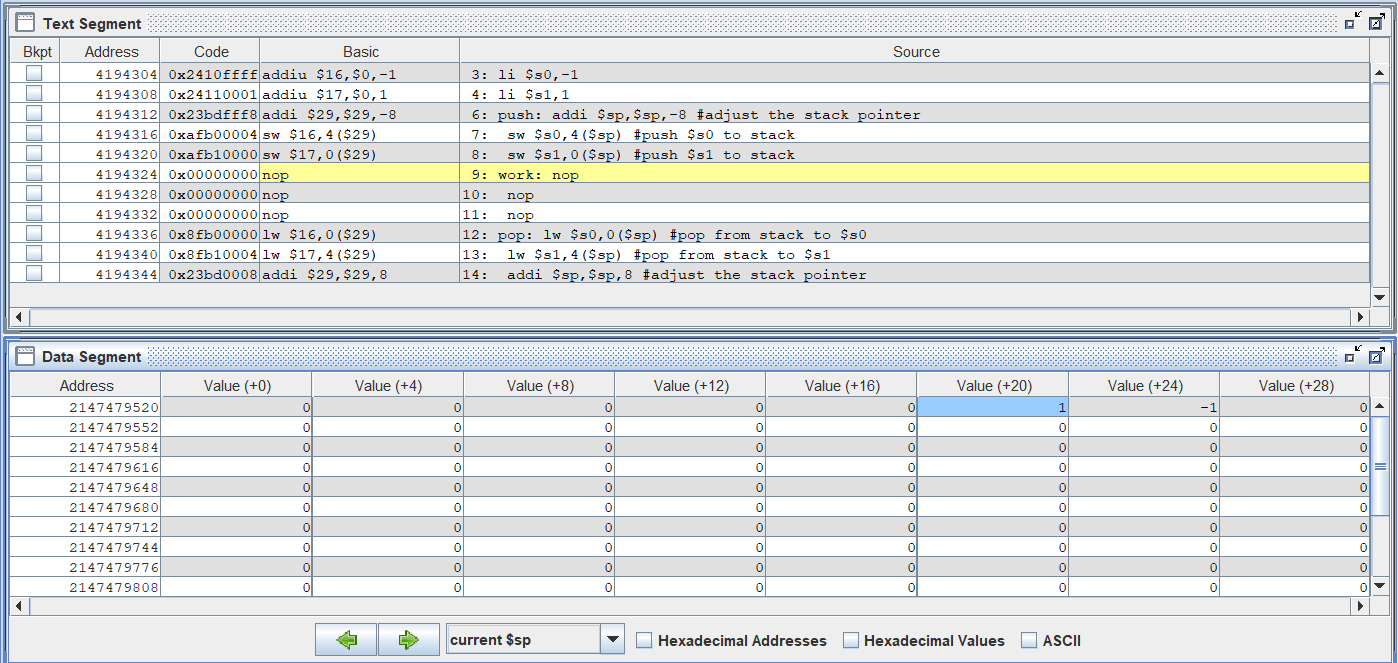
* Lệnh push: addi $sp,$sp,-8



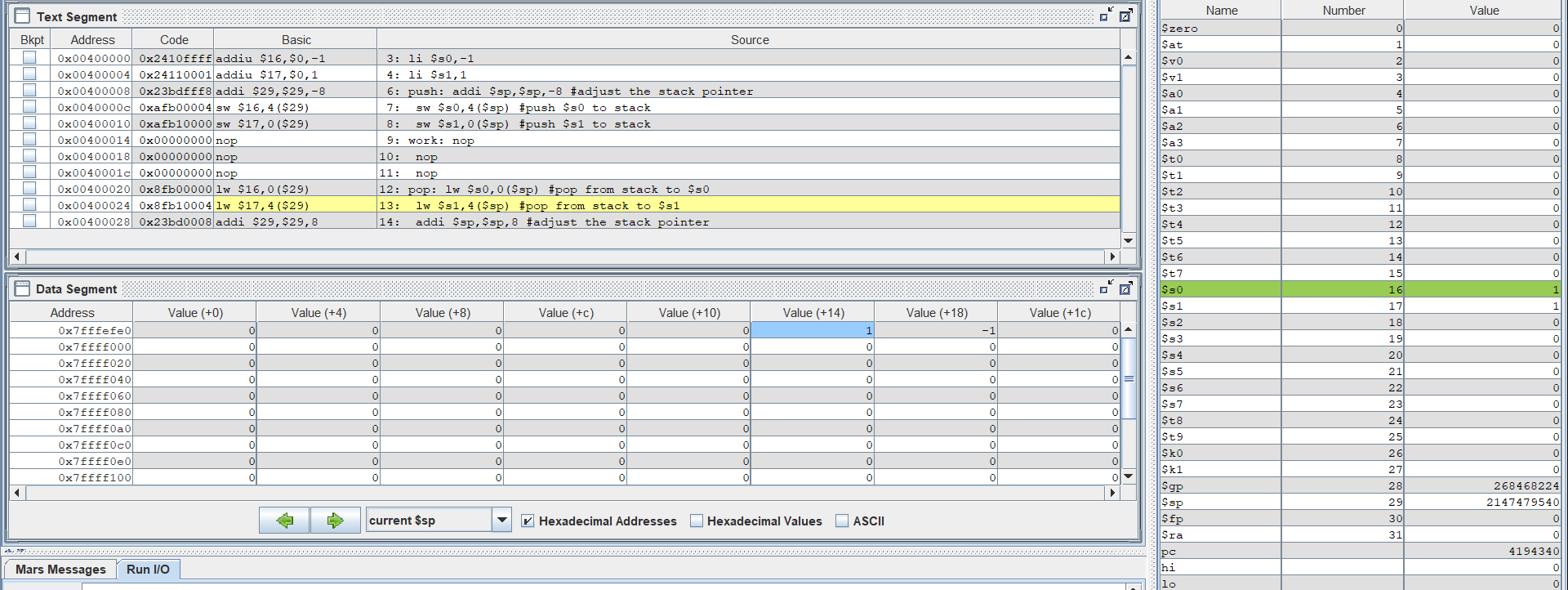
* $sp = 2147479548 – 8 = 2147479540
* Lệnh sw $s0,4($sp)



