Lớp: CE119.O11.3
Tên: Trần Ngọc Ánh *MSSV: 22520077*

BÁO CÁO LAB04 MÔN THỰC HÀNH KIẾN TRÚC MÁY TÍNH

(CE119-LAB04)

Câu 1: Thực hành với thủ tục

- **Chạy từng bước và theo dõi sự thay đổi của thanh ghi PC, $ra, $sp, $fp:**
  - *Hình 1: nhập từ cửa sổ I/O một số nguyên vào thanh ghi $s0*

```
        .data
prompt: .asciiz "Enter one number: "

        .text
main:   jal getInt
        move $s0, $v0
        j exit

getInt: li $v0, 4
        la $a0, prompt
        syscall

        li $v0, 5
        syscall
        jr $ra

exit:
```

    - Thanh ghi pc và $fp không thay đổi trực tiếp trong đoạn mã này.
    - $ra được sử dụng để lưu địa chỉ trả về sau lệnh 'jal', giúp quay lại vị trí sau lệnh gọi hàm.
    - $sp thay đổi khi cấp phát và giải phóng không gian trên ngăn xếp.
  - *Hình 2: Phép toán (a + b) – (c + d)*

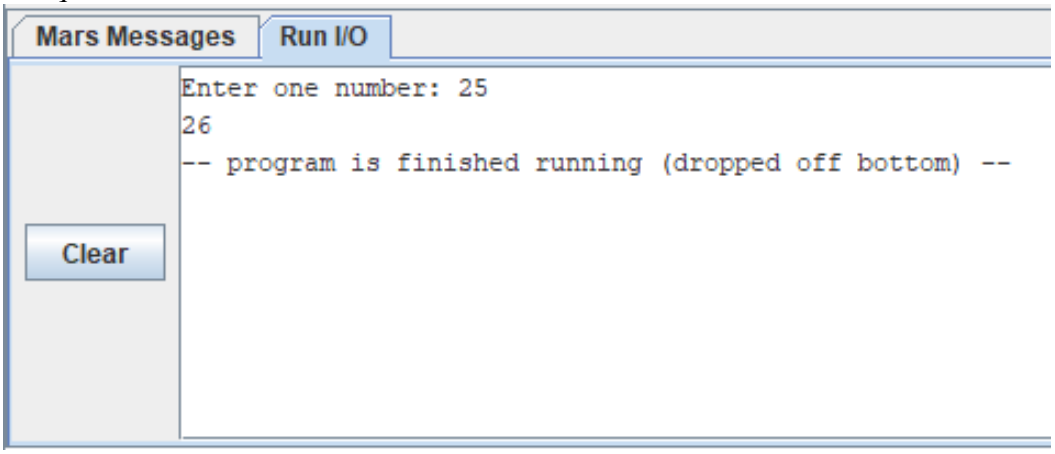
```
        move    $a0, $s0
        move    $a1, $s1
        move    $a2, $s2
        move    $a3, $s3
        jal     proc_example

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

proc_example:
        addi    $sp, $sp, -4
        sw      $s0, 0($sp)

        add     $t0, $a0, $a1
        add     $t1, $a2, $a3
        sub     $s0, $t0, $t1

        move    $v0, $s0

        lw      $s0, 0($sp)
        addi    $sp, $sp, 4

        jr      $ra
```

- Thanh ghi pc và $fp không thay đổi trực tiếp trong đoạn mã này.
- $ra được sử dụng để lưu địa chỉ trả về sau lệnh 'jal', giúp quay lại vị trí sau lệnh gọi hàm.
- $sp thay đổi khi cấp phát và giải phóng không gian trên ngăn xếp.

- **Chạy toàn chương trình một lần để xem kết quả**
- **Với code trong hình 1, nếu bỏ dòng "j exit", việc gì sẽ xảy ra?**
  - Nếu bỏ dòng "j exit" khỏi đoạn mã, thì chương trình sẽ không có lệnh nào để chấm dứt và thoát ra khỏi chương trình sau khi hàm getInt kết thúc. Khi chương trình thực hiện xong hàm getInt, nó sẽ tiếp tục thực hiện các lệnh sau đó mà không có sự kiểm soát nào về việc thoát khỏi chương trình.
- **Viết lại code trong Hình 1, thêm vào thủ tục tên showInt để in ra cửa sổ I/O giá trị của số int nhập vào cộng thêm 1.**

```
1          .data
2  prompt: .asciiz "Enter one number: "
3
4          .text
5  main:   jal getInt
6          move $s0, $v0
7          jal showInt    # Gọi thủ tục showInt với giá trị đã nhập + 1
8          j exit
9
10 getInt: li $v0, 4
11         la $a0, prompt
12         syscall
13
14         li $v0, 5
15         syscall
16         jr $ra
17
18 showInt: addi $a0, $s0, 1  # Tăng giá trị của số đã nhập lên 1
19          li $v0, 1         # Sử dụng syscall để in giá trị ra màn hình
20          syscall
21          jr $ra            # Trở về nơi gọi thủ tục
22
23 exit:
```

