Báo cáo thực hành KTMT tuần 7

Họ và tên: Đỗ Gia Huy

MSSV: 20215060

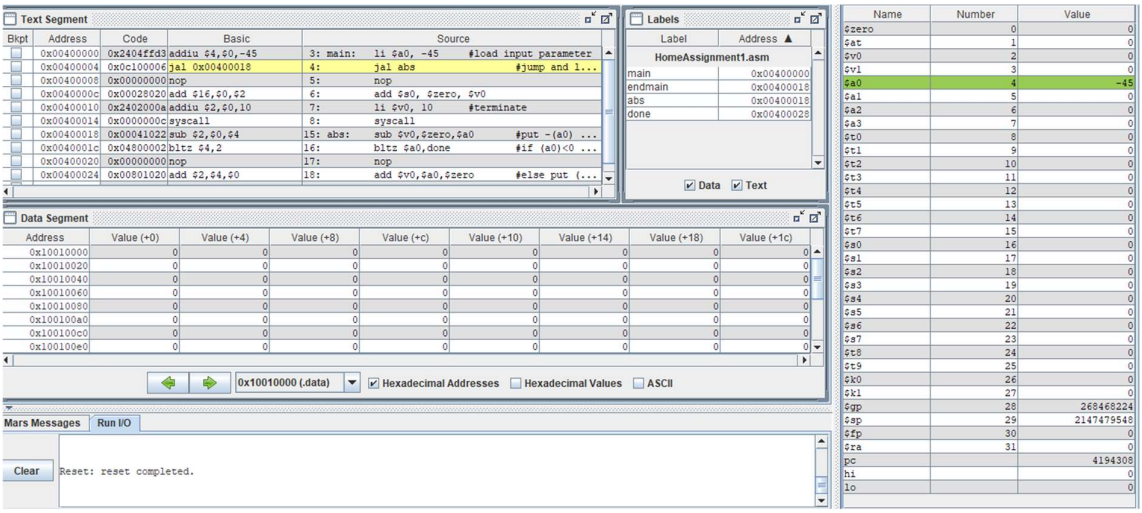
Assignment 1

Code:

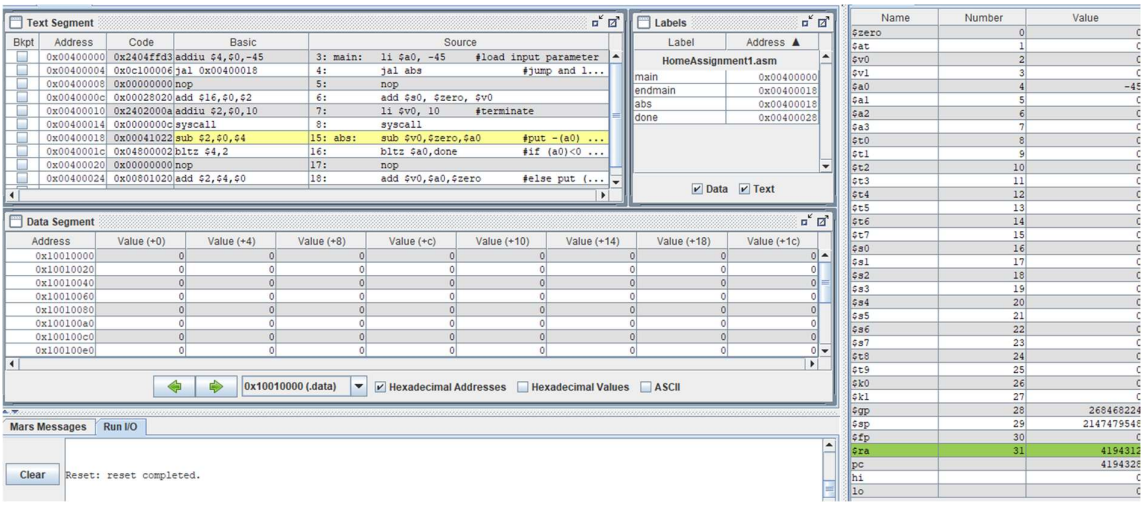
```
#Laboratory Exercise 7 Home Assignment 1
.text
main:
        li      $a0, -45        #load input parameter
        jal     abs             #jump and link to abs procedure
        nop
        add     $s0, $zero, $v0
        li      $v0, 10         #terminate
        syscall
endmain:
#---------------------------------------------------------------
# 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
```

