Báo cáo Thực hành Kiến trúc máy tính tuần 7

Họ và tên: Ôn Quang Tùng

MSSV:20226096

Assignment 1

```
#Laboratory Exercise 7 Home Assignment 1
.text
main:
 li $a0, -45 #load input parameter
 jal abs #jump and link to abs procedure
 nop
 add $s0, $zero, $v0
#exit
 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 #return address after jal abs
```

- Trước khi chạy lệnh “jal abs”:

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x2404ffd3	addiu $4,$0,0xffffffd3	3: main: li $a0, -45 #load input parameter
	0x00400004	0x0c100006	jal 0x00400018	4: jal abs #jump and link to abs procedure
	0x00400008	0x00000000	nop	5: nop
	0x0040000c	0x00028020	add $16,$0,$2	6: add $s0, $zero, $v0
	0x00400010	0x2402000a	addiu $2,$0,0x0000000a	7: li $v0, 10 #terminate
	0x00400014	0x0000000c	syscall	8: syscall
	0x00400018	0x00041022	sub $2,$0,$4	16: sub $v0,$zero,$a0 #put -(a0) in v0; in case (a0)<0
	0x0040001c	0x04800002	bltz $4,0x00000002	17: bltz $a0,done #if (a0)<0 then done
	0x00400020	0x00000000	nop	18: nop
	0x00400024	0x00801020	add $2,$4,$0	19: add $v0,$a0,$zero #else put (a0) in v0
	0x00400028	0x03e00008	jr $31	21: jr $ra

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0xfffffffd3
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400004
hi		0x00000000
lo		0x00000000

- Sau khi chạy lệnh “jal abs”:

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x2404ffd3	addiu $4,$0,0xffffffd3	3: main: li $a0, -45 #load input parameter
	0x00400004	0x0c100006	jal 0x00400018	4: jal abs #jump and link to abs procedure
	0x00400008	0x00000000	nop	5: nop
	0x0040000c	0x00028020	add $16,$0,$2	6: add $s0, $zero, $v0
	0x00400010	0x2402000a	addiu $2,$0,0x0000000a	7: li $v0, 10 #terminate
	0x00400014	0x0000000c	syscall	8: syscall
	0x00400018	0x00041022	sub $2,$0,$4	16: sub $v0,$zero,$a0 #put -(a0) in v0; in case (a0)<0
	0x0040001c	0x04800002	bltz $4,0x00000002	17: bltz $a0,done #if (a0)<0 then done
	0x00400020	0x00000000	nop	18: nop
	0x00400024	0x00801020	add $2,$4,$0	19: add $v0,$a0,$zero #else put (a0) in v0
	0x00400028	0x03e00008	jr $31	21: jr $ra

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0xffffffd3
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400008
pc		0x00400018
hi		0x00000000
lo		0x00000000

➔ Khi chạy lệnh "jal abs" (địa chỉ lệnh 0x00400004), lệnh tiếp theo của lệnh đó được lưu vào thanh ghi $ra (0x00400008). Đồng thời, thanh ghi pc được gán giá trị 0x00400018 (địa của của nhãn abs)

Assignment 2

```
#Laboratory Exercise 7, Home Assignment 2
.text
main: li $a0,2 #load test input
 li $a1,6
 li $a2,9
 jal max #call max procedure
 nop
 move $s0, $v0
 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
```

