Laboratory Exercise 7 – Report:

Procedure calls, stack and parameters

Lê Văn Duẩn - 20194508

1. Assignment 1

- Mã nguồn:

```
#Laboratory Exercise 7 Home Assignment 1


.text
main: li $a0, 20194508 #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,$a0 #put -(a0) in v0; in case (a0)<0
```

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

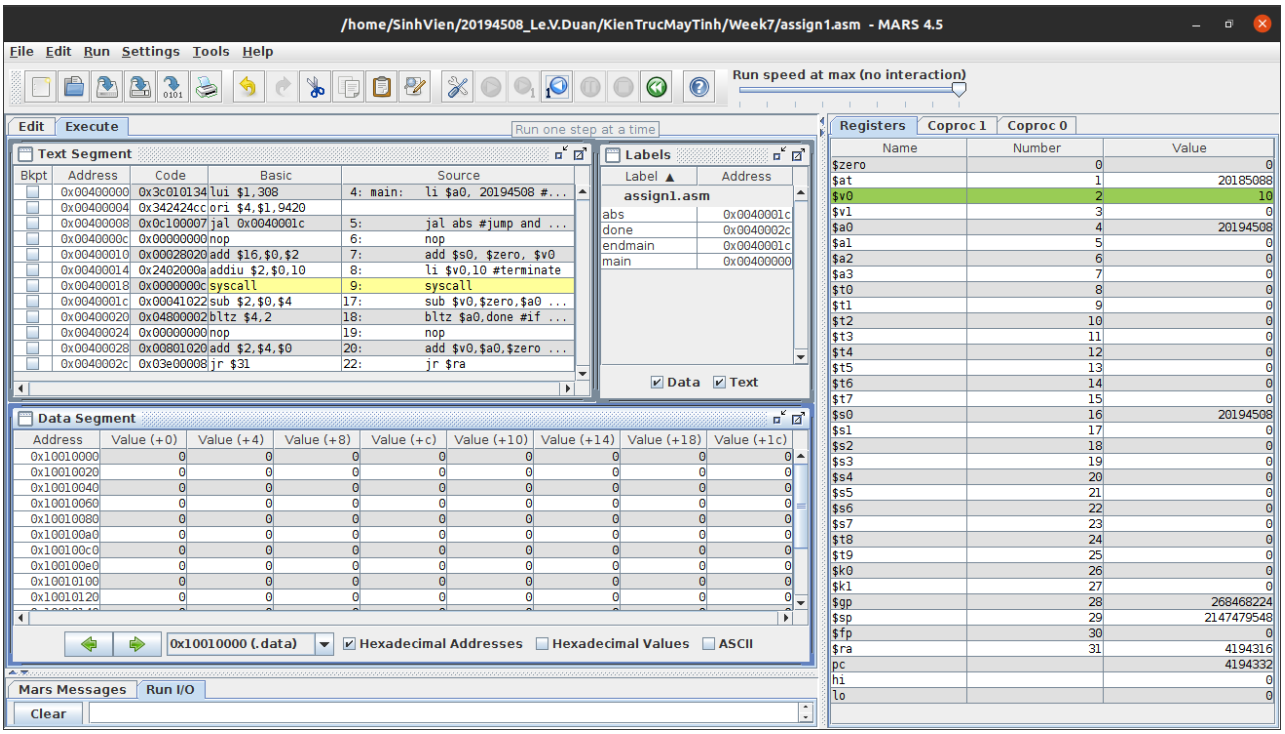
nop

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

done:

jr $ra

- Kết quả chạy mô phỏng:



- Giải thích:

+ Chương trình nhận tham số 20194508 vào thanh ghi $a0 và kết quả trị tuyệt đối của nó được lưu vào thanh ghi $v0

+ lệnh jar abs -> nhảy đến thủ tục có nhãn abs và lưu thanh ghi $ra = pc = 0x004000c (địa chỉ lệnh tiếp theo chính là lệnh nop trong chương trình chính), còn pc thay đổi từ 0x004008 (địa chỉ lện jal hiện tại) thành địa chỉ đầu tiên của thủ tục abs 0x0040001c

+ sau khi thực hiện xong thủ tục vào lưu kết quả vào $v0 trong ví dụ này là 20194508 thì thoát thủ tục bằng lệnh jr $ra -> lưu lại thanh ghi pc = $ra để quay

lại chương trình chính. Nên pc thay đổi từ 0x0040002c(địa chỉ của lệnh jr hiện tại) thành 0x004000c($ra)

2. Assignment 2

- Mã nguồn:

```
#Laboratory Exercise 7 Home Assignment 2
.text
main: li $a0,5 #load test input (20194)508
li $a1,0
li $a2,8
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 t0 = (a1)-(v0) = (a1) - (a0)
bltz $t0,okay #if (a1)-(a0)<0 then no change
```

nop

add $v0,$a1,$zero #else (a1) is largest thus far

okay: sub $t0,$a2,$v0 #compute (a2)-(v0)