❖ *Kết quả:*



- **Viết lại code trong Hình 2, lúc này chương trình chính cần tính giá trị của cả hai biểu thức: (a + b) – (c + d) và (a – b) + (c – d), hàm proc_example có hai giá trị trả về và trong thân hàm sử dụng hai biến cục bộ $s0 và $s1 ($s1 lưu kết quả của (a – b) + (c – d))**

```
1         move    $a0, $s0
2         move    $a1, $s1
3         move    $a2, $s2
4         move    $a3, $s3
5         jal     proc_example
6
7   # Lấy giá trị của (a + b) - (c + d) từ $v0
8         move    $a0, $v0
9         li      $v0, 1
10        syscall
11
12  # Lấy giá trị của (a - b) + (c - d) từ $v1
13        move    $a0, $v1
14        li      $v0, 1
15        syscall
16
17  # Kết thúc chương trình
18        li $v0, 10
19        syscall
20
21  proc_example:
22        addi    $sp, $sp, -4
23        sw      $s0, 0($sp)
24
25        add     $t0, $a0, $a1
26        add     $t1, $a2, $a3
27        sub     $s0, $t0, $t1
28
29        move    $v0, $s0
30
31  # Tính giá trị của (a - b) + (c - d)
32        add     $t2, $a0, $a1
33        add     $t3, $a2, $a3
34        sub     $s1, $t2, $t3
35
36  # Move giá trị của (a + b) - (c + d) vào $v0
37        move    $v0, $s0
38
39  # Return giá trị của (a - b) + (c - d) thông qua $v1
40        move    $v1, $s1
41
42        lw      $s0, 0($sp)
43        addi    $sp, $sp, 4
44
45        jr      $ra
```

- **Viết lại code trong Hình 2, lúc này chương trình chính cần tính giá trị của cả hai biểu thức: (a + b) – (c + d), (e - f) hàm proc_example có 6 input và hai giá trị trả về và trong thân hàm sử dụng hai biến cục bộ $s0 và $s1 ($s1 lưu kết quả của e - f)**

```
        move    $a0, $s0
        move    $a1, $s1
        move    $a2, $s2
        move    $a3, $s3
        move    $a4, $s4
        move    $a5, $s5
        jal     proc_example

# Lấy giá trị của (a + b) - (c + d) từ $v0
        move    $a0, $v0
        li      $v0, 1
        syscall

# Lấy giá trị của (e - f) từ $v1
        move    $a0, $v1
        li      $v0, 1
        syscall

# Kết thúc chương trình
        li $v0, 10
        syscall

proc_example:
        addi    $sp, $sp, -4
        sw      $s0, 0($sp)

        add     $t0, $a0, $a1
        add     $t1, $a2, $a3
        sub     $s0, $t0, $t1

        move    $v0, $s0

        lw      $s0, 0($sp)
        addi    $sp, $sp, 4

# Tính giá trị của (a + b) - (c + d)
        add     $t0, $a0, $a1
        add     $t1, $a2, $a3
        sub     $s0, $t0, $t1

# Tính giá trị của (e - f)
        sub     $s1, $a4, $a5

# Move giá trị của (a + b) - (c + d) vào $v0
        move    $v0, $s0

# Return giá trị của (e - f) thông qua $v1
        move    $v1, $s1

        jr      $ra
```

Câu 2: Thực hành với đệ quy

1. **Tiếp tục nội dung hoàn thành đoạn code MIPS tính n!.**

```
        .data
Output1: .asciiz "Nhap n: "
Output2: .asciiz "n! = "
        .text
main:
        la $a0, Output1         #Xuất thông tin của Output1
        addi $v0, $0, 4
        syscall

        addi $v0, $0, 5         #nhập n
        syscall

        addi $s0, $v0, 0        #lưu giá trị n vào $s0

        addi $s1, $0, 1         #khởi tạo $s1 = 1 (biến giai thừa)

        addi $t1, $0, 1         #khởi tạo $t1 = 1 (biến đếm)
loop:
        mult $s1, $t1   #s1 * t1
        mflo $s1                #lưu giá trị vào $s1
        addi $t1, $t1, 1        #tăng biến đếm t1
        ble $t1, $s0, loop      #t1 <= n, thực hiện loop

        la $a0, Output2         #In ra thông báo kết quả
        addi $v0, $0, 4
        syscall

        addi $a0, $s1, 0
        addi $v0, $0, 1
        syscall
```

❖ *Kết quả:*

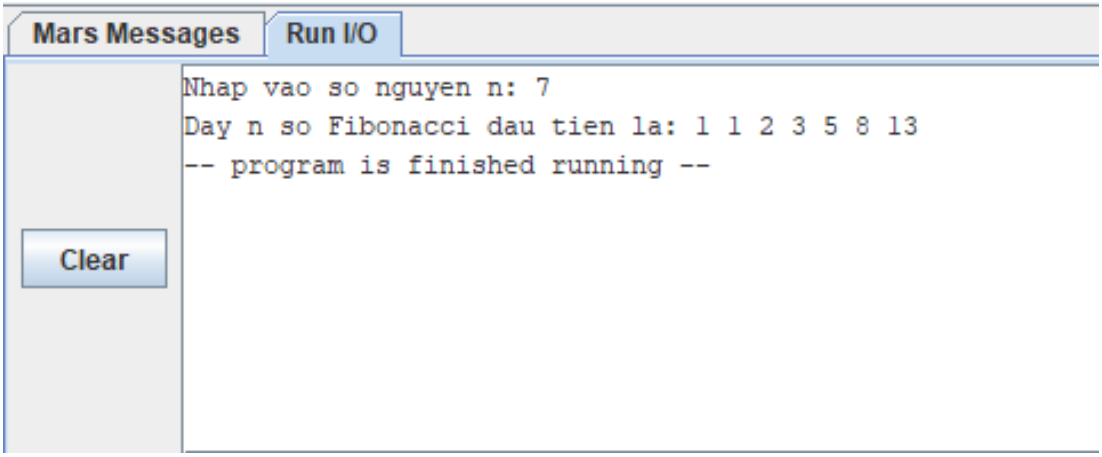
Mars Messages | Run I/O

```
Nhap n: 5
n! = 120
-- program is finished running (dropped off bottom) --


Reset: reset completed.

Nhap n: 10
n! = 3628800
-- program is finished running (dropped off bottom) --
```

2. Viết chương trình nhập vào n và xuất ra chuỗi Fibonacci tương ứng.

```
        .data
Output1: .asciiz "Nhap vao so nguyen n: "
Output2: .asciiz "Day n so Fibonacci dau tien la: "
Space: .asciiz " "
f:.word 0               #khai báo mảng f
        .text
main:
        li $v0, 4       #Xuất thông tin của Output1
        la $a0, Output1
        syscall

        li $v0, 5       #nhập số nguyên n
        syscall
        move $t0, $v0

        li $v0, 4       #Xuất thông tin của Output2
        la $a0, Output2
        syscall

        li $t4, 1

        sw $t4, f+0     #f[0] = 1
        sw $t4, f+4     #f[1] = 1

        li $t1, 1

        la $s0, f       #Lưu địa chỉ d vào $s0

loop:
        bgt $t1, $t0, end       #So sánh f[i] != 0 thực hiện continue
        lw $t5, ($s0)
        bne $t5, 0, continue
        lw $t6, -4($s0)         #giá trị f[i - 1]
        lw $t7, -8($s0)         #giá trị f[i - 2]
        add $t5, $t6, $t7       #$t5 = f[i - 1] + f[i - 2]
        sw $t5, ($s0)           #lưu $t5 vào f[i]
continue:
        li $v0, 1       #xuất f[i]
        lw $a0, ($s0)
        syscall

        li $v0, 4
        la $a0, Space
        syscall

        addi $t1, $t1, 1        #tăng đếm
        addi $s0, $s0, 4        #tăng địa chỉ f

        j loop          #nhảy về vòng lặp
end:
         li $v0, 10
         syscall
```

❖ *Kết quả:*



3. Vẽ lại hình ảnh của các stack trong trường hợp tính 5! Và 10!

Stack của 5!	
$ra	**$a0**
4194340	6
4194416	5
4194416	4
4194416	3
4194416	2
4194416	1

Stack của 10!	
$ra	**$a0**
4194340	11
4194416	10
4194416	9
4194416	8
4194416	7
4194416	6
4194416	5
4194416	4
4194416	3
4194416	2
4194416	1