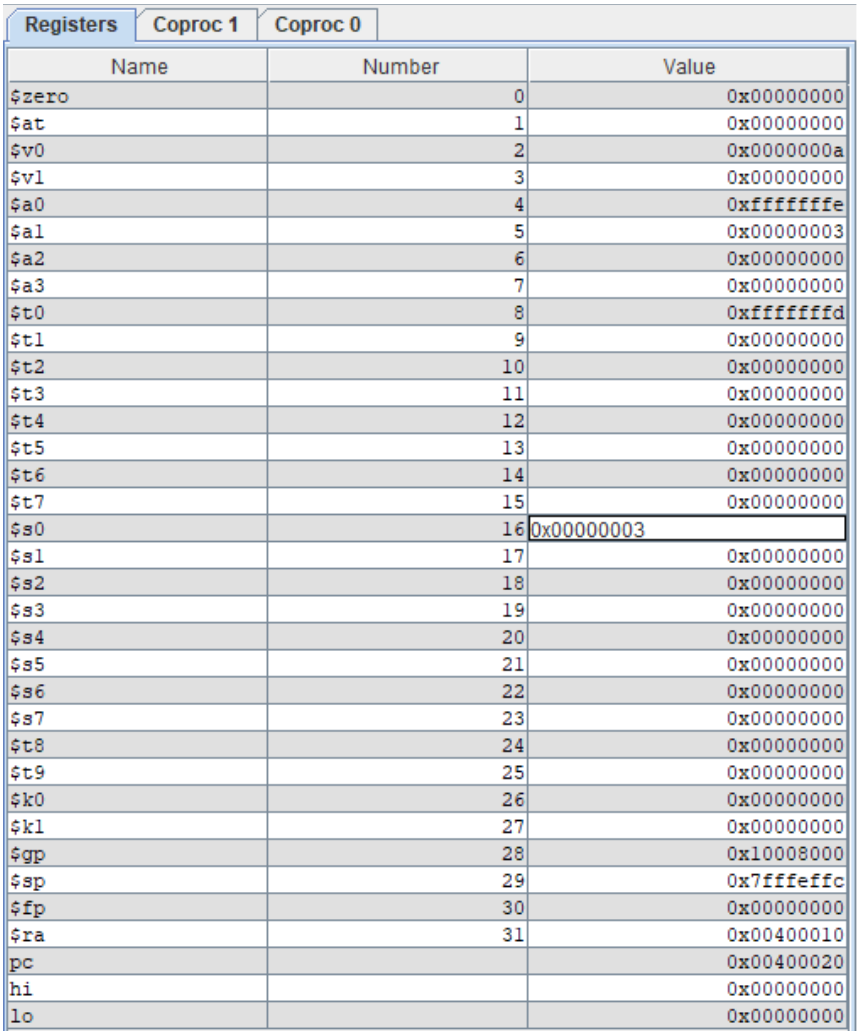
- Chạy test mẫu của đề bài:

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000002
$a1	5	0x00000006
$a2	6	0x00000009
$a3	7	0x00000000
$t0	8	0x00000003
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000009
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400010
pc		0x00400020
hi		0x00000000
lo		0x00000000

Thanh ghi $s0 = 9 là kết quả của chương trình

- Chạy test tự tạo: $a0 = -2, $a1 = 3, $a2 = 0



Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0xfffffffe
$a1	5	0x00000003
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0xfffffffd
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000003
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400010
pc		0x00400020
hi		0x00000000
lo		0x00000000

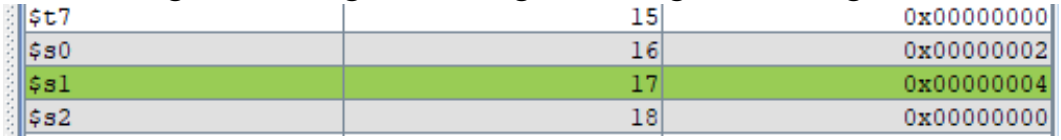
Thanh ghi $s0 = 3 là kết quả của chương trình

➔ Khi chạy lệnh “jal abs”, lệnh tiếp theo của lệnh đó được lưu vào thanh ghi $ra. Đồng thời, thanh ghi pc được gán giá trị địa của của nhãn max. Sau khi chạy đến nhãn done, lệnh “jr $ra” gán thanh ghi pc giá trị địa chỉ của thanh ghi $ra

Assignment 3

```
#Laboratory Exercise 7, Home Assignment 3
.text
 li $s0, 2
 li $s1, 4
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
```

- Gán thanh ghi $s0 với giá trị 2 và gán thanh ghi $s1 với giá trị 4



$t7	15	0x00000000
$s0	16	0x00000002
$s1	17	0x00000004
$s2	18	0x00000000

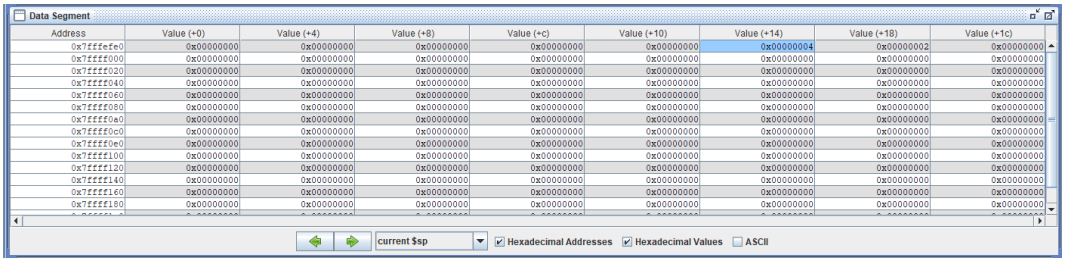
- Lệnh addi $sp,$sp,-8 ở nhãn "push" là để giảm giá trị địa chỉ của thanh ghi $sp đi 8 đơn vị. Tức là có sự cấp phát bộ nhớ 8 byte trong stack.
- Địa chỉ ban đầu của $sp