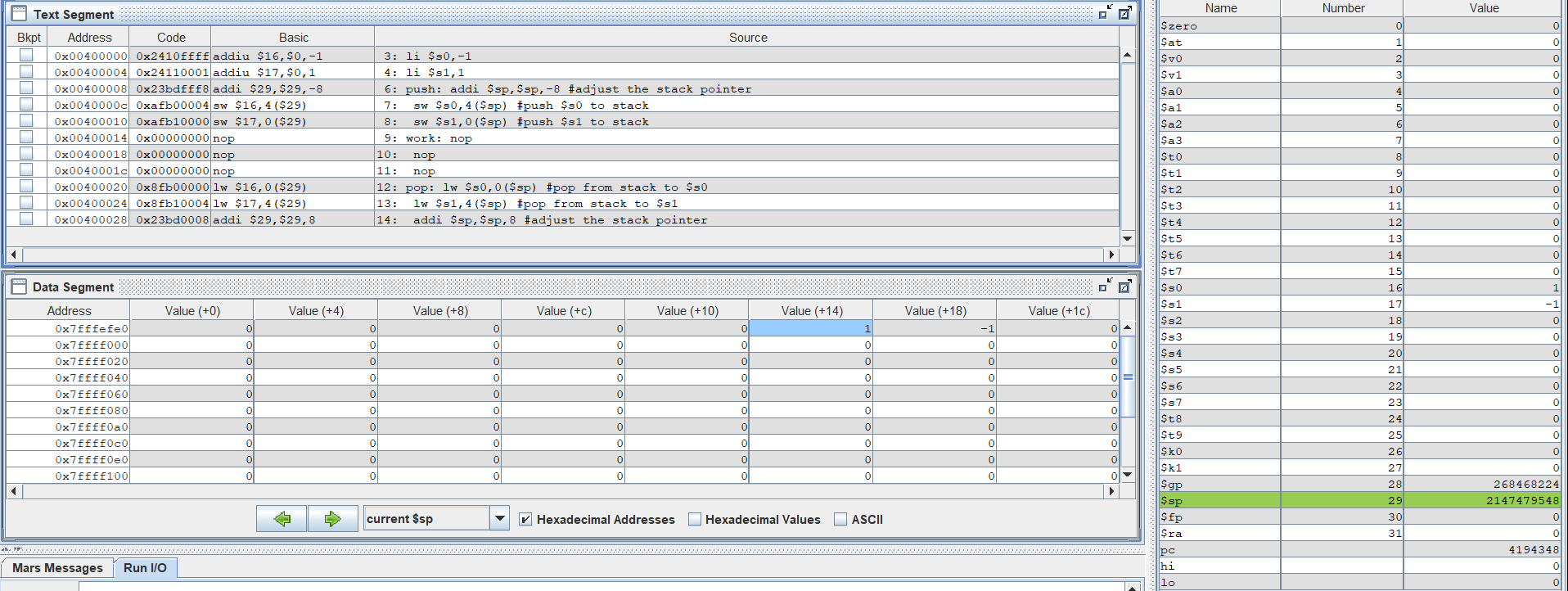
* Ở biến nhớ địa chỉ 2147479544 lưu giá trị -1
* Lệnh sw $s1,0($sp)



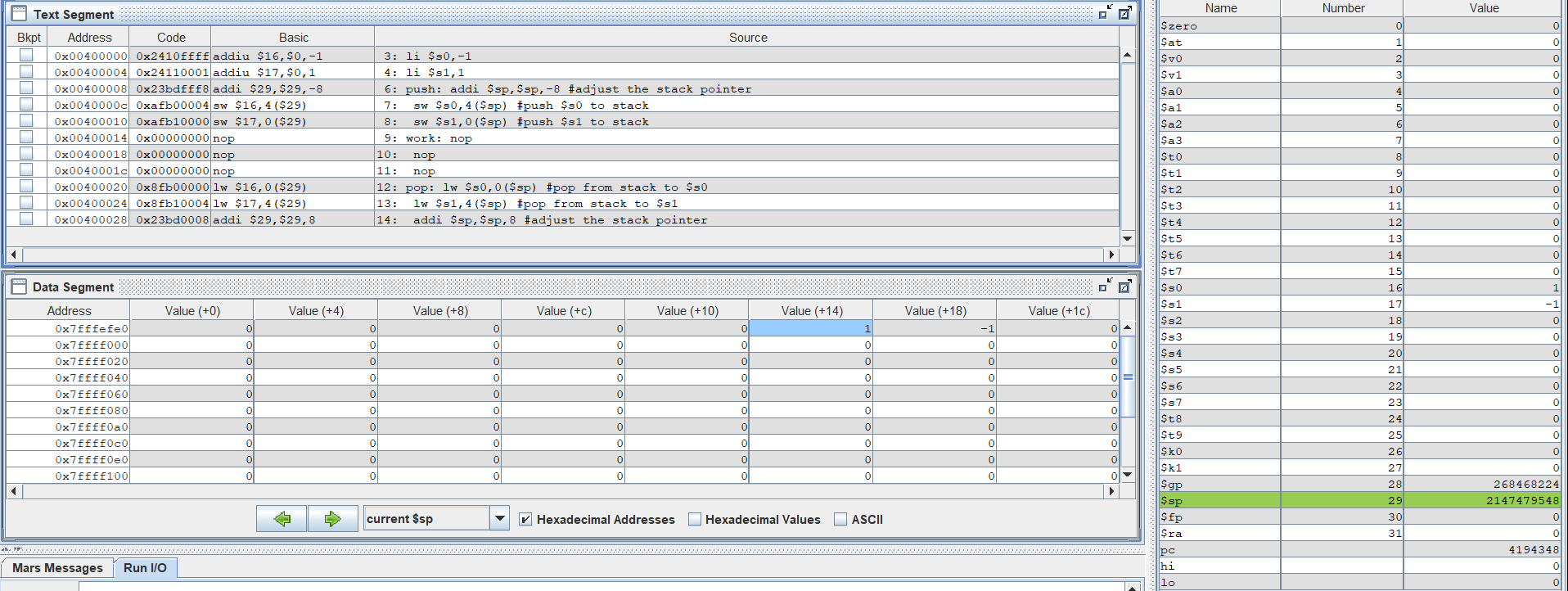
* Ở biến nhớ địa chỉ 2147479540 lưu giá trị 1( $s1 được bỏ vào ngăn xếp sau $s0)
* Lệnh pop: lw $s0,0($sp)

****

* $s0 =1 => lấy từ đầu ngăn xếp ra biến $s0
* Lệnh lw $s1,4($sp)



* $s1 = -1 => lấy ở cuối ngăn xếp ra biến $s1
* Lệnh addi $sp,$sp,8



* $sp = 214749548 => Xóa 2 mục trong stack

**Assignment 4**

* Code:

#Laboratory Exercise 7, Home Assignment 4

.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

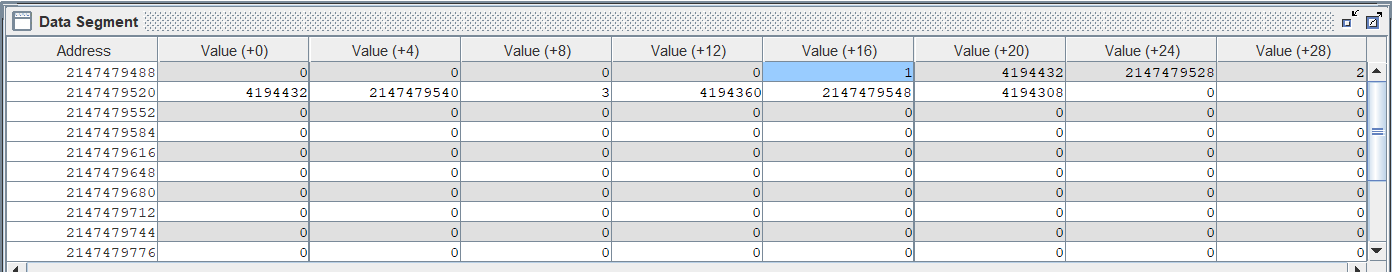
addi $sp,$fp,0 #restore stack pointer

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

jr $ra #jump to calling

fact\_end:

Bộ nhớ ngăn xếp trong trường hợp n = 3 ( 3! )



Nhận xét:

* Frame pointer được lưu mỗi khi gọi thủ tục FACT ở ô nhớ -4($sp)
* Sau đó ta cấp bộ nhớ cho các phần tử fp,$ra,$a0 trong stack bằng lệnh addi $sp,$sp,-12
* Chương trình sẽ lưu lần lượt địa chỉ trả $ra về và giá trị hiện tại của thanh ghi $a0 lần lượt vào 2 vị trí đã được cấp phát còn lại ( $a0 ở 0($sp) và $ra ở 4($sp) ) trong ngăn xếp