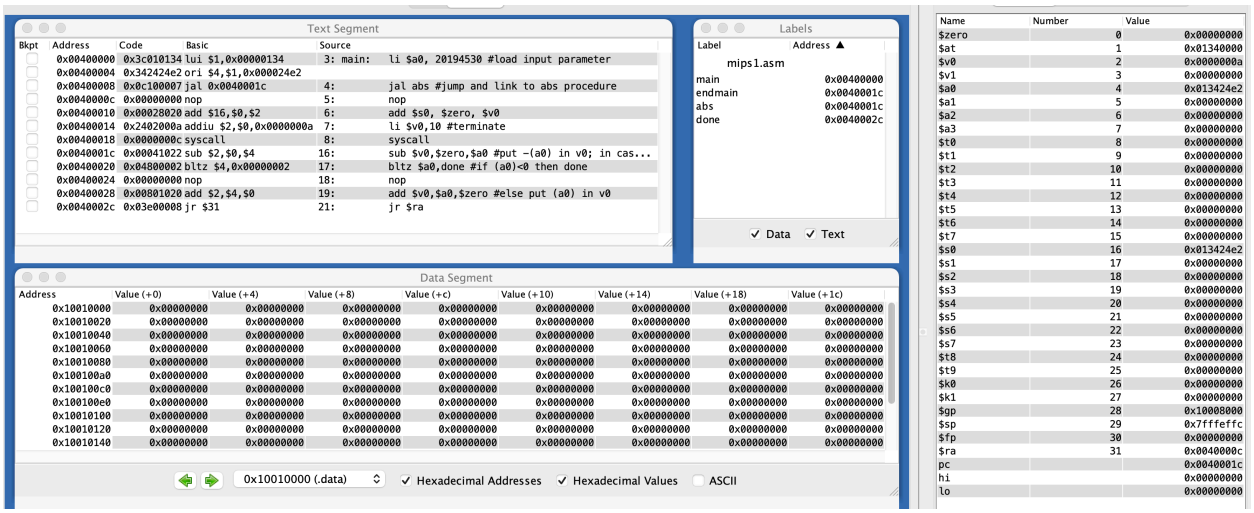
# Laboratory Exercise 7
## Procedure calls, stack and parameters


# Nguyễn Hải Dương - 20194530


### Assignment 1

```
1  #Laboratory Exercise 7 Home Assignment 1
2  .text
3  main:   li $a0, 20194530 #load input parameter
4          jal abs #jump and link to abs procedure
5          nop
6          add $s0, $zero, $v0
7          li $v0,10 #terminate
8          syscall
9  endmain:
10 #---------------------------------------------------------------
11 # function abs
12 # param[in] $a1 the interger need to be gained the absolute value
13 # return $v0 absolute value
14 #---------------------------------------------------------------
15 abs:
16         sub $v0,$zero,$a0 #put -(a0) in v0; in case (a0)<0
17         bltz $a0,done #if (a0)<0 then done
18         nop
19         add $v0,$a0,$zero #else put (a0) in v0
20 done:
21         jr $ra
```

**Chương trình:**




## Ý nghĩa các thanh ghi:

| | |
|---|---|
| $a0 | Đối số đầu vào |
| $v0 | Lưu giá trị tuyệt đối của đối số |
| $s0 | Lưu giá trị tuyệt đối để nhường thanh ghi $v0 cho syscall |
| $ra | Return, kết thúc procedure abs |

## Ý nghĩa chương trình:

- Input: một số nguyên ($a0) – MSSV: 20194530
- Ouput: giá trị tuyệt đối của số đó ($s0)
- Thuật toán:
  + gọi thủ tục abste
  + lưu số đối của số nguyên –(a0) vào thanh ghi $v0, nếu số đó là số âm -> done -> return về vị trí sau lệnh jal
  + nếu không thì ta lưu số nguyên đó (a0) vào $v0 -> done -> return về sau lệnh jal
  + lưu giá trị ở $v0 vào $s0 sau đó sử dụng $v0 để gọi syscall
- Giải thích các lệnh quan trọng:
- Jal <target> : nhảy đến target, đồng thời set thanh ghi $ra bằng thanh ghi $pc ( tức là lệnh tiếp theo ngay sau lệnh jal )
- Jr <register> : nhảy đến địa chỉ lưu trong thanh ghi, lệnh jal và jr $ra thường đi kèm với nhau để tạo nên một thủ tục