$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000

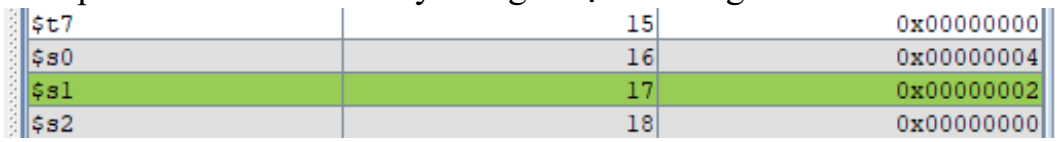
- Địa chỉ $sp sau khi trừ đi 8 đơn vị

$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeff4
$fp	30	0x00000000
$ra	31	0x00000000

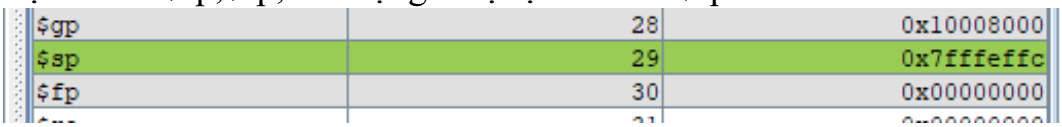
- Sau đó ta lưu giá trị của $s0 vào $sp + 4 và $s1 vào $sp + 0



- Trong nhãn "pop", 2 lệnh lw thực hiện lưu giá trị có địa chỉ $sp + 0 vào $s0 và $sp + 4 vào $s1. Điều này làm giá trị 2 thanh ghi đổi chỗ cho nhau



- Lệnh addi $sp,$sp,8 trả lại giá trị địa chỉ của $sp về ban đầu



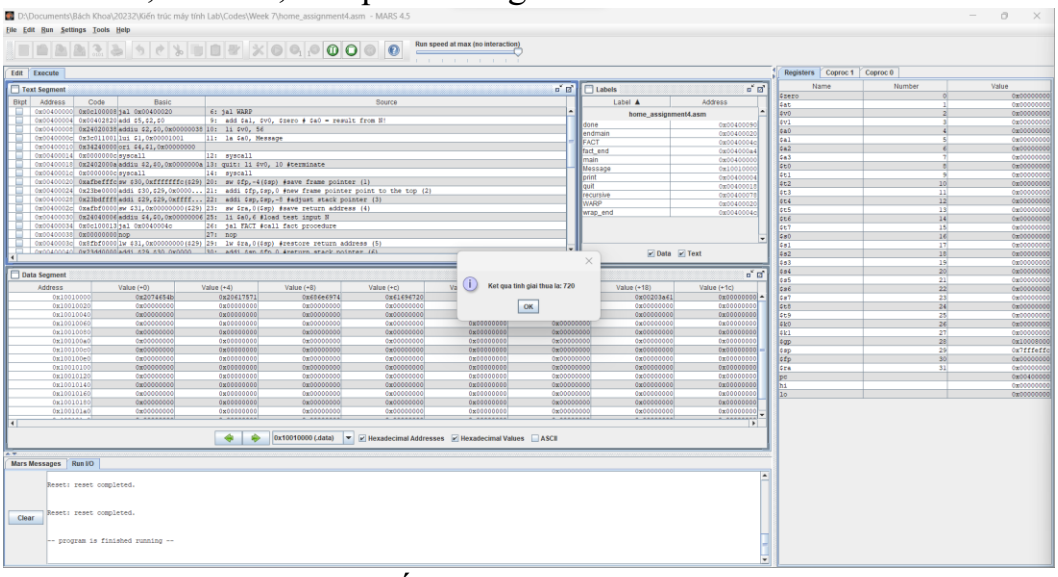
Assignment 4

```
#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 $a0,6 #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
```