Thực thi:

- **Trước khi chạy lệnh:** `jal abs`

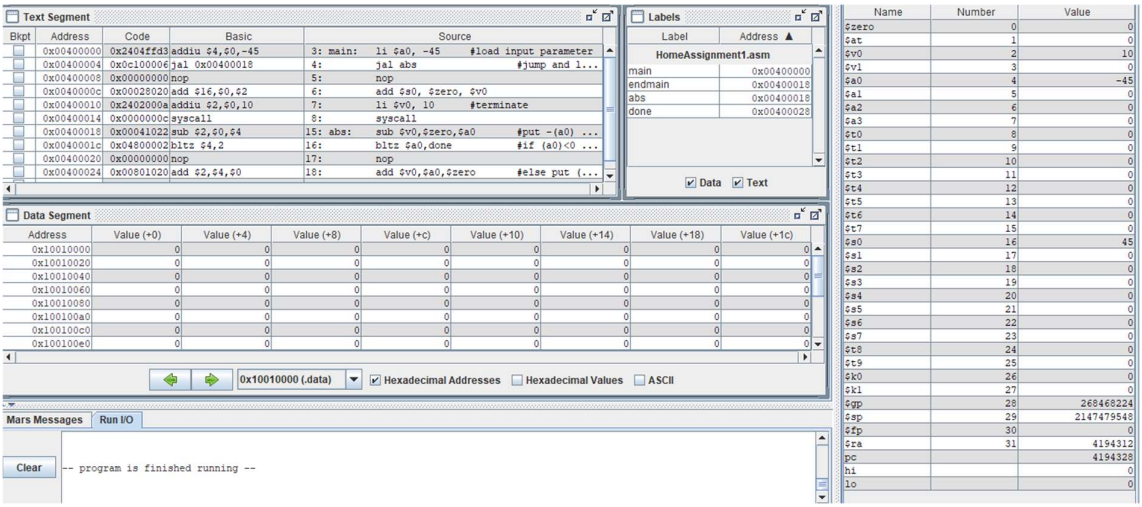


- **Sau khi chạy lệnh:** `jal abs`



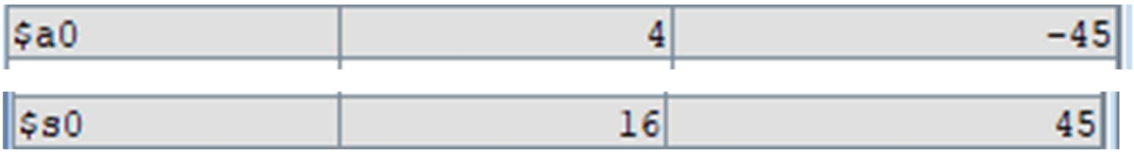
Result:

- **Kết quả sau khi thực thi xong toàn bộ chương trình:**



- **Nhận xét:**

- Khi chạy lệnh `jal abs` (địa chỉ lệnh 0x00400004) thì thanh ghi $ra được gán bằng địa chỉ của câu lệnh tiếp theo (0x00400008) và thanh ghi pc được gán bằng địa chỉ 0x00400018 (địa chỉ tại nhãn *abs*)
- Kết quả cuối cùng ta lấy được giá trị tuyệt đối của số được nạp vào trong thanh ghi $a0 và ghi kết quả đó vào thanh ghi $s0