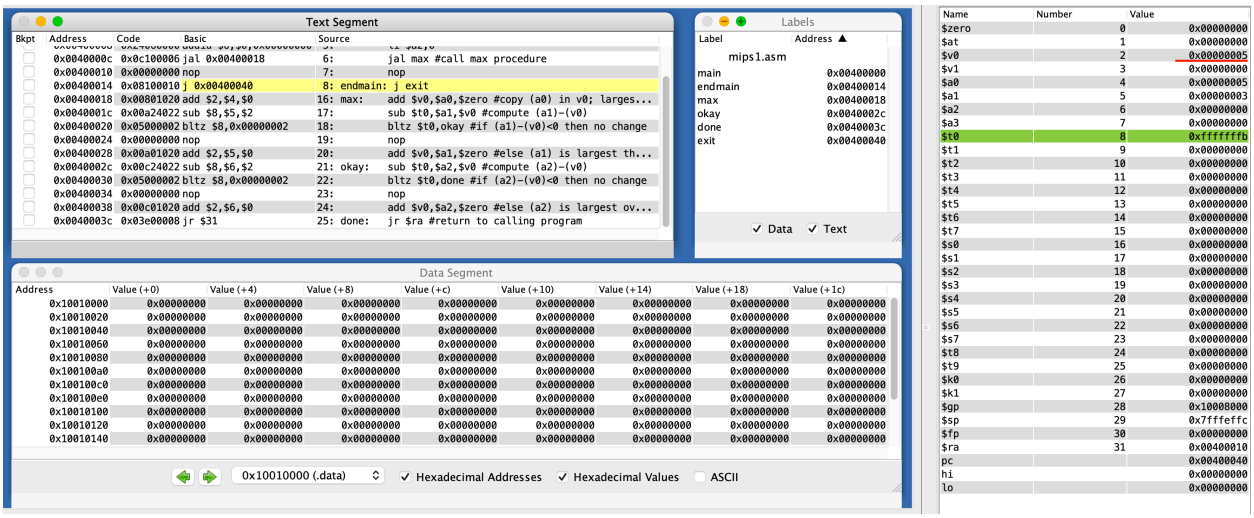
## Debug từng dòng:

| Step | $pc | Giá trị thanh ghi thay đổi |
|---|---|---|
| 1 | 0x00400004 | $at = 0x01340000 |
| 2 | 0x00400008 | $a0 = 0x013424e2 |
| 3 | 0x0040001c | $ra = 0x0040000c |
| 4 | 0x004000020 | $v0 = 0xfecbdb1e |
| 5 | 0x004000014 | $s0 = 0x013424e2 |
| 6 | 0x004000018 | $v0 = 0x0000000a |
| 7 | 0x00400001c | $v0 = 0x0000000a |

## Assignment 2

```
1  #Laboratory Exercise 7, Home Assignment 2
2  .text
3  main:   li $a0,5 #load test input
4          li $a1,3
5          li $a2,0
6          jal max #call max procedure
7          nop
8  endmain: j exit
9  #------------------------------------------------------------------
10 #Procedure max: find the largest of three integers
11 #param[in] $a0 integers
12 #param[in] $a1 integers
13 #param[in] $a2 integers
14 #return $v0 the largest value
15 #------------------------------------------------------------------
16 max:    add $v0,$a0,$zero #copy (a0) in v0; largest so far
17         sub $t0,$a1,$v0 #compute (a1)-(v0)
18         bltz $t0,okay #if (a1)-(v0)<0 then no change
19         nop
20         add $v0,$a1,$zero #else (a1) is largest thus far
21 okay:   sub $t0,$a2,$v0 #compute (a2)-(v0)
22         bltz $t0,done #if (a2)-(v0)<0 then no change
23         nop
24         add $v0,$a2,$zero #else (a2) is largest overall
25 done:   jr $ra #return to calling program
26
27 exit:
```

## Chương trình:




## Ý nghĩa các thanh ghi:

| | |
|---|---|
| $a0 | Đối số đầu vào |
| $a1 | Đối số đầu vào |
| $a2 | Đối số đầu vào |
| $v0 | Lưu số lớn nhất |
| $t0 | Biến sử dụng cho phép so sánh <0 ( hiệu của hai số) |

## Ý nghĩa chương trình:

- Input: 3 số nguyên ($a0,$a1,$a2)
- Ouput: số nguyên lớn nhất trong 3 số ($v0)
- Thuật toán:

  + gọi thủ tục max :

    - gán $v0 = $a0 ( tạm coi $a0 là max)
    - so sánh $v0 với $a1, nếu $a1 lớn hơn thì đưa $a1 lên làm max
    - ngược lại tiếp tục so sánh $v0 với $a2 nếu $a2 lớn hơn thì đưa $a2 lên làm max

    - return về vị trí sau lệnh jal

## - Debug từng dòng:

| Step | $pc | Giá trị thanh ghi thay đổi |
|---|---|---|
| 1 | 0x00400004 | $a0 = 0x00000005 |

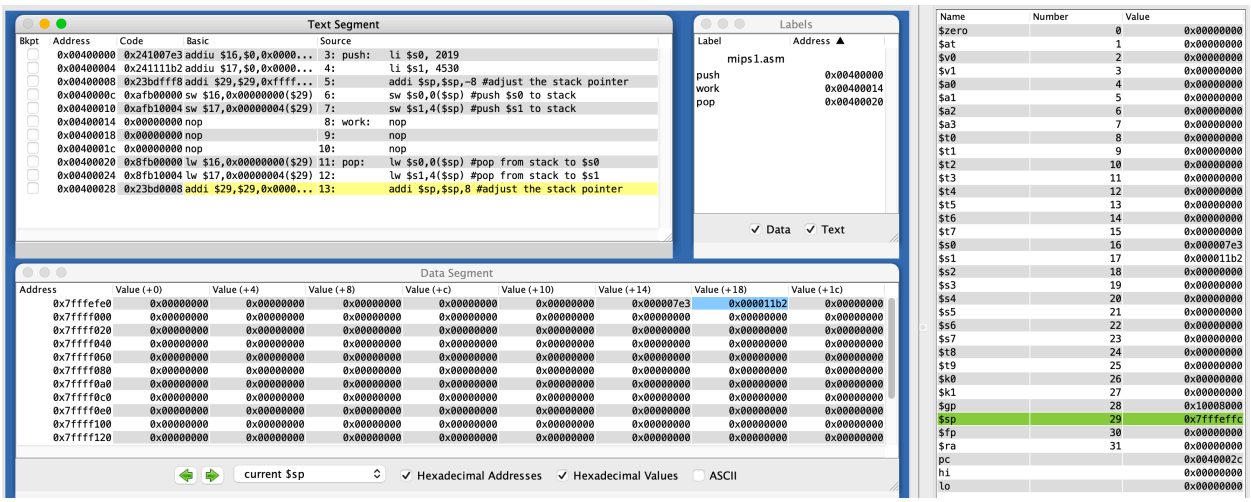
| | | |
|---|---|---|
| 2 | 0x00400008 | $a1 = 0x00000003 |
| 3 | 0x0040000c | $a2 = 0x00000000 |
| 4 | 0x004000018 | $ra = 0x00000010 |
| 5 | 0x00400001c | $v0 = 0x00000005 |
| 6 | 0x004000020 | $t0 = 0xfffffffe |
| 7 | 0x00400002c | $t0 = 0xfffffffe |
| 8 | 0x004000030 | $t0 = 0xfffffffb |
| 9 | 0x00400003c | $t0 = 0xfffffffb |
| 10 | 0x004000010 | $t0 = 0xfffffffb |
| 11 | 0x004000014 | $t0 = 0xfffffffb |
| 12 | 0x00400003c | $t0 = 0xfffffffb |
| 13 | 0x004000010 | $t0 = 0xfffffffb |

