Lab 7

Procedure calls, stack, and parameters

Đinh Ngọc Khánh Huyền – 20225726

Sample Code 4

Trạng thái của ngăn xếp khi chạy n = 3 là

0x7fffeff8	$fp = 0x00000000
0x7fffeff4	$ra = 0x00400004
0x7fffeff0	$fp = 0x7fffeffc
0x7fffeffec	$ra = 0x00400038
0x7fffefe8	$a0 = 0x00000003
0x7fffefe4	$fp = 0x7fffeff4
0x7fffefe0	$ra = 0x00400080
0x7fffefdc	$a0 = 0x00000002
0x7fffefd8	$fp = 0x7fffefe4
0x7fffefd4	$ra = 0x00400080
0x7fffefd0	$a0 = 0x000000001

Explain:

Do thanh ghi $fp chứa con trỏ thanh ghi khung, chứa đỉnh của stack trước đó để có thể chạy thủ tục tiếp theo. Thanh ghi $ra chứa địa chỉ trả về. Còn thanh ghi $a0 chứa giá trị đang được gọi đệ quy.

Assignment 1

Code:

```
.data
largest:   .asciiz "Largest: "
smallest:  .asciiz "\nSmallest: "
comma:     .asciiz ", "
open:      .asciiz "("
close:     .asciiz ")"
```

```
.text

li $s0, 7
li $s1, 58
li $s2, 3
li $s3, 1
li $s4, 6
li $s5, 5
li $s6, -5
li $s7, 5

main:
   jal findmaxmin
   nop

quit:
   li $v0, 10
   syscall

end_main:

findmaxmin:
   add $fp, $sp, $zero     # fp points to the bottom of the stack
   addi $sp, $sp, -48      # make room for the 8 elements and 4 return values

   # load 8 parameters into the stack
```

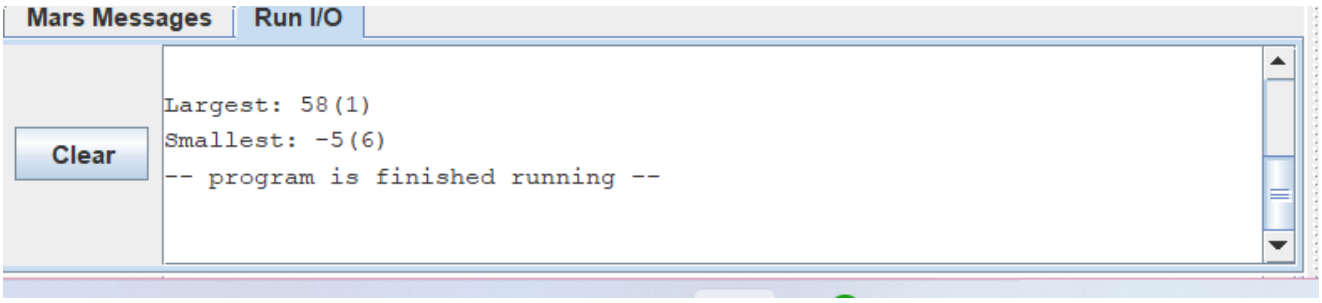
```
    sw $s0,  44($sp)
    sw $s1,  40($sp)
    sw $s2,  36($sp)
    sw $s3,  32($sp)
    sw $s4,  28($sp)
    sw $s5,  24($sp)
    sw $s6,  20($sp)
    sw $s7,  16($sp)

    addi $t1, $zero, 8     # last index
    lw $a0, 44($sp)        # a0 = first element
    addi $t0, $zero, 0     # t0 = 0

    # max and min so far is the first element
    sw $a0, 12($sp)
    sw $t0,  8($sp)
    sw $a0,  4($sp)
    sw $t0,  0($sp)
    addi $t0, $t0, 1       # t0 = 1

loop:
    bge $t0, $t1, end_loop   # end loop if t0 reaches the last element
    addi $t2, $t0, 1         # t2 = t0 + 1, when we take fp - 4 * (t0 + 1) we get a[t0]
    add $t2, $t2, $t2        # t2 = 2 * t0 + 2
    add $t2, $t2, $t2        # t2 = 4 * t0 + 4
    sub $t2, $fp, $t2        # t2 = address(a[i])
```

```
    lw $a0, 0($t2)          # a0 = a[i]
    lw $a1, 12($sp)          # a1 = current max
    lw $a2,  4($sp)         # a2 = current min

max:
    ble $a0, $a1, min       # if a0 <= a1 (a[i] <= current max), skip to check if a[i] is the smallest value
    sw $a0, 12($sp)          # else save the new max into the stack
    sw $t0,  8($sp)         # and save the max index into the stack

min:
    bge $a0, $a2, next       # if a0 >= a1 (a[i] >= current min), skip to next element
    sw $a0,  4($sp)          # else save the new min into the stack
    sw $t0,  0($sp)         # and save the min index into the stack

next:
    addi $t0, $t0, 1        # t0++
    j loop
    nop

end_loop:

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

```
syscall

lw $a0, 12($sp)
li $v0, 1
syscall

li $v0, 4              # Print message Open
la $a0, open
syscall

lw $a0,  8($sp)
li $v0, 1
syscall

li $v0, 4              # Print message Close
la $a0, close
syscall

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

lw $a0,  4($sp)
li $v0, 1
syscall
```

```
    li $v0, 4           # Print message open
    la $a0, open
    syscall

    lw $a0, 0($sp)
    li $v0, 1
    syscall

    li $v0, 4           # Print message close
    la $a0, close
    syscall

end_print:
return:
    jr $ra
```

Result:



Explain: giải thích các câu lệnh trong hàm findmaxmin:

- add $fp, $sp, $zero: Thiết lập con trỏ khung ($fp) để trỏ vào đáy của ngăn xếp.
- addi $sp, $sp, -48: Cấp phát không gian trên ngăn xếp cho 8 phần tử và 4 giá trị trả về
- Thiết lập chỉ số vòng lặp và khởi tạo giá trị max và min với phần tử đầu tiên của list.

- Lặp qua danh sách các phần tử, so sánh mỗi phần tử với các giá trị max và min hiện tại và cập nhật.
- In ra giá trị max và min và vị trí của chúng

Assignment 2

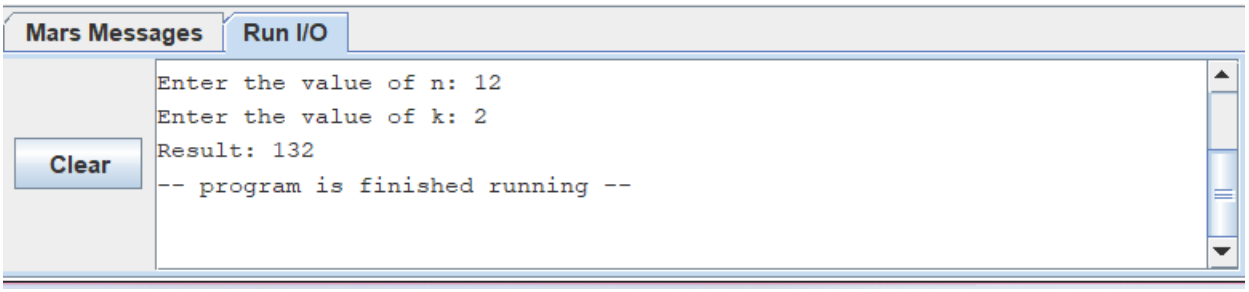
Code:

```
.data
prompt: .asciiz "Enter the value of n: "
prompt2: .asciiz "Enter the value of k: "
result: .asciiz "Result: "

.text
.globl main

main:
    # Print prompt for n
    li $v0, 4
    la $a0, prompt
    syscall

    # Read n
    li $v0, 5
    syscall
    move $s0, $v0    # $s0 = n

    # Print prompt for k
    li $v0, 4
```

```
la $a0, prompt2
syscall

# Read k
li $v0, 5
syscall
move $s1, $v0    # $s1 = k

# Calculate n!
li $t0, 1        # $t0 = n!
li $t1, 1        # counter for n!
loop1:
   bgt $t1, $s0, end_loop1   # exit loop if counter > n
   mul $t0, $t0, $t1         # $t0 = $t0 * $t1
   addi $t1, $t1, 1          # increment counter
   j loop1

end_loop1:

# Calculate (n-k)!
sub $t2, $s0, $s1       # $t2 = n - k
li $t3, 1               # $t3 = (n-k)!
li $t4, 1               # counter for (n-k)!
loop2:
   bgt $t4, $t2, end_loop2    # exit loop if counter > (n-k)
   mul $t3, $t3, $t4          # $t3 = $t3 * $t4
```

```
    addi $t4, $t4, 1          # increment counter
    j loop2

end_loop2:

# Calculate result = n!/(n-k)!
div $t0, $t0, $t3
mflo $t0            # quotient is stored in $t0

# Print result
li $v0, 4
la $a0, result
syscall

li $v0, 1
move $a0, $t0
syscall

# Exit program
li $v0, 10
syscall
```

Result:



Explain:

Ta sử dụng 2 vòng lặp.

1 vòng lặp từ 1 đến n để tính n!

1 vòng lặp từ 1 đến (n-k) để tính (n-k)!

Sau đó tính giá trị biểu thức n!/(n-k)!