Assignment 2

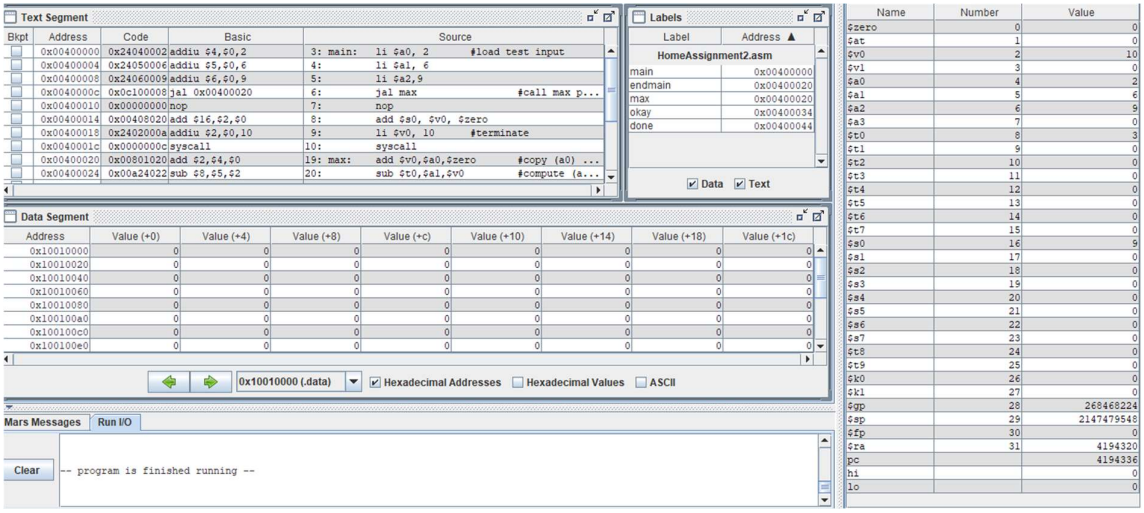
Code:

```
#Laboratory Exercise 7, Home Assignment 2
.text
main:
    li      $a0, 2          #load test input
    li      $a1, 6
    li      $a2, 9
    jal     max             #call max procedure
    nop
    add     $s0, $v0, $zero
    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
```

Result:

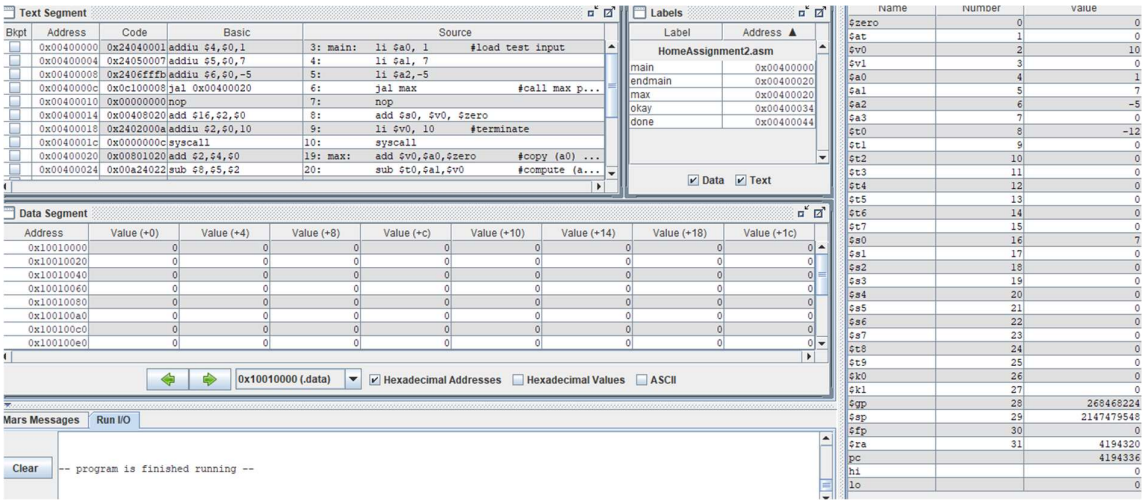
- **Kết quả chạy chương trình mẫu:**



Thanh ghi $s0 = 9 là thanh ghi chứa giá trị lớn nhất

- **Sau khi sửa giá trị các thanh ghi:**

$a0 = 1, $a1 = 7, $a2 = -5



Kết quả thu được: $s0 = 7

- **Nhận xét:**

- Khi chạy lệnh `jal` thì thanh ghi $ra được gán bằng giá trị của địa chỉ của câu lệnh tiếp theo sau jal trong nhãn main. Thanh ghi pc được gán bằng địa chỉ của nhãn max để câu lệnh tiếp tục được thực hiện bắt đầu từ nhãn max. Sau khi chạy đến jr $ra thì pc được gán bằng địa chỉ trong $ra (địa chỉ của nop)