**Kết quả:** Số lớn nhất trong {5, 3, 0} là 5

## Assignment 3

```
1  #Laboratory Exercise 7, Home Assignment 3
2  .text
3  push:   li $s0, 2019
4          li $s1, 4530
5          addi $sp,$sp,-8 #adjust the stack pointer
6          sw $s0,0($sp) #push $s0 to stack
7          sw $s1,4($sp) #push $s1 to stack
8  work:   nop
9          nop
10         nop
11 pop:    lw $s0,0($sp) #pop from stack to $s0
12         lw $s1,4($sp) #pop from stack to $s1
13         addi $sp,$sp,8 #adjust the stack pointer
14
```

**Chương trình:**




## Ý nghĩa các thanh ghi:

| | |
|---|---|
| $s0 | Đối số đầu vào (4 số đầu MSSV) |
| $s1 | Đối số đầu vào (4 số cuối MSSV) |
| $sp | Đóng vai trò như một ngăn xếp để lưu các giá trị |

## Ý nghĩa chương trình:

- Input: 2 số nguyên ($s0,$s1)
- Ouput: đổi vị trí 2 số nguyên đó trong 2 thanh ghi $s0, $s1
- Thuật toán:
  + khai báo một stack chứa được 2 số nguyên
  + lưu lần lượt 2 số nguyên vào stack
  + pop chúng ra theo thứ tự ngược lại vào địa chỉ của 2 thanh ghi $s1 và $s0

| Step | $pc | Giá trị thanh ghi thay đổi |
|---|---|---|
| 1 | 0x00400004 | $s0 = 0x000007e3 |
| 2 | 0x00400008 | $s1 = 0x000011b2 |
| 3 | 0x0040000c | $sp = 0x7fffeff4 |
| 4 | 0x00400010 | $sp = 0x7fffeff4 |
| 5 | 0x00400014 | $sp = 0x7fffeff4 |
| 6 | 0x00400018 | $sp = 0x7fffeff4 |

| | | |
|---|---|---|
| 7 | 0x0040001c | $sp = 0x7fffeff4 |
| 8 | 0x00400020 | $sp = 0x7fffeff4 |
| 9 | 0x00400024 | $sp = 0x7fffeff4 |
| 10 | 0x00400028 | $s1 = 0x000011b2 |
| 11 | 0x0040002c | $sp = 0x7fffeffc |

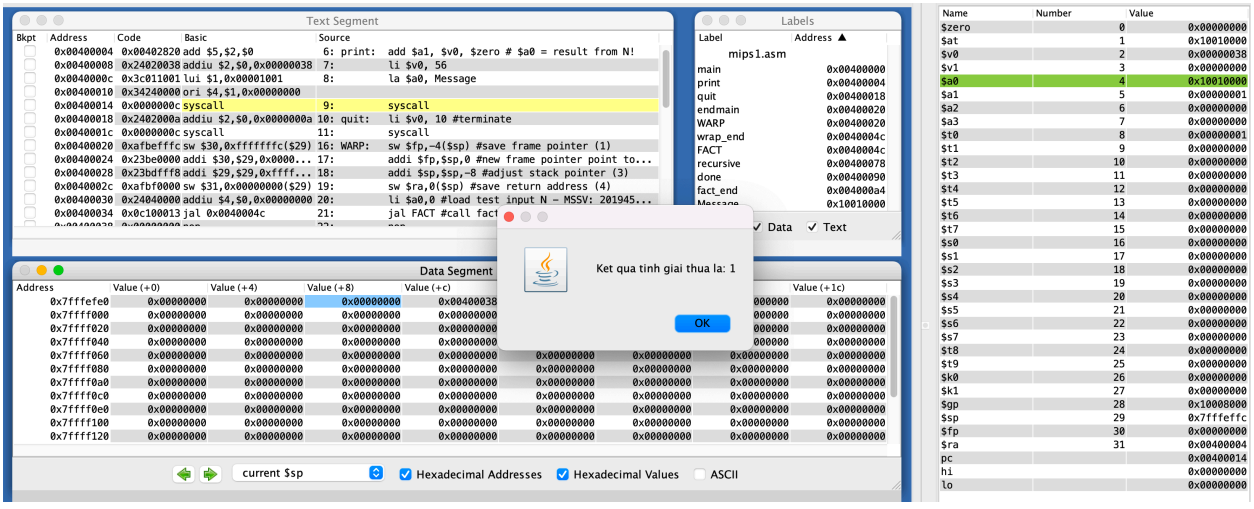
## Home Assignment 4

```
1  #Laboratory Exercise 7, Home Assignment 4
2  .data
3  Message: .asciiz "Ket qua tinh giai thua la: "
4  .text
5  main:   jal WARP
6  print:  add $a1, $v0, $zero # $a0 = result from N!
7          li $v0, 56
8          la $a0, Message
9          syscall
10 quit:   li $v0, 10 #terminate
11         syscall
12 endmain:
13 #---------------------------------------------------------------
14 #Procedure WARP: assign value and call FACT
15 #---------------------------------------------------------------
16 WARP:   sw $fp,-4($sp) #save frame pointer (1)
17         addi $fp,$sp,0 #new frame pointer point to the top (2)
18         addi $sp,$sp,-8 #adjust stack pointer (3)
19         sw $ra,0($sp) #save return address (4)
20         li $a0,0 #load test input N - MSSV: 20194530, get the last digit = 0
21         jal FACT #call fact procedure
22         nop
23         lw $ra,0($sp) #restore return address (5)
24         addi $sp,$fp,0 #return stack pointer (6)
25         lw $fp,-4($sp) #return frame pointer (7)
26         jr $ra
27 wrap_end:
28 #---------------------------------------------------------------
29 #Procedure FACT: compute N!
30 #param[in] $a0 integer N
31 #return $v0 the largest value
32 #---------------------------------------------------------------
33 FACT:   sw $fp,-4($sp) #save frame pointer
34         addi $fp,$sp,0 #new frame pointer point to stack's top
```

```
#-----------------------------------------------------------------------------
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:

```

## Chương trình:




**Kết quả:** 0! = 4 (0 là số cuối MSSV)

## Ý nghĩa các thanh ghi:

| | |
|---|---|
| $sp | Con trỏ stack pointer |
| $fp | Lưu con trỏ đến khung trang frame pointer |
| $a0 | Kết quả n! |
| $v0 | Lưu số lớn nhất, đồng thời sử dụng cho syscall |
| $ra | Thanh ghi chứa địa chỉ return |
| $t0 | Thanh ghi chứa biến so sánh làm điều kiện kết thúc |
| $v1 | Thanh ghi chứa giá trị lấy ra từ stack |
| $a1 | Lưu kết quả cuối cùng sử dụng cho hiển thị của lệnh syscall 56 |

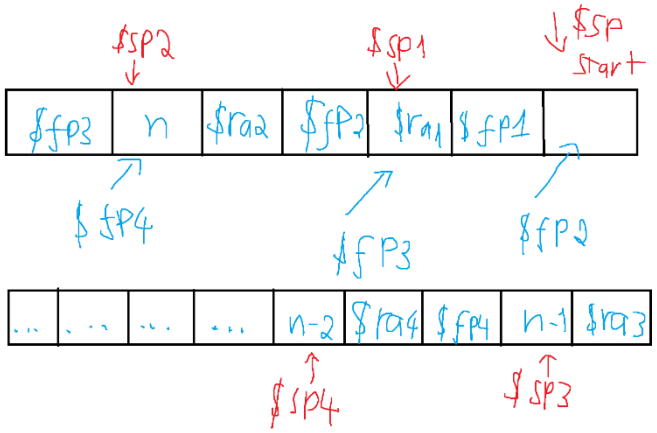
**Ý nghĩa chương trình:**

- Input: số nguyên n>0
- Ouput: n!
- Thuật toán:
  + gọi thủ tục wrap :
    - trong thủ tục này ta khai báo 2 vị trí cho khung trang và địa chỉ return
    - load giá trị n vào thanh ghi $a0
    - gọi đến thủ tục fact

  + thủ tục fact:
    - khai báo 3 vị trí cho khung trang, địa chỉ return và số nguyên dùng cho việc tính n!
    - Khung trang trỏ đến vị trí số nguyên trước đó được lưu lại ở mỗi lần lặp
    - Khi đã lưu xong các số nguyên dùng cho việc tính n! Thì nhảy đến done
- Khi nhảy đến nhãn done ta lần lượt pop ra các giá trị đã lưu ra từ trước để tính toán. Các giá trị thanh ghi $ra trước đó đã lưu trong stack giúp ta return về vị trí lệnh jal qua đó kết thúc từng thủ tục một và quay về với chương

trình chính (main). Các giá trị của thanh ghi $fp đưa ta đến từng khung trang, nơi mà mỗi vị trí tương ứng sẽ có 3 tham số n, $ra, $fp

*mô tả trong hình dưới



**Debug từng dòng:**

| 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 |
| … | … | … |

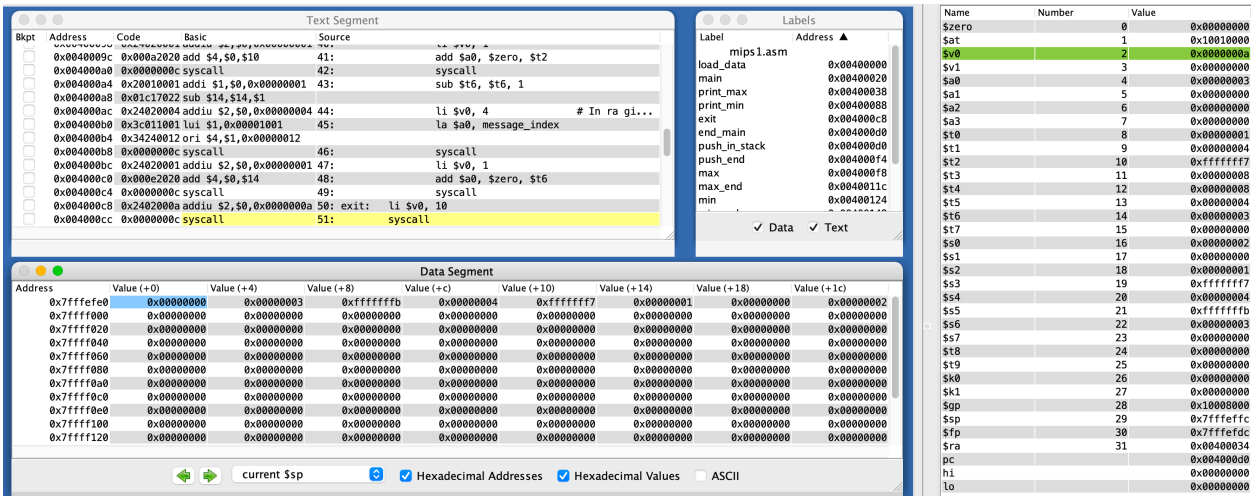
## Assignment 5

**Code:**

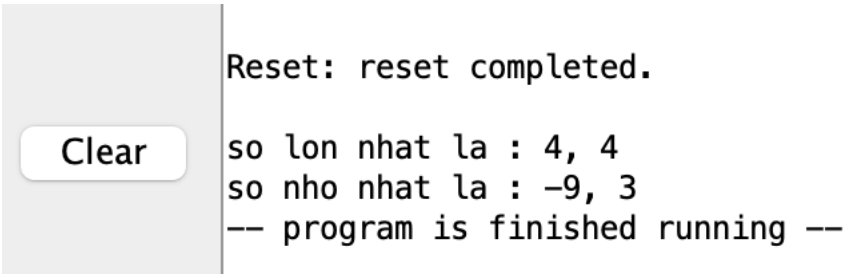
```
.data
message_max: .asciiz "so lon nhat la : "
message_index: .asciiz ", "
message_min: .asciiz "so nho nhat la : "
enter: .asciiz "\n"
.text
load_data:      li $s0, 2       # Nạp dữ liệu: 20194530
                li $s1, 0
                li $s2, 1
                li $s3, -9
                li $s4, 4
                li $s5, -5
                li $s6, 3
                li $s7, 0
main:   jal push_in_stack               # Goi ham luu cac gia tri s0 -> s7 vao stack
        nop
        jal max                         # Goi ham tim max
        nop
        jal min                         # Goi ham tim min
        nop
print_max:      li $v0, 4               # In ra giá trị max
                la $a0, message_max
                syscall
                li $v0, 1
                add $a0, $zero, $t1
                syscall
                sub $t5, $t5, 1
                li $v0, 4               # In ra giá trị max
                la $a0, message_index
                syscall
                li $v0, 1
                add $a0, $zero, $t5
                syscall
                li $v0, 4
                syscall
print_min:      li $v0, 4               # In ra giá trị min
                la $a0, message_min
                syscall
                li $v0, 1
                add $a0, $zero, $t2
                syscall
                sub $t6, $t6, 1
                li $v0, 4               # In ra giá trị max
                la $a0, message_index
                syscall
                li $v0, 1
                add $a0, $zero, $t6
                syscall
exit:   li $v0, 10
        syscall
end_main:
#-----------------------------------
push_in_stack:  sw $s0, 0($sp)
                sw $s1, -4($sp)
                sw $s2, -8($sp)
                sw $s3, -12($sp)
                sw $s4, -16($sp)
                sw $s5, -20($sp)
                sw $s6, -24($sp)
                sw $s7, -28($sp)
                add $fp, $sp, -32
push_end:       jr $ra
#-----------------------------------------------
max:    lw $v1, 0($sp)          # Lay du lieu ra tu stack pointer
        add $t3, $t3, 1         # $t3 luu index cua phan tu trong stack. $t3 = $t3 + 1
        add $sp, $sp, -4        # Dieu chinh stack pointer
```

```
max:    lw $v1, 0($sp)          # Lay du lieu ra tu stack pointer
        add $t3, $t3, 1         # $t3 luu index cua phan tu trong stack. $t3 = $t3 + 1
        add $sp, $sp, -4        # Dieu chinh stack pointer
        beq $sp, $fp, max_end   # stack pointer = frame pointer -> dừng hàm
        slt $t0, $v1, $t1       # v1 < t1 (t1 la max)
        bne $t0, $zero, max     # Neu v1 < t1 -> phan tu tiep theo
        add $t1, $zero, $v1     # max = v1
        add $t5, $zero, $t3     # index cua phan tu max
        j max
max_end:        add $sp, $sp, 32 #adjust stack pointer to the top of stack
                jr $ra
#---------------------------------------------
min:    lw $v1, 0($sp)
        add $t4, $t4, 1         # $t4 luu index cua phan tu trong stack. $t4 = $t4 + 1
        add $sp, $sp, -4 #adjust stack pointer
        beq $sp, $fp, min_end
        slt $t0, $t2, $v1
        bne $t0, $zero, min
        add $t2, $zero, $v1
        add $t6, $zero, $t4     # Index cua phan tu min
        j min
min_end:        add $sp, $sp, 32 #adjust stack pointer to the top of stack
                jr $ra
```

## Chương trình:




## Output:



**Debug từng dòng:**

| Step | $pc | Giá trị thanh ghi thay đổi |
|---|---|---|
| 1 | 0x00400004 | $s0 = 0x00000002 |
| 2 | 0x00400008 | $s1 = 0x00000000 |
| 3 | 0x0040000c | $s2 = 0x00000001 |
| 4 | 0x0040002c | $sp = 0x7fffeff4 |
| 4 | 0x00400010 | $s3 = 0xfffffff7 |
| 3 | 0x00400014 | $s4 = 0x00000004 |
| 4 | 0x00400018 | $s5 = 0xfffffffb |
| 7 | 0x0040001c | $s6 = 0x00000003 |
| 8 | 0x00400020 | $s7 = 0x0000000 |
| 9 | 0x00400088 | $ra = 0x00400024 |
| … | … | … |
| 4 | 0x004000ac | $fp = 0x7fffefdc |
| 3 | 0x004000b0 | $ra = 0x0040002c |
| 4 | 0x004000b4 | $v1 = 0x00000002 |
| 7 | 0x004000b8 | $sp = 0x7fffeff8 |
| 8 | 0x004000c0 | $t0 = 0x00000000 |
| 9 | 0x004000c8 | $t1 = 0x00000002 |
| 4 | 0x004000b4 | $v1 = 0x00000000 |
| 7 | 0x004000b8 | $sp = 0x7fffeff4 |
| 8 | 0x004000c0 | $t0 = 0x00000001 |
| 9 | 0x004000b4 | $v1 = 0x00000001 |
| 7 | 0x004000b8 | $sp = 0x7fffeff0 |

| | | |
|---|---|---|
| ... | ... | ... |