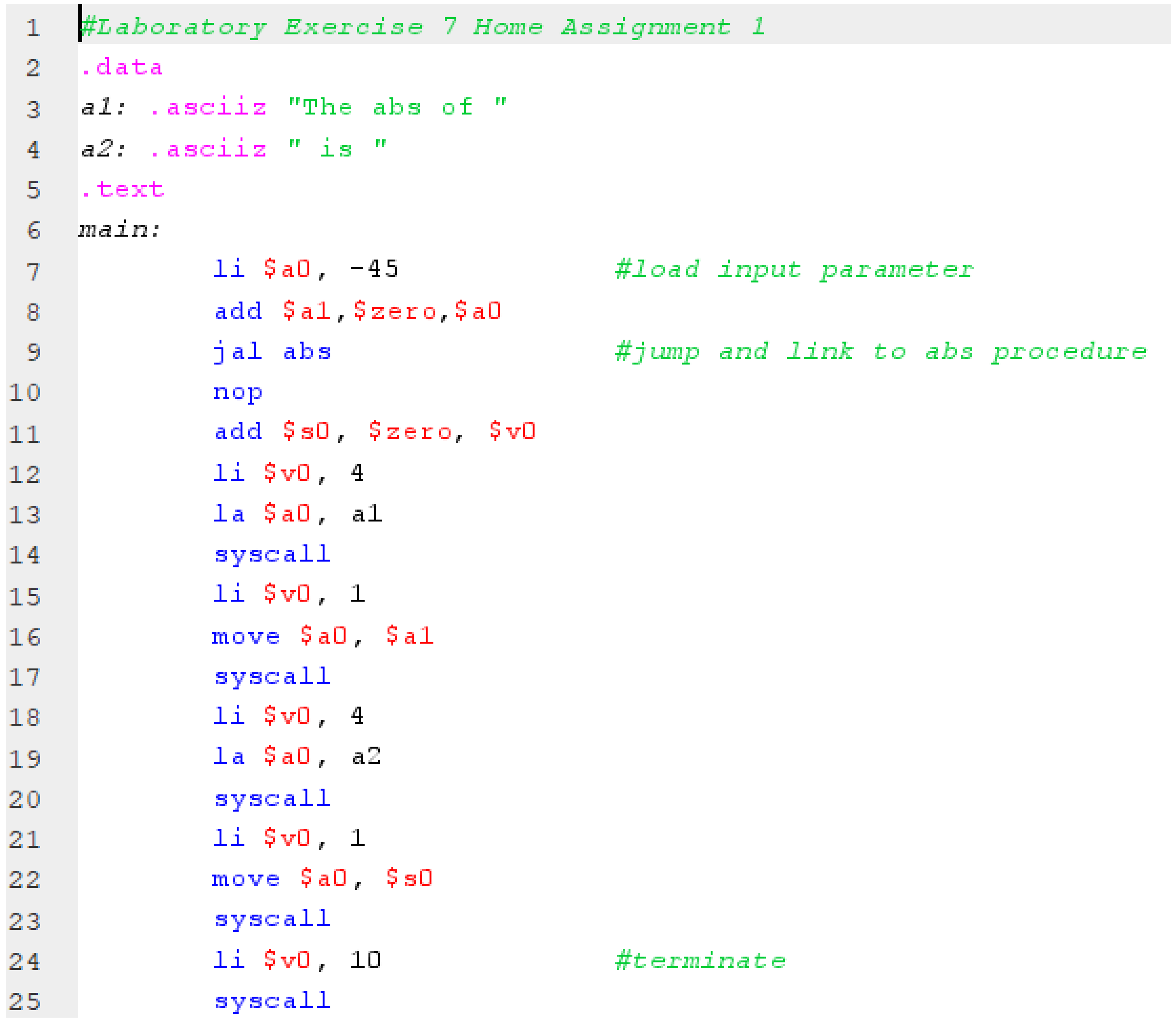
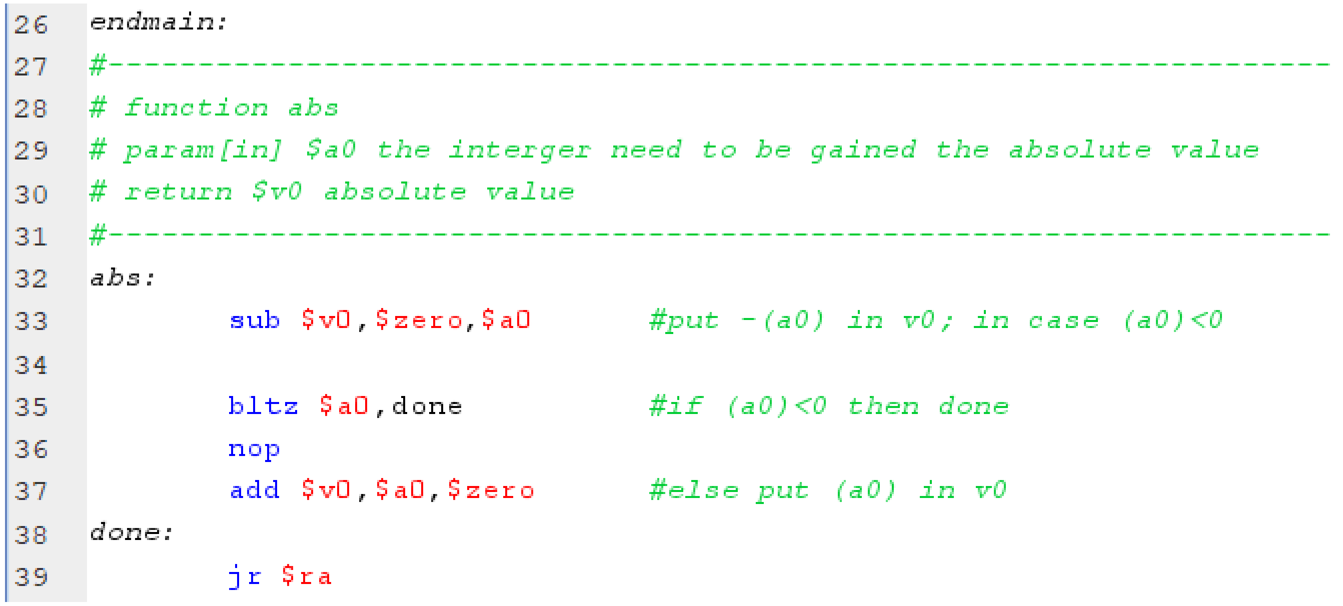
Laboratory Exercise 7

Procedure calls, stack and parameters

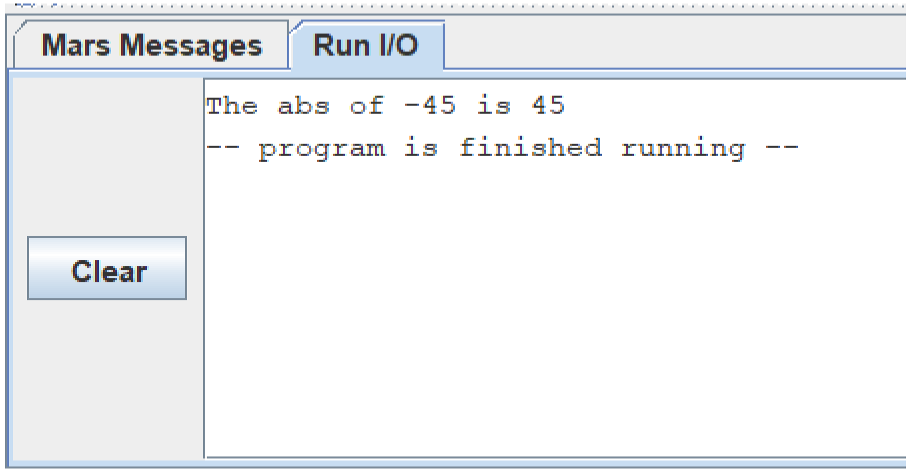
1. **Assignment 1:** *Create a new project to implement the program in Home Assignment 1. Compile and upload to simulator. Change input parameters and observe the memory when run the program step by step. Pay attention to register $pc, $ra to clarify invoking procedure process.*



```
#Laboratory Exercise 7 Home Assignment 1
.data
a1: .asciiz "The abs of "
a2: .asciiz " is "
.text
main:
        li $a0, -45             #load input parameter
        add $a1,$zero,$a0
        jal abs                 #jump and link to abs procedure
        nop
        add $s0, $zero, $v0
        li $v0, 4
        la $a0, a1
        syscall
        li $v0, 1
        move $a0, $a1
        syscall
        li $v0, 4
        la $a0, a2
        syscall
        li $v0, 1
        move $a0, $s0
        syscall
        li $v0, 10              #terminate
        syscall
```

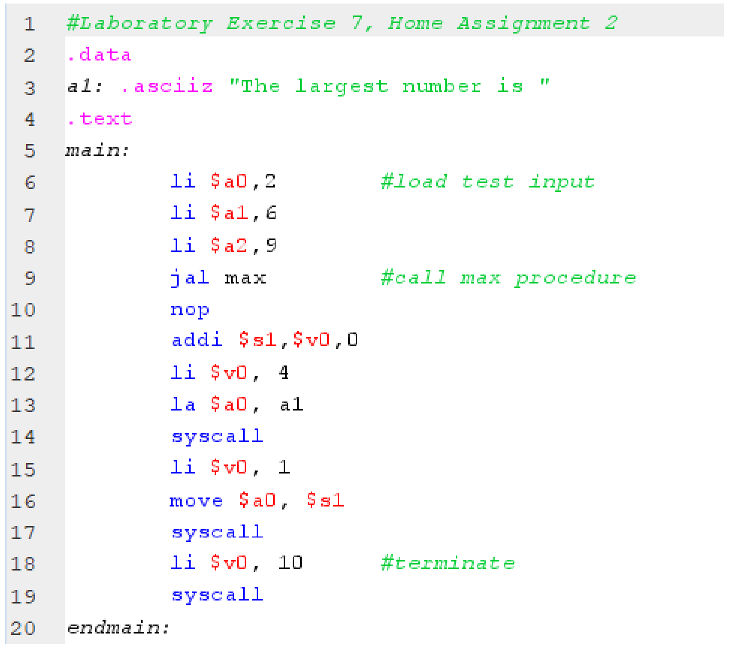


```
endmain:
#-----------------------------------------------------------------------
# function abs
# param[in] $a0 the interger need to be gained the absolute value
# return $v0 absolute value
#-----------------------------------------------------------------------
abs:
        sub $v0,$zero,$a0       #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
```

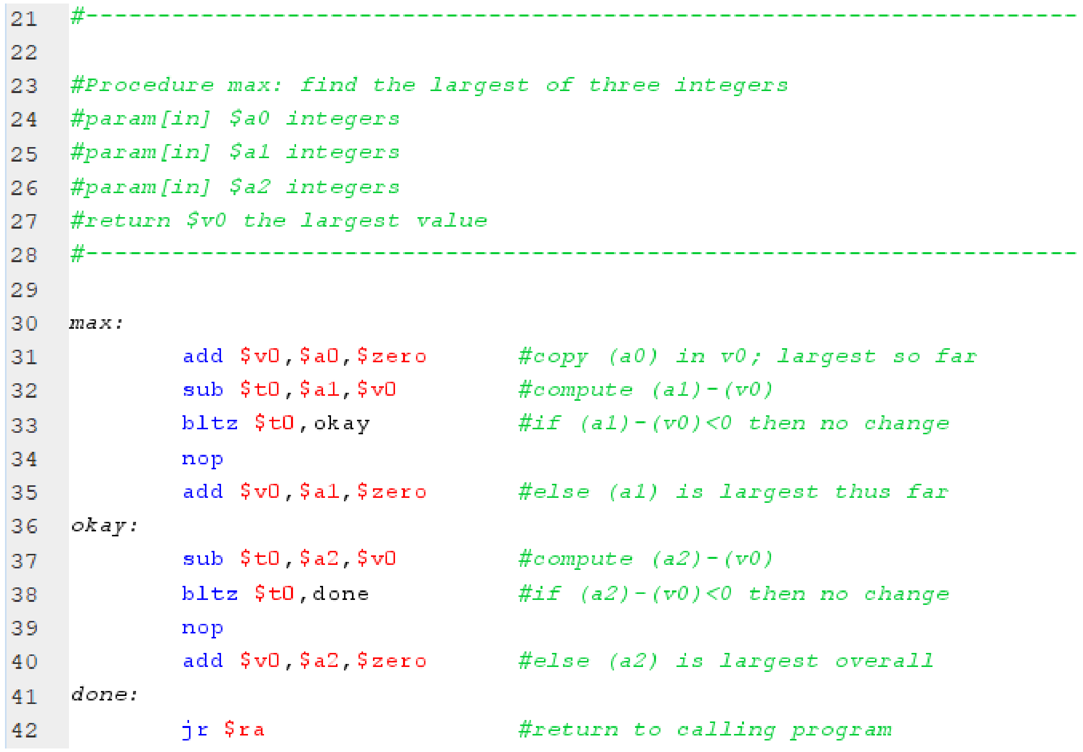


```
The abs of -45 is 45
-- program is finished running --
```

