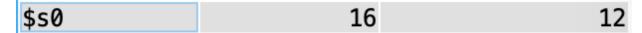
## Week7

### **Assignment 1**

Create a new project to implement the program in Home Assignment 1. Compile and upload to simulator. Change input parameters and observe the memory when run the program step by step. Pay attention to register pc, \$rato clarify invoking procedure process (Refer to figure 7).

```
#Laboratory Exercise 7 Home Assignment 1
.text
main:
    li $a0,-12
    jal abs
    nop
    add $s0, $zero, $v0
    li $v0,10
    syscall
endmain:
abs:
    sub $v0,$zero,$a0
    bltz $a0,done
    nop
    add $v0,$a0,$zero
done:
    jr $ra
    #Result at $s0
```

Kết quả: Đầu vào \$a0 = -12 => đầu ra lưu tại \$s0 = 12



## Assignment 2

Create a new project to implement the program in Home Assignment 2. Compile and upload to simulator. Change input parameters (register a0, a1, a2) and observe the memory when run the program step by step. Pay attention to register pc, \$rato clarify invoking procedure process (Refer to figure 7).

```
#Laboratory Exercise 7, Home Assignment 2
.text
main:
   li $a0,9
                     # load test input
   li $a1,-3
   li $a2,10
   jal max
                     # call max proceduce
   nop
   add $s0, $zero, $v0
   li $v0,10
   syscall
endmain:
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:
                         # return to calling program
   jr $ra
   # Largest number a $s0
```

Kết quả: Đầu vào là 3 số 9, -3, 10 => max = 10

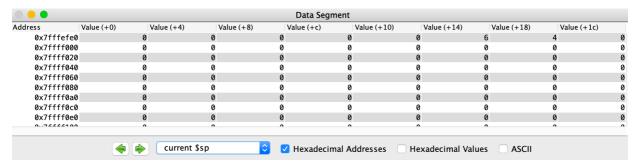
\$50 IO I	\$s0	16	10
-----------	------	----	----

# **Assignment 3**

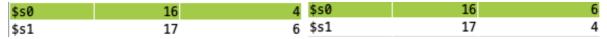
Create a new project to implement the program in Home Assignment 3. Compile and upload to simulator. Pass test value to registers \$s0 and \$s1, observe run process, pay attention to stack pointer. Goto memory space that pointed by sp register to view push and pop operations in detail.

```
#Laboratory Exercise 7, Home Assignment 3
.text
    li $s0, 4
    li $s1, 6
push:
                    # add just stack pointer
    addi $sp,$sp,-8
   sw $s0,4($sp)
                       # push $s0 to stack
                        # push $s1 to stack
    sw $s1,0($sp)
work:
    nop
pop:
                       # pop from stack to $s0
    lw $s0,0($sp)
    lw $s1,4($sp)
                        # pop from stack to $s1
    addi $sp,$sp,8
                        # adjust the stack pointer
```

#### Kết quả: Các giá trị lưu ở vùng nhớ stack



Giá trị thanh ghi \$s0, \$s1 được được hoán đổi lại:



# **Assignment 4**

Create a new project to implement the program in Home Assignment 4. Compile and upload to simulator. Pass test input through register a0, run this program and test result in register v0. Run this program in step by step mode, observe the changing of register pc, ra, sp and fp. Draw the stack through this recursive program in case of n=3 (compute 3!).

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

#### Kết quả:

0x3, 0x2, 0x1 lần lượt là các giá trị \$a0 sau mỗi vòng lặp được stack giữ lại Các giá trị khác 0 còn lại theo từng khoảng bộ nhớ là lưu trữ của \$ra và con trỏ \$fp (Ví dụ bộ ba: 0x3, 0x7fffeff4, 0x400080 tương ứng là \$a0, \$fp, \$ra ở lần gọi thứ nhất)

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x7fffefc0	0×00000000	0x00000000	0×00000000	0×000000000	0x00000001	0x00400080	0x7fffefe8	0×00000002
0x7fffefe0	0x00400080	0x7fffeff4	0×00000003	0x00400038	0x7fffeffc	0x00400004	0×00000000	0×00000000

#### Khi N = 3:

Stack
\$fp = 0x7fffefcd
\$a0 = 0x01
\$ra = 0x40080
\$fp = 0x7fffefe8
\$a0 = 0x0000002
\$ra = 0x0040080
\$fp = 0x7fffeff4
\$a0 = 0x0000003
\$ra = 0x0040038
\$fp = 0x7fffeffc
\$ra = 0x0040004

## **Assignment 5**

Write a procedure to find the largest, the smallest and these positions in a list of 8 elements that are stored in regsiters \$s0 through s7. For example: Largest: 9,3 -> The largest element is stored in \$s3, largest value is 9 Smallest: -3,6 -> The smallest element is stored in s6, smallest value is -3 Tips: using stack to pass arguments and return results.

```
.text
main:
                  # load data into $s0 -> $s7
   li $s0,-34
   li $s1,-33
   li $s2,6
   li $s3,-2
   li $s4,-9
   li $s5,100
   li $s6,4
   li $s7,-1
   li $t1,1
                  # init position into $t1, $t2, $t3
   li $t2,1
   li $t3,1
    jal init
   nop
   li $t4,9
   sub $a0,$t4,$t2 # position of max
    sub $a1,$t4,$t3 # position of min
    j end
   nop
endmain:
init:
   add v0, s7, zero # assign max = v0 = s7
   add v1,\$s7,\$zero # assign min = v1 = s7
push:
   addi $sp,$sp,-32
                       # allocate space for $s0->$s7 in stack
   sw $s0,28($sp)
   sw $s1,24($sp)
   sw $s2,20($sp)
   sw $s3,16($sp)
   sw $s4,12($sp)
   sw $s5,8($sp)
   sw $s6,4($sp)
   sw $s7,0($sp)
pop:
    addi $sp,$sp,4
                       # Nhay xuong 1 o nho trong stack
   lw $a1,0($sp)
                       # $a1 = current_value
   addi $t1,$t1,1
    sub $t0,$a1,$v0
   bltz $t0,okay1
                       # if $a0 < max --> jump okay1
   nop
    add v0, a1, e=0 # else --> max = v0 = a1
    add $t2,$t1,$zero # $t2 = position of max
okay1:
    sub $t0,$a1,$v1
   bgtz $t0,okay2  # if $a1 > min jump okay2
    nop
```

```
add $v1,$a1,$zero # else min = $v1 = $a1
add $t3,$t1,$zero # $t3 = position of min
okay2:
   bne $a1,$s0,pop # loop while $a1 != $s0
   nop
done:
   jr $ra # continue main
   # Largest: $v0,$a0
   # Smallest: $v1,$a1
end:
```

# Kết quả:

• Input: Giá trị \$s7->\$s0 và thứ tự lưu trữ của \$sp

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	
0x7fffefe0	)	4	100	-9	-2	6	-33	-34

• Output:

\$v0	2	100
\$v0 \$v1	3	-34
\$a0	4	6
\$a1	5	1