Bài thực hành số 7: Lời gọi thủ tục, ngăn xếp và các tham số

Nguyễn Văn Hiếu – 20225717

Assginment 1:

```
.text
main:   li      $a0, -50        #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
```

Ban đầu khởi tạo giá trị lưu vào thanh ghi $a0 có giá trị là -50.

Trước khi chạy lệnh jar abs:

$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	-50
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	0
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0

$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0

$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	4194312
pc		4194328
hi		0
lo		0

Sau khi chạy lệnh jar abs:

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	10
$v1	3	0
$a0	4	-50
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	0
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	50
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	4194312
pc		4194328
hi		0
lo		0

Nhận xét:

Khi chạy lệnh jar abs thì thanh ghi $ra được gán bằng địa chỉ của câu lệnh tiếp theo và thanh ghi pc được gán bằng địa chỉ tại nhãn dán abs.

Kết quả cuối cùng ta thấy được khi nạp kết quả vào thanh ghi $a0 là -50 thì kết quả lấy giá trị tuyệt đối được lưu ở thanh ghi $s0 là 50.

⇨ Kết quả thu được đúng lí thuyết.

Assginment 2:

mips1.asm*

```
#Laboratory Exercise 7, Home Assignment 2
.text
main:
li $a0, -3 #load test input
 li $a1, 9
 li $a2, 20
 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
```

Đoạn chương trình trên thực hiện công việc tìm số lớn nhất trong 3 số được lưu 3 thanh ghi $a0, $a1 và $a2. Kết quả thu được sẽ được lưu ở thanh ghi $s0.

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).

Kết quả thu được:

$a0	4	-3
$a1	5	9
$a2	6	20

$s0	16	20

⇨ Kết quả thu được đúng với lí thuyết.

Assginment 3:

```
#Laboratory Exercise 7, Home Assignment 3
.text
      li $s0, 10
      li $s1, -5
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
```

Trước khi chạy lệnh push: addi $sp, $sp, -8 thì giá trị trên thanh ghi $s0 là 0x7fffeffc. Sau khi chạy lệnh trên thì giá trị trên thanh ghi $sp là 0x7fffeff4.

Thanh ghi $sp bị giảm đi 8 bytes tức là có sự cấp phát bộ nhớ cho stack 8 bytes.

Sau đó lần lượt ghi giá trị trong thanh ghi trong $s0, $sp + 4, giá trị trong $s1 vào $sp + 0.

Sau khi chạy lệnh addi ở nhãn dán 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:

```
#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)
```

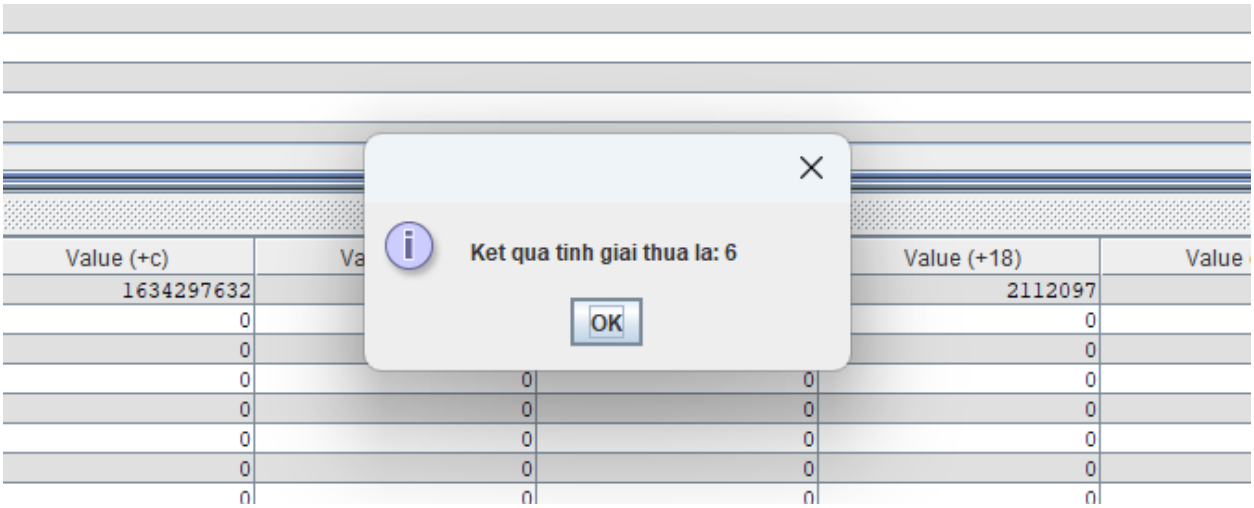
```
#save return address (4)
li    $a0,3
#load test input N
jal  FACT
#call fact procedure
nop
lw  $ra,0($sp)
#restore return address (5)
addi  $sp,$fp,0
#return stack pointer  (6)
 lw    $fp,-4($sp)   #return frame pointer  (7)
      jr    $ra
wrap_end:

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

Đoạn code trên thực hiện công việc nhập vào giá trị n bất kì rồi tính n! sau đó in ra màn hình. Đoạn code trên giá trị n nhập vào đang được lưu ở thanh ghi $a0 với giá trị là 3. Kết quả thu được sẽ là 3! = 6.



Bảng thể hiện giá trị của ngăn xếp: (với n = 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

Assginment 5:

```
.data
  message1: .asciiz "Largest: "
  message2: .asciiz "\nSmallest: "
  message3: .asciiz ","
.text
 main:
```

```
  li $s0, 5
  li $s1, -3
  li $s2, 9
  li $s3, 11
  li $s4, 20
  li $s5, -5
  li $s6, 23
  li $s7, -20
jal save
nop
li $v0,4
la $a0, message1
syscall
add $a0, $t0, $zero   #in max
li $v0,1
syscall
li $v0, 4
la $a0, message3
syscall
add $a0, $t5, $zero     #in dia chia thanh ghi chua max
li $v0,1
syscall
li $v0, 4
```

```
 la $a0, message2
 syscall
 add $a0, $t1, $zero     #in ra min
 li $v0,1
 syscall
 li $v0,4
 la $a0, message3
 syscall
 add $a0, $t6, $zero     #in ra dia chi cua min
 li $v0, 1
 syscall
endmain: li $v0,10          #Ket thuc
      syscall
max:
  add $t0,$t3,$zero
  add $t5,$t2,$zero
  jr $ra
min:
  add $t1,$t3,$zero
  add $t6,$t2,$zero
  jr $ra
save:
  add $t9,$sp,$zero
```

```
  addi $sp,$sp,-32
  sw $1,0($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)
  add $t0, $s0, $zero       #gia tri max luu thanh ghi s0
  add $t1, $s0, $zero       #gia tri min luu thanh ghi s0
  li $t5,0
  li $t6,0
  li $t2,0
tim:
 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, max  # If $t3 > Max branch to the swapMax
```

nop

sub $t4,$t3,$t1

bltzal $t4, min # If $t3 < Min branch to the swapMin

nop

j tim # Repeat

done:

lw $ra, -4($sp)

jr $ra   # Return to calling program

Đoạn code trên thực hiện công việc tìm giá trị lớn nhất và giá trị nhỏ nhất khi nhập vào giá trị cho các thanh ghi và in ra địa chỉ của các thanh ghi đó.

Ban đầu nhập các giá trị của các thanh ghi s0,s1,s2,s3,s4,s5,s6,s7 lần lượt là 5, -3, 9, 11, 20, -5, 23, -20. Kết quả thu được khi đó là:

```
Reset: reset completed.

Largest: 23,6
Smallest: -20,7
-- program is finished running --
```

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	10
$v1	3	0
$a0	4	7
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	23
$t1	9	-20
$t2	10	7
$t3	11	4194340
$t4	12	0
$t5	13	6

$t6	14	7
$t7	15	0
$s0	16	5
$s1	17	-3
$s2	18	9
$s3	19	11
$s4	20	20
$s5	21	-5
$s6	22	23
$s7	23	-20
$t8	24	0
$t9	25	2147479548
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	4194340
pc		4194464
hi		0
lo		0

Kết quả thu được là max = 23 ở thanh ghi 6 và min là -20 ở thanh ghi 7.

⇨ Kết quả thu được đúng với lí thuyết.