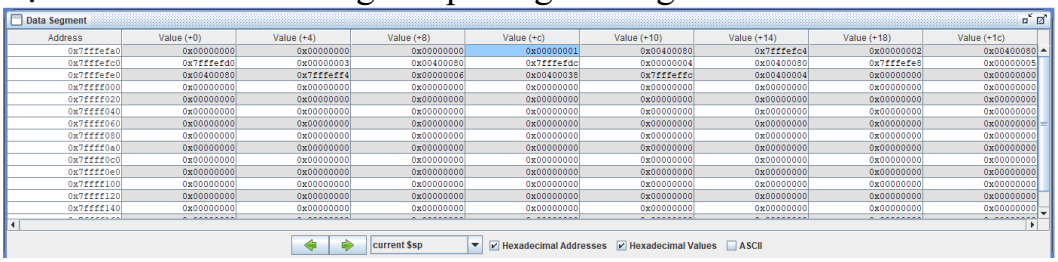
```
lw $a0,0($sp) #restore a0
addi $sp,$fp,0 #restore stack pointer
lw $fp,-4($sp) #restore frame pointer
jr $ra #jump to calling
fact_end:
```

- Với n = 6, $a0 = 6, kết quả chương trình là 720



➔ Chương trình chạy đúng kết quả mong đợi

- Sự thanh đổi của thanh ghi $sp trong chương trình



- Bảng biểu diễn stack giá trị các ngăn nhớ trong trường hợp n = 3

0x 7fffef8	$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

Assignment 5

```
# #Laboratory Exercise 7, Assignment 5

.data
 largest: .asciiz "Largest: "
 smallest: .asciiz "\nSmallest: "
 comma: .asciiz ","

.text
main:
# 3 5 8 6 1 7 2 4
 li $s0, 3
 li $s1, 5
 li $s2, 8
 li $s3, 6
 li $s4, 1
 li $s5, 7
 li $s6, 2
 li $s7, 4

 jal save
 nop
 print:
 # Print message largest
 li $v0, 4
```

```
la $a0, largest
syscall

# Print max - $t1 = max
li $v0, 1
move $a0, $t1
syscall

# Print message Comma
li $v0, 4
la $a0, comma
syscall

# Print max index - $t5 = max index
li $v0, 1
move $a0, $t5
syscall

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

# Print max - $t2 = min
li $v0, 1
move $a0, $t2
syscall

# Print message Comma
li $v0, 4
la $a0, comma
syscall

# Print min index - $t6 = min index
li $v0, 1
move $a0, $t6
syscall

end_main:
li $v0, 10
```

```
 syscall

save:
 move $t9, $sp # save the address of the top stack - $t9 = address of top stack
 addi $sp, $sp, -32 # get 32 bytes
 sw $ra, 0($sp) # address to return to main
 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)

 add $t1, $s0, $0 # $t1 = max = $s0
 add $t2, $s0, $0 # $t2 = min = $s0
 li $t5, 0 # $t5 = max index = 0
 li $t6, 0 # $t6 = max index = 0
 li $t0, 0 # $t0 = i = 0

# 3 5 8 6 1 7 2 4
find:
 addi $sp, $sp, 4
 addi $t0, $t0, 1
 beq $sp, $t9, end_find # if $sp = top stack, jump to end_find
 nop
 lw $t3, 0($sp) # load the current value
 sub $t8, $t1, $t3 # $t8 = max - current value
 bltzal $t8, swap_max # if max - current value <= 0, then jump to swap_max
 nop
 sub $t8, $t3, $t2 # $t8 = current value - min
 bltzal $t8, swap_min # if current value - min <= 0, then jump to swap_min
 nop
 j find #loop
 nop
end_find:
 lw $ra -32($sp)
 jr $ra

# 3 5 8 6 1 7 2 4
```

```
swap_max:
move $t1, $t3 # $t1 = current $sp
move $t5, $t0 # $t5 = current index
jr $ra # return to find

# 3 5 8 6 1 7 2 4
 swap_min:
move $t2, $t3 # $t1 = current $sp
move $t6, $t0 # $t6 = current index
jr $ra # return to find
```

- Test: $s0 = 3, $s1 = 5, $s2 = 8, $s3 = 6, $s4 = 1, $s5 = 7, $s6 = 2, $s7 = 4
- Kết quả của các thanh ghi và bộ nhớ stack sau khi thực hiện chương trình trên: