Bài thực hành số 7

Lớp: 139365 – Học phần: Thực hành Kiến Trúc Máy Tính

Họ và tên: Đinh Thị Hồng Phúc **MSSV: 20215118**

Bài 1.

```
1  #Laboratory Exercise 7 Home Assignment 1
2  .data
3          mess: .asciiz "Abs = "
4  .text
5  main:
6          li $a1, -45      #load input parameter
7          jal abs          #jump and link to abs procedure
8          nop
9          add $s0, $zero, $v0
10         li $v0, 4
11         la $a0, mess
12         syscall
13         add $a0, $s0, $zero
14         li $v0, 1
15         syscall
16         li $v0, 10       #terminate
17         syscall
18 endmain:
19 #---------------------------------------------------------------
20 # function abs
21 # param[in] $a1 the interger need to be gained the absolute value
22 # return $v0 absolute value
23 #---------------------------------------------------------------
24 abs:
25         sub $v0, $zero, $a1      #put -(a0) in v0; in case (a0)<0
26         bltz $a1, done           #if (a0)<0 then done
27         nop
28         add $v0, $a1, $zero      #else put (a0) in v0
29 done:
30         jr $ra
```

*Thực hiện gõ chương trình vào công cu **MARS***

Giải thích:

- Trong hàm *main*, khai báo giá trị cần tính trị tuyệt đối vào thanh ghi $a1. Lệnh *jal* để nhảy tới phần *abs* sau đó quay lại đây và sẽ lưu địa chỉ trả về của chương trình ở thanh ghi $ra. Dòng lệnh từ 10-17 là để in giá trị tuyệt đối ra

màn hình và kết thúc chương trình. Giá trị tuyệt đối được lưu ở thanh ghi $s0 (dòng 9)

- Trong hàm *abs*, kiểm tra số có âm hay không bằng cách lấy 0 trừ đi số ban đầu, sau đó so sánh giá trị có được với 0. Nếu số ban đầu là số âm thì giá trị tuyệt đối của nó được lưu vào thanh ghi $v0 (dòng 28). Nếu là số dương thì chương trình in ra số ban đầu
- Dòng 30: Trở về địa chỉ được lưu trong thanh ghi $ra để tiếp tục chương trình

```
Abs = 45
-- program is finished running --
```

*Thực hiện chạy chương trình với **MARS***

Bài 2.

```
1  #Laboratory Exercise 7, Home Assignment 2
2  .data
3          mess: .asciiz "Max = "
4  .text
5  main:
6          li $v0, 4
7          la $a0, mess
8          syscall
9
10         li $a0, 11      #load test input
11         li $a1, 16
12         li $a2, 2003
13         jal max         #call max procedure
14         nop
15 endmain:
16         add $a0, $v0, $zero
17         li $v0, 1
18         syscall
19
20         li $v0, 10
21         syscall
22 #---------------------------------------------------------------
23 #Procedure max: find the largest of three integers
24 #param[in] $a0 integers
25 #param[in] $a1 integers
26 #param[in] $a2 integers
27 #return $v0 the largest value
28 #---------------------------------------------------------------
```

.data Subsequent items stored in Data segment a

```
29  max:
30          add $v0, $a0, $zero     #copy (a0) in v0; largest so far
31          sub $t0, $a1, $v0       #compute (a1)-(v0)
32          bltz $t0, okay          #if (a1)-(v0)<0 then no change
33          nop
34          add $v0, $a1, $zero     #else (a1) is largest thus far
35  okay:
36          sub $t0, $a2, $v0       #compute (a2)-(v0)
37          bltz $t0, done          #if (a2)-(v0)<0 then no change
38          nop
39          add $v0, $a2, $zero     #else (a2) is largest overall
40  done:
41          jr $ra                  #return to calling program
```

*Thực hiện gõ chương trình vào công cụ **MARS***

Giải thích:

- Trong hàm *main,* in chuỗi "Max = " ra màn hình, khai báo 3 giá trị so sánh với nhau
- Hàm *end_main*, in giá trị lớn nhất ra màn hình và kết thúc chương trình
- Hàm *max*, ban đầu giả sử giá trị lớn nhất là a0 và gán vào thanh ghi $v0. So sánh các số còn lại với v0 bằng các lấy số đó trừ đi v0, nếu được số nhỏ hơn 0 thì vẫn giữ nguyên giá trị v0, nếu lớn hơn 0 thì gán giá trị mới cho v0
- Giá trị lớn nhất được trả về bởi hàm *max* được lưu trong thanh ghi $v0

```
Max = 2003
-- program is finished running --
```

*Thực hiện chạy chương trình với **MARS***

Bài 3.

```
1  #Laboratory Exercise 7, Home Assignment 3
2  .text
3  push:
4          li $s0, 29
5          li $s1, 119
6          addi $sp, $sp, -8       #adjust the stack pointer
7          sw $s0, 4($sp)          #push s0 to stack
8          sw $s1, 0($sp)          #push s1 to stack
9  work:
10         nop
11 pop:
12         lw $s0, 0($sp)          #pop from stack to s0
13         lw $s1, 4($sp)          #pop from stack to s1
14         addi $sp, $sp, 8        #adjust the stack pointer
```

*Thực hiện gõ chương trình vào công cụ **MARS***

Giải thích:

- Hàm *push*, gán giá trị $s0, $s1 lần lượt là 29, 119. Sau đó điều chỉnh con trỏ stack bằng cách giảm giá trị của $sp đi 8 byte (push 2 giá trị vào stack). Lưu giá trị của $s0, $s1 vào stack bằng lệnh *sw*, địa chỉ được $s0, $s1 lưu trữ tại 4($sp) và 0($sp)
- Hàm *pop*, lấy giá trị từ stack bằng lệnh *lw*, lưu lần lượt các giá trị được lấy ra vào thanh ghi $s0, $s1. Sau đó điều chỉnh con trỏ stack về vị trí ban đầu bằng cách tăng giá trị $sp lên 8 byte (pop 2 giá trị)

$t7	15	0
$s0	16	119
$s1	17	29
$s2	18	0

*Thực hiện chạy chương trình với **MARS***

Bài 4.

```
#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, 5               #load test input N
        jal FACT                #call fact procedure
        nop
```

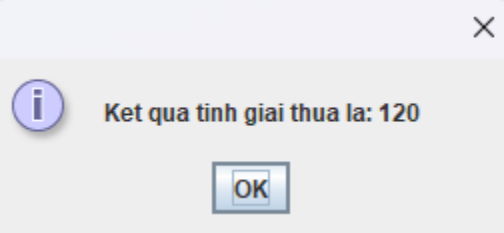
```
        lw $ra, 0($sp)          #restore return address (5)
        addi $sp, $fp, 0        #return stack pointer (6)
        lw $fp, -4($sp)         #return frame pointer (7)
        jr $ra
wrap_end:
#---------------------------------------------------------------------
#Procedure FACT: compute 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
```

```
52  recursive:
53          addi $a0, $a0, -1       #adjust input argument
54          jal FACT                #recursive call
55          nop
56          lw $v1, 0($sp)          #load a0
57          mult $v1, $v0           #compute the result
58          mflo $v0
59  done:
60          lw $ra, 4($sp)          #restore return address
61          lw $a0, 0($sp)          #restore a0
62          addi $sp, $fp, 0        #restore stack pointer
63          lw $fp, -4($sp)         #restore frame pointer
64          jr $ra                  #jump to calling
65  fact_end:
```

*Thực hiện gõ chương trình vào công cu **MARS***

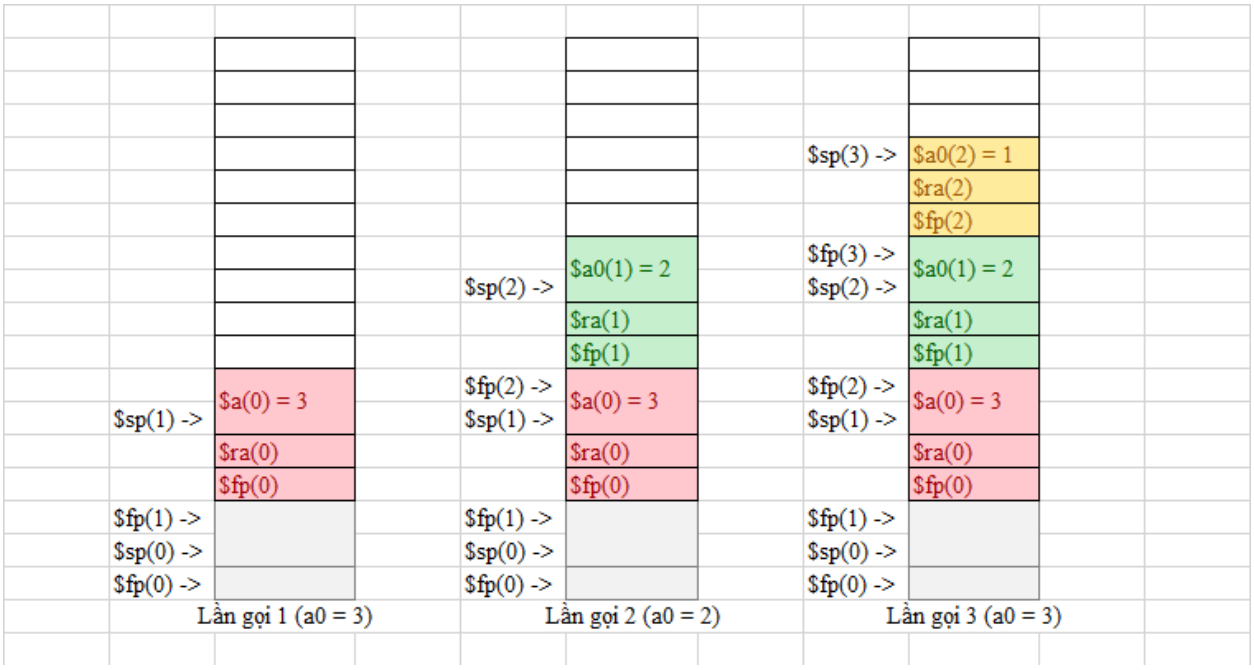
Giải thích:

- Dòng 7-14: In kết quả ra dialog và kết thúc chương trình
- Hàm *WARP*: Thực hiện các các bước để tính N! bằng cách gọi hàm *FACT*
- Hàm *FACT*: Tính giai thừa bằng cách sử dụng đệ qui. Nhận số nguyên N đầu vào $a0 và trả về giá trị giai thừa $v0. Kiểm tra điều kiện dừng khi N<2 (dòng 46) thì hàm trả về giá trị 1. Nếu N >=2 thì tính giai thừa của N-1 bằng cách gọi đến chính hàm *FACT* và nhân với N để được kết quả cuối cùng
- Các dòng lệnh 20-23, 29-32, 40-44, hàm *done* dùng để lưu trữ và lấy lại địa chỉ con trỏ của frame ban đầu, tạo con trỏ mới cho frame mới



*Thực hiện chạy chương trình với **MARS***

Vẽ stack:


$sp(1) ->
$a(0) = 3
$ra(0)
$fp(0)
$fp(1) ->
$sp(0) ->
$fp(0) ->
Lần gọi 1 (a0 = 3)
$sp(2) ->
$a0(1) = 2
$ra(1)
$fp(1)
$fp(2) ->
$a(0) = 3
$sp(1) ->
$ra(0)
$fp(0)
$fp(1) ->
$sp(0) ->
$fp(0) ->
Lần gọi 2 (a0 = 2)
$sp(3) ->
$a0(2) = 1
$ra(2)
$fp(2)
$fp(3) ->
$a0(1) = 2
$sp(2) ->
$ra(1)
$fp(1)
$fp(2) ->
$a(0) = 3
$sp(1) ->
$ra(0)
$fp(0)
$fp(1) ->
$sp(0) ->
$fp(0) ->
Lần gọi 3 (a0 = 3)

Bài 5.

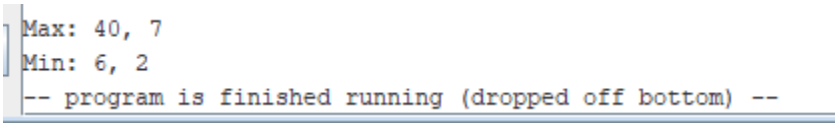
```
#Laboratory Exercise 7 Assignment 5
.data
        A: .asciiz "Max: "
        B: .asciiz "\nMin: "
        C: .asciiz ", "
.text
push:
        li $s0, 19
        li $s1, 13
        li $s2, 6
        li $s3, 8
        li $s4, 9
        li $s5, 29
        li $s6, 12
        li $s7, 40
        addi $t0, $s0,0         #max
        addi $t1, $s0,0         #min
        li $t2, 0               #i
        li $t3, 8               #n
        li $a1, 0               #index of max = 0
        li $a3, 0               #index of min = 0
        addi $sp, $sp, -32
        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)
main:
        jal work
        j print
work:
        slt $t5, $t2, $t3
        beq $t5, $zero, done
        mul $t6, $t2, 4
        add $t7, $t6, $sp
        lw $t8, 0($t7)
        slt $t5, $t8, $t0
        beq $t5, $zero, swap_max
continue1:
        slt $t5, $t8, $t1
        bne $t5, $zero, swap_min
```

```
continue:
        addi $t2, $t2, 1
        j work
swap_max:
        add $t0, $t8, $zero
        addi $a1, $t2, 0
        j continue1
swap_min:
        add $t1, $t8, $zero
        addi $a3, $t2, 0
        j continue
done:
        jr $ra
print:
        li $v0, 4
        la $a0, A
        syscall

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

        li $v0, 4
        la $a0, C
        syscall

        li $v0, 1
        move $a0, $a1
        syscall

        li $v0, 4
        la $a0, B
        syscall

        li $v0, 1
        move $a0, $t1
        syscall

        li $v0, 4
        la $a0, C
        syscall

        li $v0, 1
        move $a0, $a3
        syscall
```

*Thực hiện gõ chương trình vào công cụ **MARS***

Giải thích:

- Hàm *push:* Gán giá trị dãy số vào các thanh ghi từ $s0 đến $s8, sau đó đẩy các số vào stack. Khởi tạo các biến max, min, biến đếm, độ dài dãy số và vị trí của max, min
- Hàm *main*: Sử dụng hàm *work* để tìm giá trị lớn nhất và nhỏ nhất của dãy số, sau đó in ra màn hình các giá trị này bằng hàm *print*
- Hàm *work*: Tìm giá trị lớn nhất, nhỏ nhất bằng cách so sánh giá trị của $t2 với giá trị lớn nhất hiện tại, nếu lớn hơn thì sẽ cập nhật giá trị lớn nhất, chỉ số sẽ được lưu vào thanh ghi $a1. Nếu giá trị của $t2 nhỏ hơn giá trị nhỏ nhất hiện tại thì sẽ cập nhật giá trị nhỏ nhất, chỉ số sẽ được lưu vào thanh ghi $a3.



*Thực hiện chạy chương trình với **MARS***