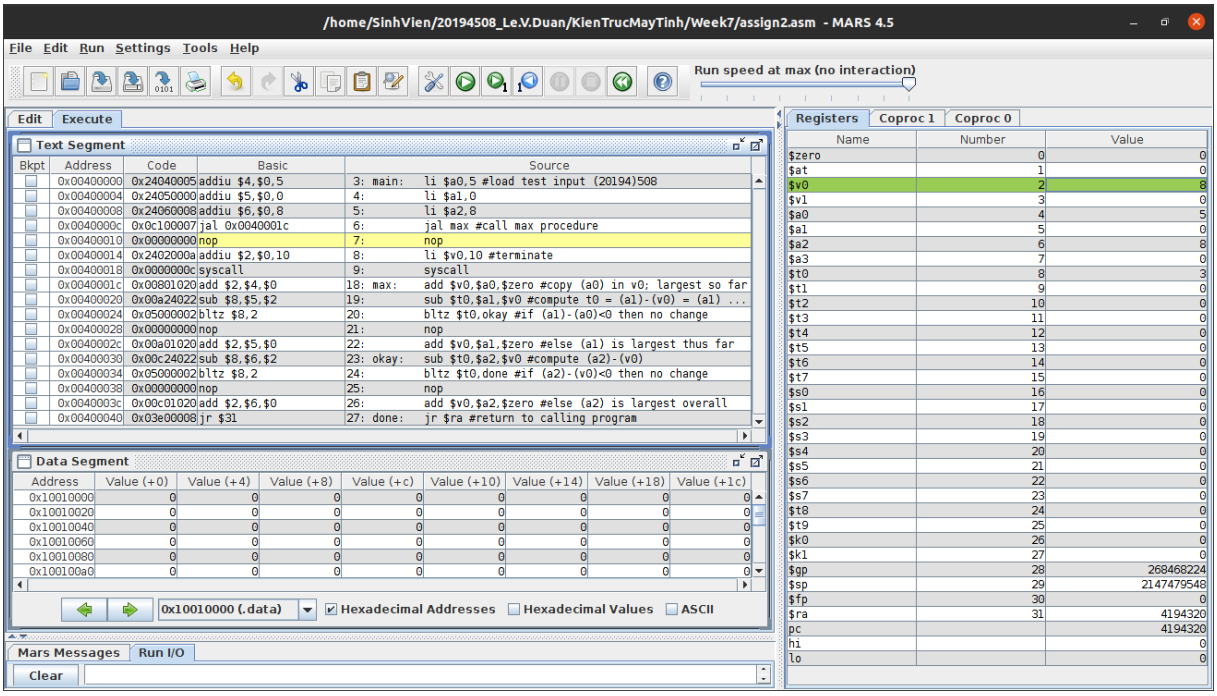
bltz $t0,done #if (a2)-(v0)<0 then no change

nop

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

done: jr $ra #return to calling program

- Kết quả chạy:



- Giải thích:

+ Chương trình nhận 3 tham số đầu vào lưu lần lượt trong 3 thanh ghi $a0, $a1, $a2 tìm max trong 3 số vào lưu vào thanh ghi $v0

+ lệnh jal max -> nhảy đến thủ tục với nhãn max, thanh ghi $ra = pc nên $ra = 0x0040000c là giá trị hiện tại của pc và pc thay đổi từ 0x0040000c thành 0x004001c là địa chỉ của nhãn max

+ Thủ tục max:

add $v0,$a0,$zero -> gán giá trị max tạm thời vào thanh ghi $v0 = $a0 = 5

sub $t0,$a1,$v0 -> tính hiệu $a1 - $a0 và lưu vào $t0 nên giá trị là 0 - 5 = -5

bltz $t0,okay -> nếu $t0 < 0 -> $a0 > $a1 -> $v0 = $a0 = 5

nop

add $v0,$a1,$zero -> nếu $t0 > 0 -> $a1 > $a0 -> cập nhật max = $a0 -> $v0 = $a1 nhưng trong ví dụ này ko xảy ra nên $v0 không thay đổi

okay: sub $t0,$a2,$v0 -> tương tự xét hiệu của max hiện tại với $a2

bltz $t0,done -> nếu $t < 0 thì max hiện tại chính là max trong 3 số -> done

nop

add $v0,$a2,$zero -> ngược lại $t0 > 0 -> $a2 > max hiện tại trong $v0 đúng trong ví dụ là 8 > 5 nên thanh ghi $v0 được cập nhật là giá trị của $a2 là 8

done: jr $ra -> pc = $ra để trở lại chương trình chính nên nó thay đổi từ 0x40040 (địa chỉ lệnh jr hiện tại) thành 0x400010 (địa chỉ tiếp theo hay chính là lệnh nop trên chương trình chính)

+ kết quả: trước khi gán lại $v0 = 10 để kết thúc chương trình thì $v0 = 8 đúng với kết quả mong muốn

3. Assignment 3

-Mã nguồn:

#Laboratory Exercise 7, Home Assignment 3

.text

push: li $s0, 2019

li $s1, 4508

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

sw $s0,0($sp) #push $s0 to stack

sw $s1,4($sp) #push $s1 to stack

work: nop

nop

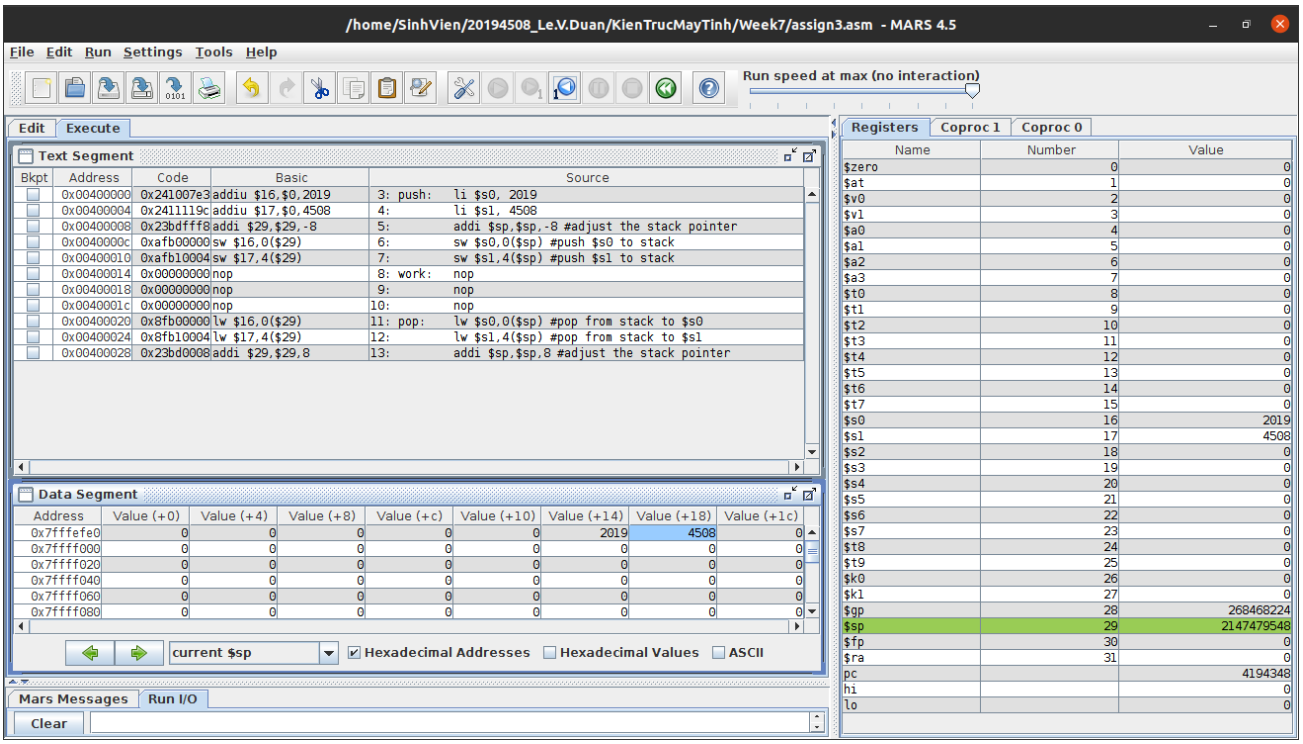
nop

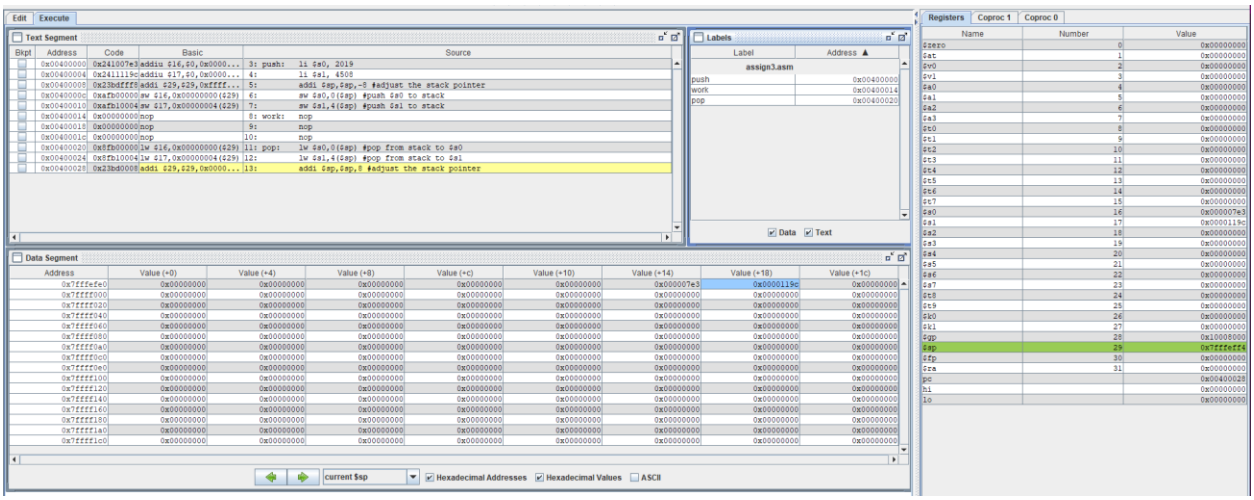
pop: lw $s0,0($sp) #pop from stack to $s0

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

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

- Kết quả chạy mô phỏng:





- Giải thích:

+ Tạo ngăn xếp để push và pop 2 tham số vào 2019, 4508 được lưu vào các thanh ghi $s0, $s1

+ addi $sp,$sp,-8 -> thanh ghi $sp lùi 8 byte để đủ chỗ cho 2 phần tử của stack và vì stack có đáy ở dưới và chiều địa chỉ từ trên xuống nên phải là -8. $sp cũng đóng vai trò chứa địa chỉ của các phần tử trên cùng của ngăn xếp.

-> giá trị $sp thay đổi từ giá trị ban đầu 0x7fffeffc thành 0x7fffeff4

+ 2 lệnh sw để push lần lượt vào thanh ghi nên gía trị sẽ được lưu và hiển thị trong data segment ở địa chỉ tương ứng như trong hình.

+ pop phần tử ra khỏi stack bằng cách lw phần tử và điều chỉnh con trỏ $sp như trong ví dụ chúng ta pop cả 2 phần tử nên phải +8 (vì kích thước phần tử là 4byte) -> $sp = 0x7fffeffc chính là địa chỉ ban đầu -> stack rỗng

4. Assignment 4

- Mã nguồn:

#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 $a0,8 #load test input N = 8
        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 stackâ€™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
```

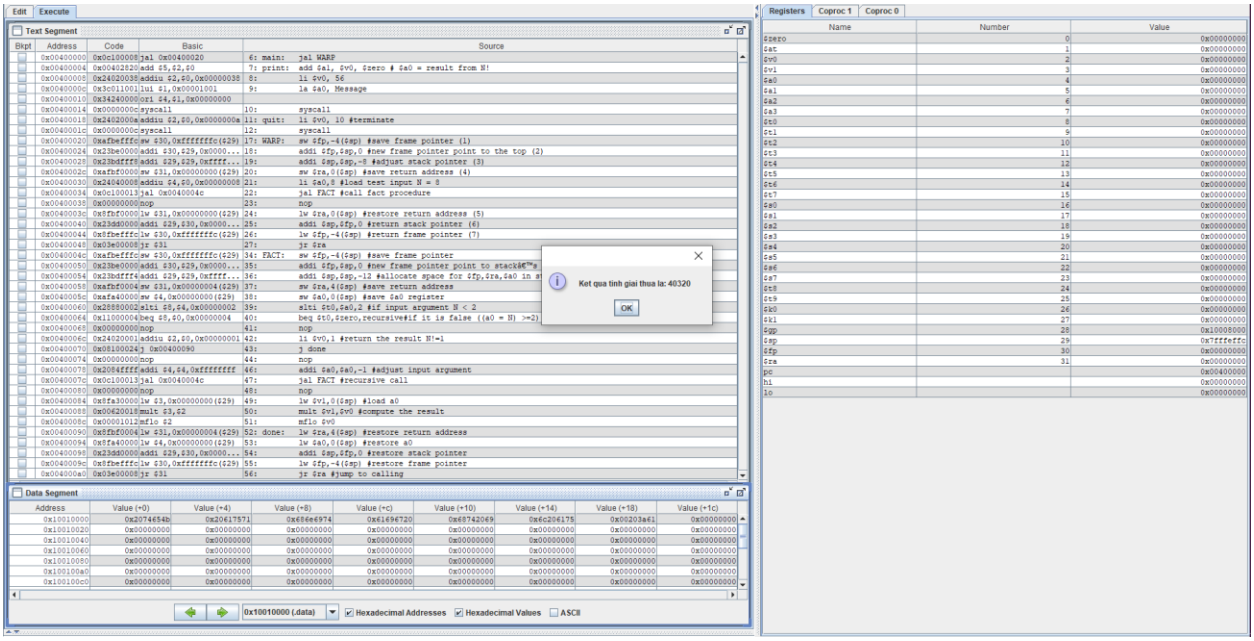
addi $sp,$fp,0 #restore stack pointer

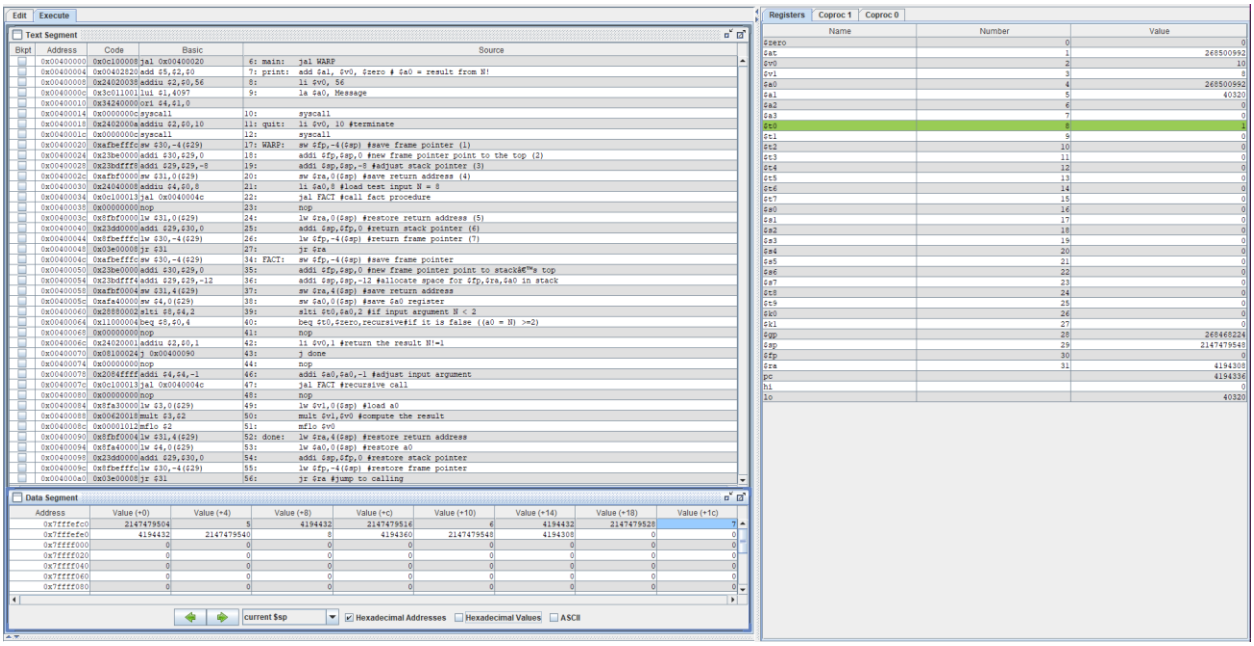
lw $fp,-4($sp) #restore frame pointer

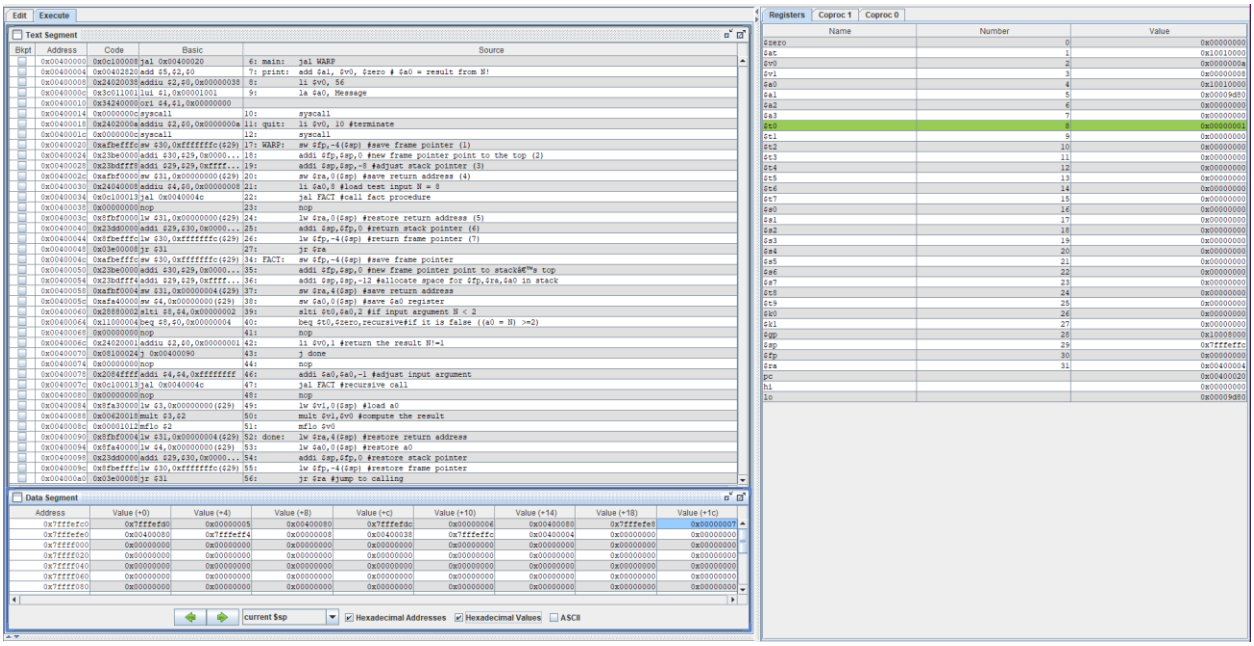
jr $ra #jump to calling

fact_end:

- Kết quả chạy:







- Giải thích:

+ thanh ghi $fp luuw con trỏ đến khung trang frame pointer, $sp – stack pointer

+ thanh ghi $a0 lưu kết quả của chương trình

+ thủ tục WRAP:

- Khai báo 2 vị trí trong stack cho khung trang và địa chỉ return -> $fp, $ra

-> vì thế nên địa chỉ thanh và địa chỉ thanh return nên giá trị thanh ghi $fp thanh đổi sang 0x7fffeffc, vì có thủ tục lồng nhau nên ta dùng lưu địa chỉ $ra ra bộ nhớ ở địa chỉ 0($sp) bằng lệnh sw -> hiển thị trong data segment

- Lưu giá trị N vào thanh ghi $a0 nên $a0 = 0x00000008

- gọi thủ tục FACT

- Restore giá trị từ 0($sp) vào lại $ra để gọi lệnh jr $ra trở về chính xác địa chỉ lệnh tiếp theo trong chương trình chính

+ Thủ tục FACT:

- Khai báo 3 vị trí trong stack trong cho khung trang và các giá trị trả về $fp, $ra, $a0 -> $sp lùi 12 giá trị -> $sp = 0x7fffeff4 -> 0x7fffefe8, cũng như WRAP lưu giá trị lần lượt vào stack

- Xét số đưa vào <2 -> kết quả = 1

- nếu >= 2 đi đến thủ tục RECURSIVE

- Nhảy đến nhãn done để restore lại các giá trị return vào thanh ghi $ra để lệnh jr $ra được thực thi chính xác

+ Giá trị thay đổi của các thanh ghi:

Step	$pc	Giá trị thanh ghi thay đổi
1	0x00400020	$ra = 0x00400004
2	0x00400024	$ra = 0x00400004
3	0x00400028	$fp = 0x7fffeffc
4	0x0040002c	$sp = 0x7fffeff4
5	0x00400030	$sp = 0x7fffeff4
6	0x00400034	$a0 = 0x00000000
7	0x0040004c	$ra = 0x00000038
8	0x00400050	$ra = 0x00000038
9	0x00400054	$fp = 0x7fffeff4
10	0x00400058	$sp = 0x7fffeff8
11	0x00400064	$t0 = 0x00000001
12	0x00400070	$v0 = 0x00000001
13	0x00400094	$ra = 0x00000038
14	0x00400098	$a0 = 0x00000000
15	0x0040009c	$fp = 0x7fffeff4
16	0x004000a0	$fp = 0x7fffeffc
Recurise ...	…	…

5. Assignment 5

- Mã nguồn:

```
#Laboratory Exercise 7 Home Assignment 5

.data

message_max: .asciiz "Largest: "

message_index: .asciiz ", "

message_min: .asciiz "Smallest: "

enter: .asciiz "\n"
```

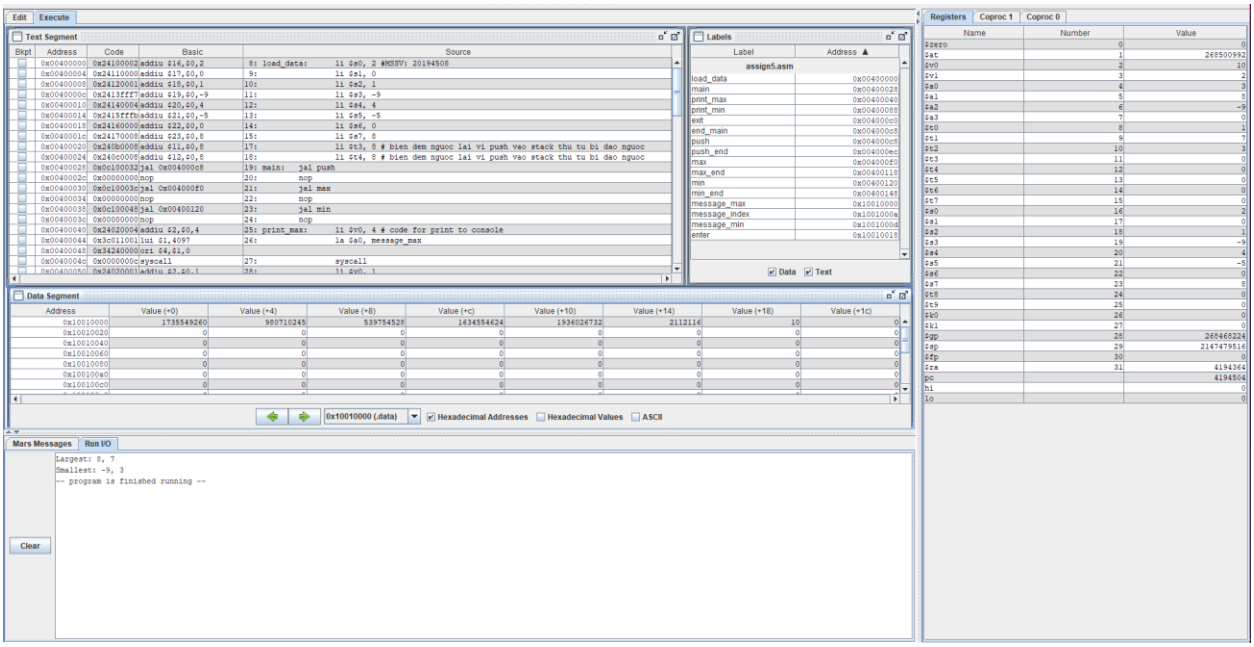
```
.text
load_data:  li $s0, 2 #MSSV: 20194508
            li $s1, 0
            li $s2, 1
            li $s3, -9
            li $s4, 4
            li $s5, -5
            li $s6, 0
            li $s7, 8

            li $t3, 8 # bien dem nguoc lai vi push vao stack thu tu bi dao nguoc
            li $t4, 8 # bien dem nguoc lai vi push vao stack thu tu bi dao nguoc
main: jal push
      nop
      jal max
      nop
      jal min
      nop
print_max:  li $v0, 4 # code for print to console
            la $a0, message_max
            syscall
            li $v0, 1
            add $a0, $zero, $a1 # $a1 = max
            syscall
```

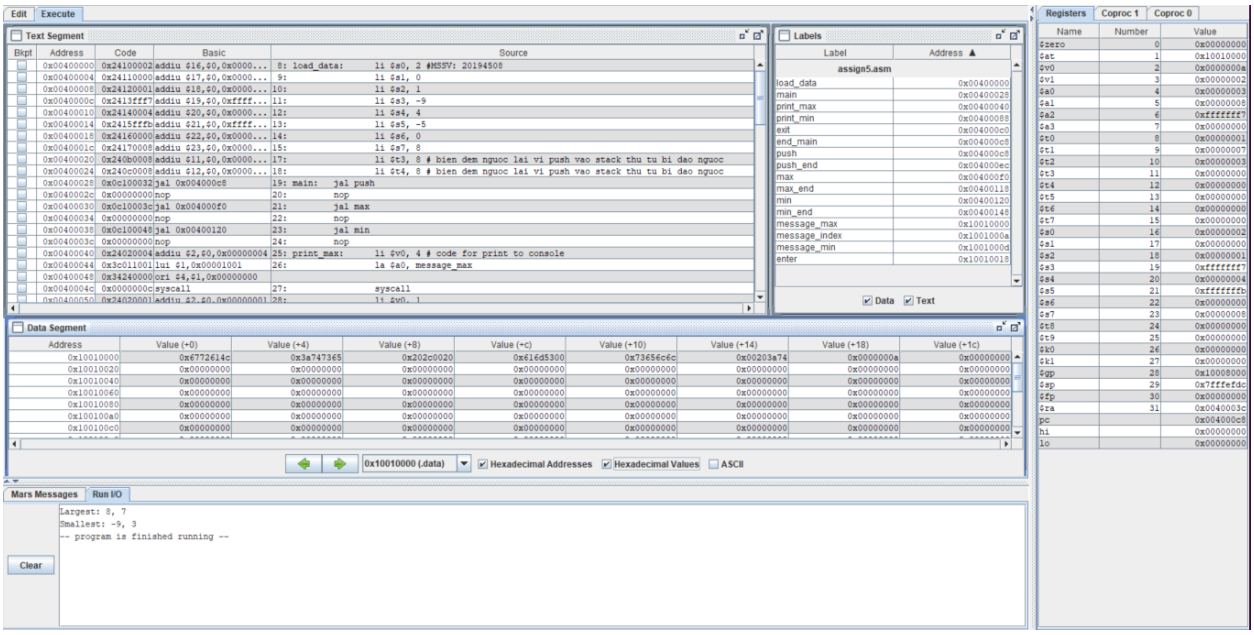
```
                li $v0, 4
                la $a0, message_index
                syscall
                li $v0, 1
                add $a0, $zero, $t1 # $t1 -> register containing the max value
                syscall
                li $v0, 4
                la $a0, enter
                syscall
print_min:      li $v0, 4
                la $a0, message_min
                syscall
                li $v0, 1
                add $a0, $zero, $a2 # $t3 = min
                syscall
                li $v0, 4
                la $a0, message_index
                syscall
                li $v0, 1
                add $a0, $0, $t2  # $t4 -> register containing the min value
                syscall
exit:   li $v0, 10
        syscall
end_main:
```

```
#---------------------------------
push: addi $sp, $sp, -32 # stack chua 8 ptu
        sw $s0, 28($sp)
        sw $s1, 24($sp)
        sw $s2, 20($sp)
        sw $s3, 16($sp)
        sw $s4, 12($sp)
        sw $s5, 8($sp)
        sw $s6, 4($sp)
        sw $s7, 0($sp)
push_end:   jr $ra
#-------------------------------------------
max:  lw $v1, 0($sp)        # pop phan tu dau tien vao $v1
        addi $sp, $sp, 4
        addi $t3, $t3,-1  # index -1, giam sau moi lan lap

        beq $t3, 0, max_end # kiem tra xem da den phan tu cuoi hay chua
        slt $t0, $v1, $a1 # $a1 luu gia tri max hien tai
        bne $t0, $zero, max
        add $t1, $zero, $t3 #luu index cua max
        add $a1, $zero, $v1 #luu max vao tu $v1 sang $t5
        j max
max_end:    add $sp, $sp, -32 #adjust stack pointer to the top of stack
                jr $ra
```

```
#-----------------------------------------
min:   lw $v1, 0($sp)        #pop tu stack
       addi $sp, $sp, 4
       addi $t4, $t4, -1 # $t4 -> index = 1, tang sau moi lan lap

       beq $t4, 0, min_end # khi stack rong -> min_end
       slt $t0, $a2, $v1 # so sanh $1 dang chua min hien tai voi $t0 la ptu hien tai
       bne $t0, $zero, min
       add $t2, $zero, $t4 # luu index cua min
       add $a2, $zero, $v1 # luu gia tri min
       j min
min_end:     add $sp, $sp, -32 #adjust stack pointer to the top of stack
             jr $ra
```

- Kết quả chạy mô phỏng:





- Giải thích:

+ Chương trình tìm max, min và vị trí thanh ghi lưu trữ nó trong dãy số được lưu lần lượt vào lần lượt thanh ghi $s0 -> $s7 vì thế giá trị các thanh ghi thay đổi từ 0x00000000 sang giá trị tương ứng. Thanh ghi $a1, $a2 lưu max và min, thanh ghi $t1, St2 lưu index max và min tương ứng.

+ Thanh ghi $t3, $t4 lưu giá trị biến index để đếm sau các vòng lặp để tìm min, max. Vì dùng stack nên khi push vào thứ tự bị đảo ngược nên index cũng đếm lùi và vì có 8 phần tử nên gán giá trị ban đầu là 8 nên giá trị 2 thanh ghi đều là 0x00000008

+ Thủ tục main: - nhảy đến thủ tục push để lưu các giá trị dãy số vào lần lượt

- nhảy đến thủ tục max để tìm max

- nhảy đến thủ tục min để tìm min

+ Thủ tục max: - pop từ stack vào lưu vào $v1 nên giá trị sẽ thay đổi trở thành giá trị của phần tử pop ra, và giá trị của $sp tăng lên 4 để trỏ đến phần tử top stack

- giảm biến đếm bằng thanh ghi $t3

- kiểm tra kết thúc bằng $t3 khi nó đếm về 0 -> hết dãy

- so sánh giá trị $v1 (giá trị phần tử trong dãy vừa pop ra) và $a0 (giá trị max hiện tại) để xem có cập nhật max là $a0 hay tiếp tục lặp

- sau khi kết thúc hàm thì đưa con trỏ stack về lại với dãy ban đầu nên -32 giá trị -> thay đổi từ 0x7fffefdc tăng dần 4 đơn vị sau mỗi lần lặp và khi kết thúc -32 thì nó trở lại địa chỉ ban đầu trước khi pop là 0x7fffefdc

+ thủ tục min: hoạt động tương tự max như thay vào đó là sự tham gia của thanh ghi $a2 lưu min, $t4 lưu index

-> vì các thủ tục không lồng nhau nên thanh ghi $ra ko phải lưu lại ra bộ nhớ để restore nên nó thay đổi theo địa chỉ con trỏ pc.

Cụ thể:

Lệnh	Thanh ghi $ra	Giá trị sau khi thực thi
jal push	0x00000000	0x0040002c = pc địa chỉ lệnh đầu thủ tục push
jal max	0x0040002c	0x00400034 = pc địa chỉ lệnh đầu thủ tục max
jal min	0x00400034	0x0040003c = pc địa chỉ lệnh đầu thủ tục min