Assignment 3

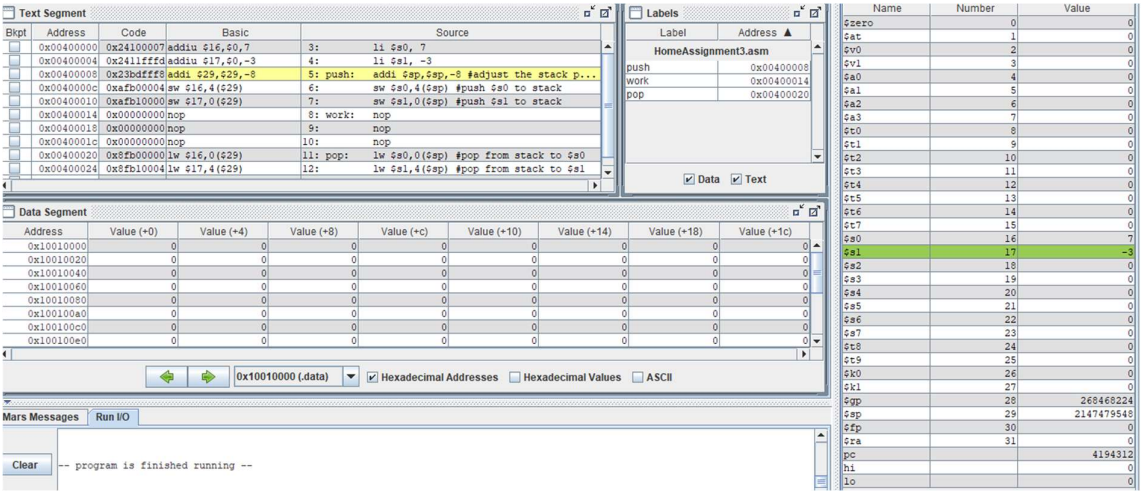
Code:

```
#Laboratory Exercise 7, Home Assignment 3
.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
```

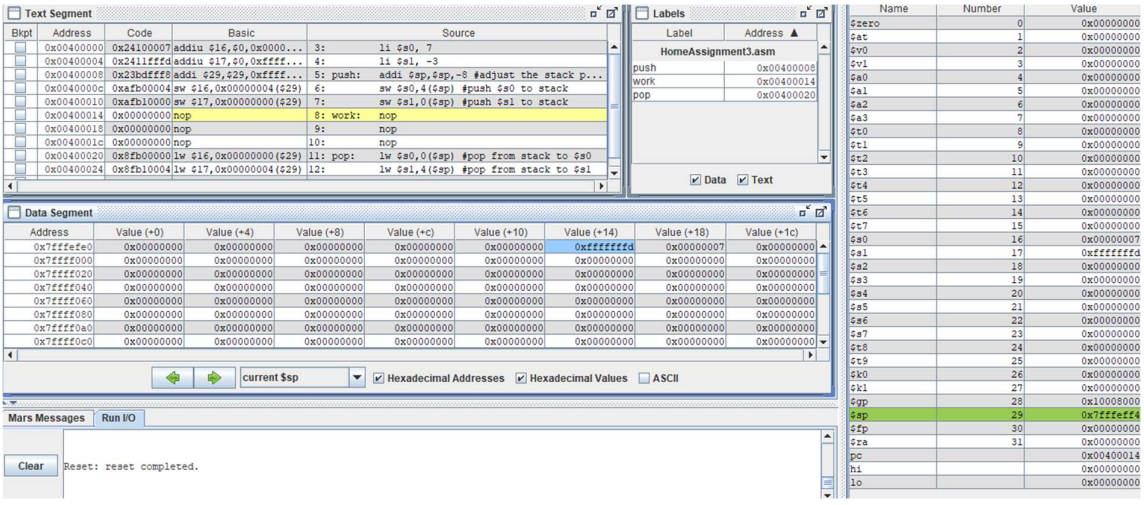
Result:

- **Trước khi chạy lệnh `addi` ở nhãn push:**



$sp = 0x7fffeffc

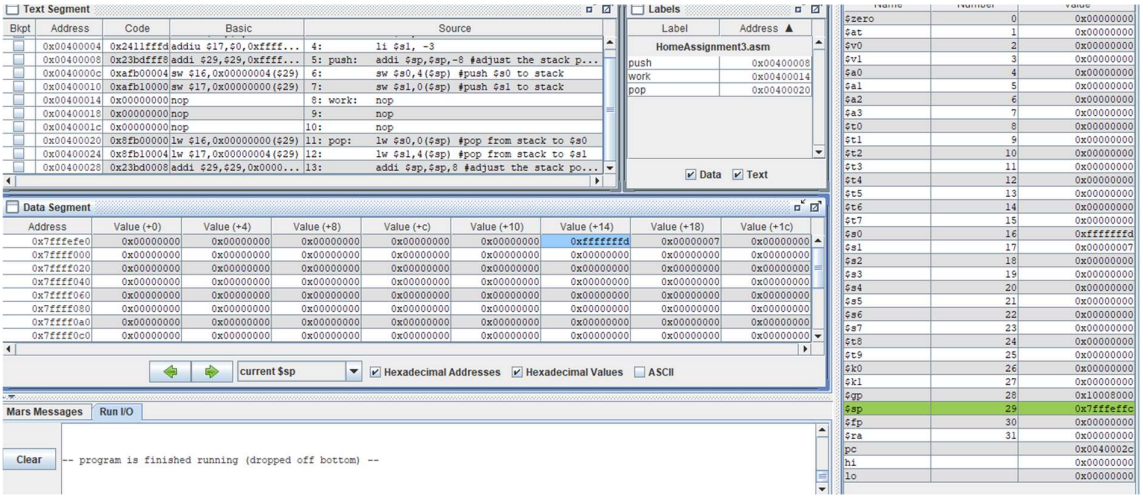
- **Sau khi chạy lệnh `addi` ở nhãn push:**



$sp = 0x7fffeff4

- Thanh ghi $sp được giảm đi 8 byte (tức là có sự cấp phát cho bộ nhớ stack 8 byte)
- Sau đó lần lượt ghi giá trị trong $s0 vào $sp + 4, giá trị trong $s1 vào $sp + 0

- **Sau khi chạy lệnh `addi` ở nhãn pop:**



- Thực hiện đổi chỗ hai số bằng cách load giá trị tại địa chỉ $sp + 0 vào $s1, load giá trị tại địa chỉ $sp + 4 vào $s1
- Lệnh add $sp, $sp, 8 (giúp giải phóng stack, trả lại đỉnh stack)

Assignment 4

Code tính giai thừa có sử dụng thanh ghi $fp:

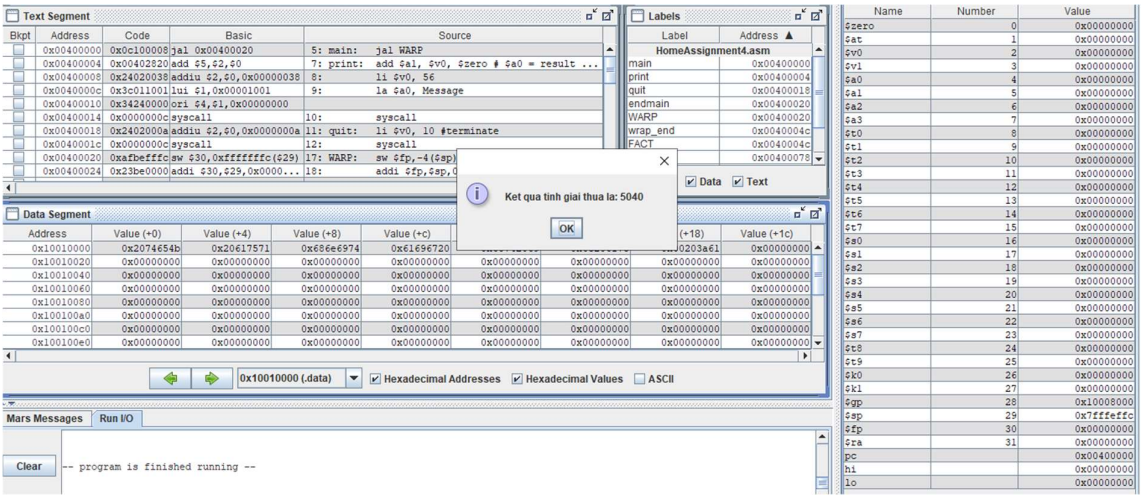
```
#Laboratory Exercise 7, Home Assignment 4

.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      $a0,7 #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
```

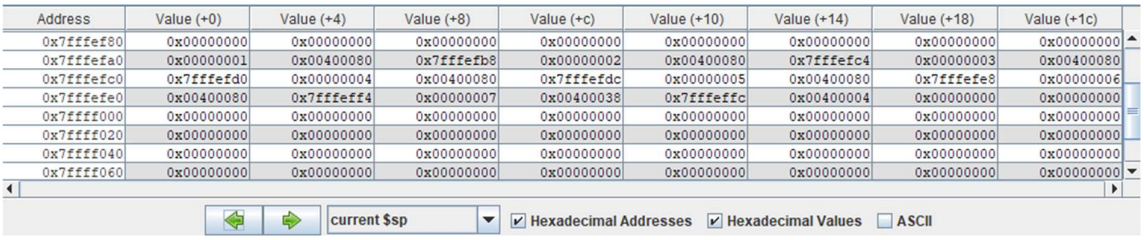
```
        sw      $a0,0($sp) #save $a0 register


        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:
```

Thực thi, kết quả và nhận xét:

Với n = 7, tức là $a0 = 7:



- Kết quả chương trình: 7! = 5040, kết quả thực thi đúng với lý thuyết
- Sự thay đổi của thanh ghi $sp:



Bảng thể hiện giá trị ngăn xếp:

Chạy trường hợp n=3 (Tính 3!):

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 = 0x7fffefe4
0x7fffefd4	$ra = 0x00400080
0x7fffefd0	$a0 = 0x00000001

Code tính giai thừa không sử dụng thanh ghi $fp:

```
.data
    message1: .asciiz "Nhap so N (N khong am): "
    message2: .asciiz "Ket qua tinh giai thua la: "
```

```
.text
main: li      $v0, 51
      la      $a0, message1          # Hien message1 & Nhap so N
      syscall
      bltz    $a0, main          # Kiem tra N khong am
      nop

      jal     fact
      nop

      add     $a1, $v0, $zero     # $a1 = ket qua
      li      $v0, 56
      la      $a0, message2          # Hien message2 va ket qua
      syscall

      li      $v0, 10            # Exit
      syscall

fact: addi    $sp, $sp, -8  # Cap phat Stack
      sw      $ra, ($sp)
      sw      $s0, 4($sp)

      li      $v0, 1         # v0 = 1
      beq     $a0, $zero, endfact          # N = 0 branch to the endfact
      nop
      add     $s0, $a0, $zero     # s0 = N
      addi    $a0, $a0, -1 # N = N - 1
      jal     fact
```

```
        nop
        mul   $v0, $v0, $s0        # N! = N * (N-1) * (N-2) *...* 1
endfact:    lw    $ra, ($sp)
            lw    $s0, 4($sp)
            addi  $sp, $sp, 8          # Giai phong Stack
            jr    $ra
```

Assignment 5

Code:

```
# Laboratory Exercise 7, Assignment 5
.data
        largest: .asciiz "Largest: "
        smallest: .asciiz "\nSmallest: "
        comma: .asciiz ", "
.text
main: li    $s0, 5
        li    $s1, -12
        li    $s2, 56
        li    $s3, 12
        li    $s4, 87
        li    $s5, -2
        li    $s6, -343
        li    $s7, 23

        jal   saveNumbers
        nop

        li    $v0, 4         # Print message Largest
        la    $a0, largest
```

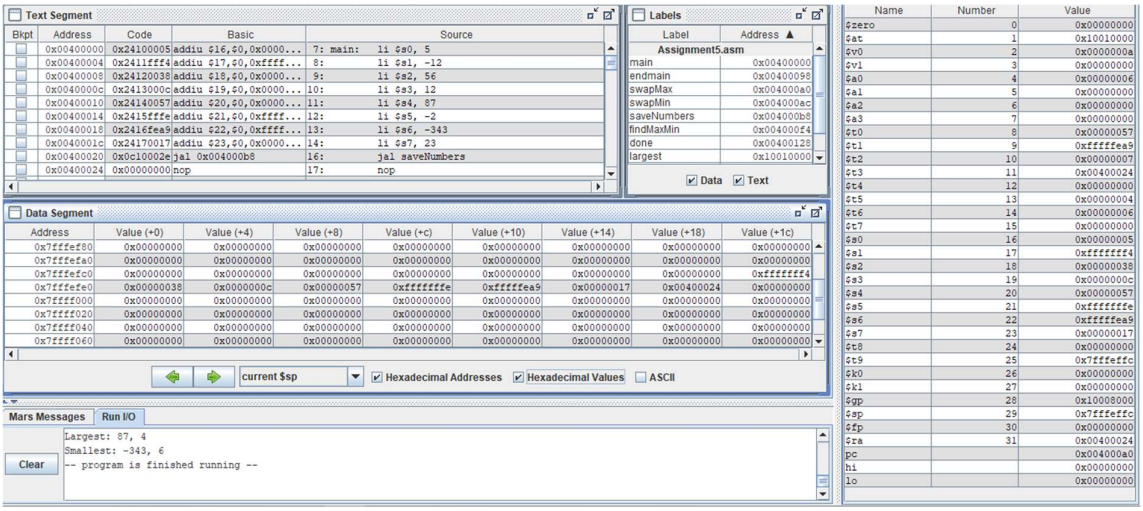
```
syscall

add    $a0, $t0, $zero      # Print Max
li       $v0, 1
syscall

li       $v0, 4          # Print message Comma
la       $a0, comma
syscall

add    $a0, $t5, $zero
li       $v0, 1          # Print the register number of Max
syscall

li       $v0, 4          # Print message Smallest
la       $a0, smallest
syscall

add    $a0, $t1, $zero      # Print Min
li       $v0, 1
syscall

li       $v0, 4          # Print message Comma
la       $a0, comma
syscall

add    $a0, $t6, $zero
li       $v0, 1          # Print the register number of Min
```

```
        syscall
endmain:    li      $v0, 10             # Exit
            syscall
#---------------------------------------------------------------------
# Return $t0 = Max
# Return $t1 = Min
# Index of Max = $t5
# Index of Min = $t6
#return $v0 the largest value
#---------------------------------------------------------------------
swapMax:
        add   $t0,$t3,$zero
        add   $t5,$t2,$zero
        jr    $ra
swapMin:
        add   $t1,$t3,$zero
        add   $t6,$t2,$zero
        jr    $ra
saveNumbers:
        add   $t9,$sp,$zero     # Save address of origin $sp
        addi  $sp,$sp, -32
        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)
```

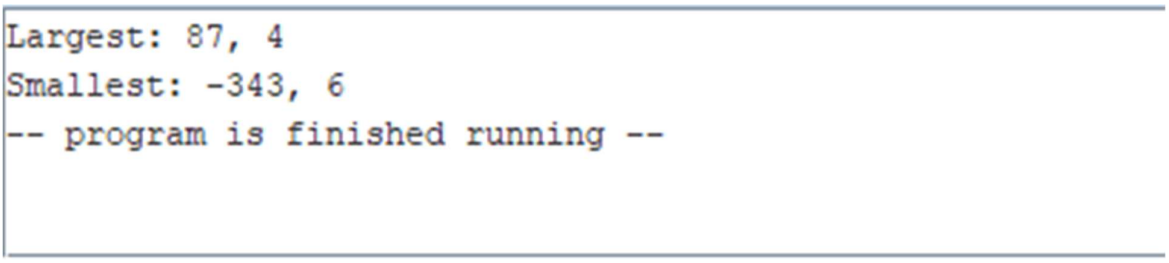
```
        sw      $ra, 28($sp)            # Save $ra for main
        add     $t0,$s0,$zero           # Max = $s0
        add     $t1,$s0,$zero           # Min = $s0
        li      $t5, 0                  # Index of Max to 0
        li      $t6, 0                  # Index of Min to 0
        li      $t2, 0                  # i = 0
findMaxMin:
        addi    $sp,$sp,4
        lw      $t3,-4($sp)
        sub     $t4, $sp, $t9
        beq     $t4,$zero, done         # If $sp = $fp branch to the 'done'
        nop
        addi    $t2,$t2,1               # i++
        sub     $t4,$t0,$t3
        bltzal  $t4, swapMax            # If $t3 > Max branch to the swapMax
        nop
        sub     $t4,$t3,$t1
        bltzal  $t4, swapMin        # If $t3 < Min branch to the swapMin
        nop
        j       findMaxMin          # Repeat
done:
        lw      $ra, -4($sp)
        jr      $ra                 # Return to calling program
```

Result:

- **Kết quả các thanh ghi sau khi thực hiện xong chương trình trên:**



- **Kết quả màn hình Run I/O:**



~ THE END ~