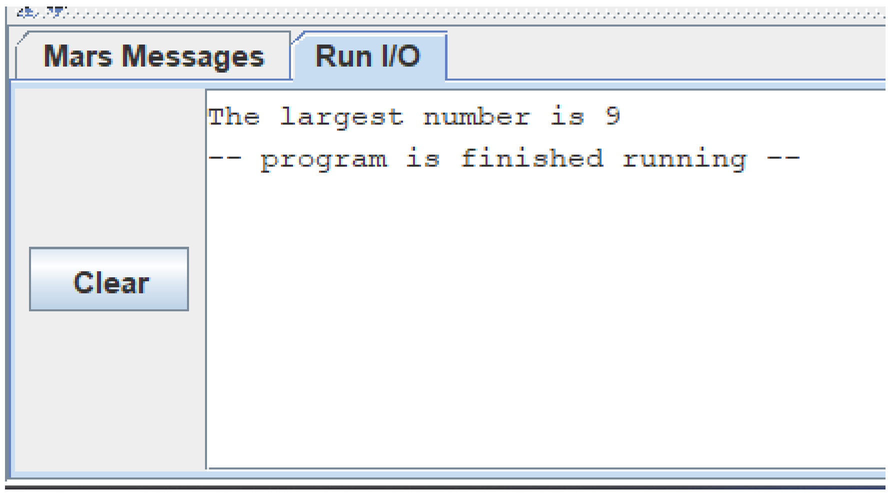
2. **Assignment 2:** *Create a new project to implement the program in Home Assignment 2. Compile and upload to simulator. Change input parameters (register $a0, $a1, $a2) and observe the memory when run the program step by step. Pay attention to register $pc, $ra to clarify invoking procedure process.*



```
#Laboratory Exercise 7, Home Assignment 2
.data
a1: .asciiz "The largest number is "
.text
main:
        li $a0,2        #load test input
        li $a1,6
        li $a2,9
        jal max         #call max procedure
        nop
        addi $s1,$v0,0
        li $v0, 4
        la $a0, a1
        syscall
        li $v0, 1
        move $a0, $s1
        syscall
        li $v0, 10      #terminate
        syscall
endmain:
```



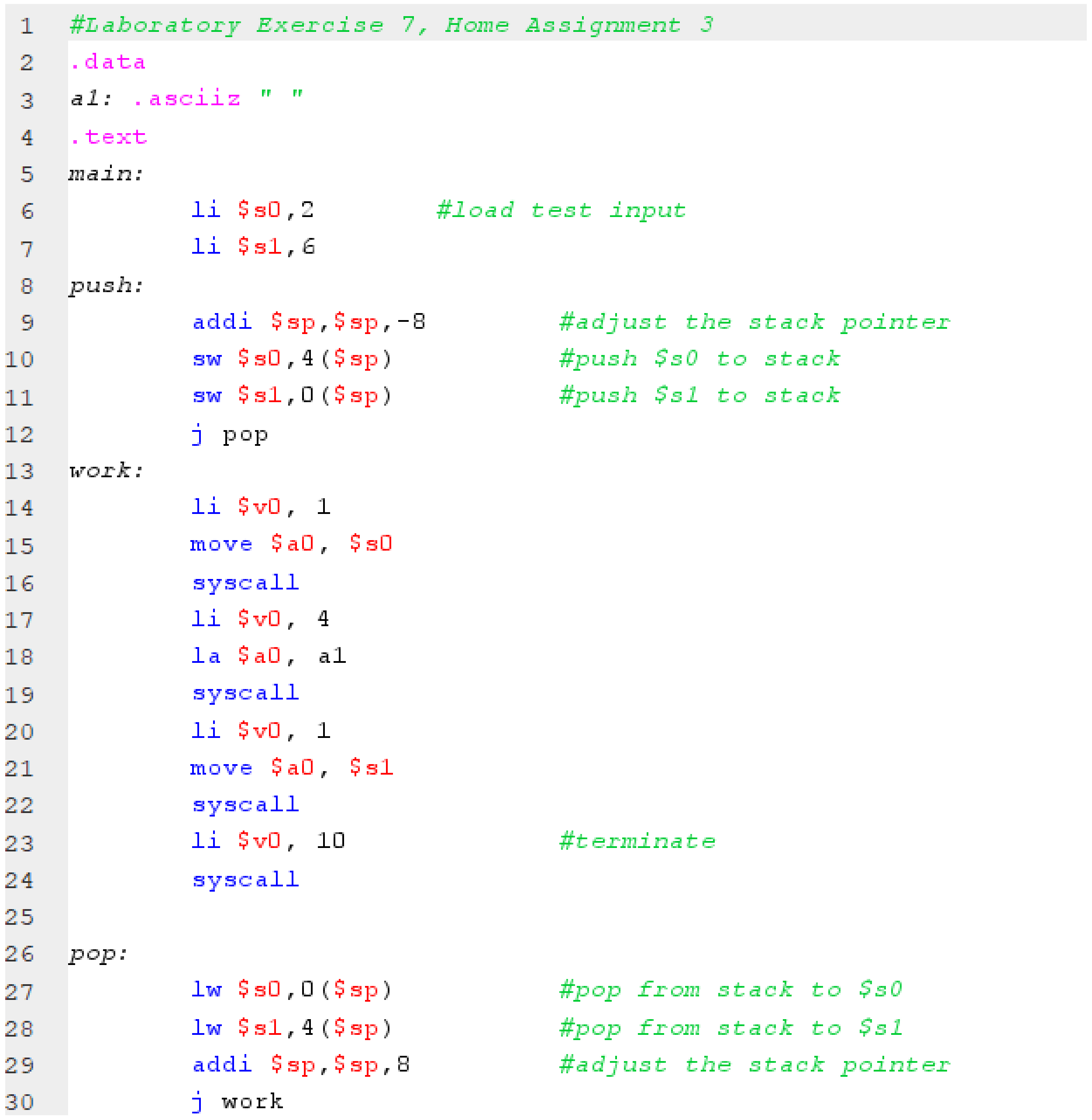
```
#---------------------------------------------------------------------

#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
```


Mars Messages
Run I/O
Clear

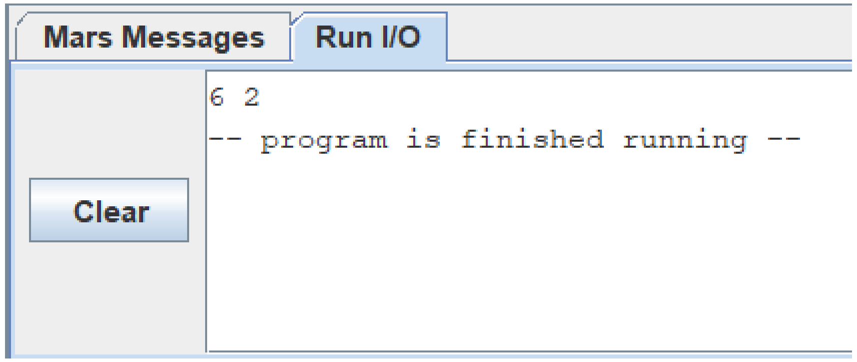
```
The largest number is 9
-- program is finished running --
```

3. **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.*

The input is $s0 = 2 and $s1 = 6, after that the program swap the value of 2 registers ⇒ $s0 = 6 and $s1 = 2



```
#Laboratory Exercise 7, Home Assignment 3
.data
a1: .asciiz " "
.text
main:
        li $s0,2        #load test input
        li $s1,6
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
        j pop
work:
        li $v0, 1
        move $a0, $s0
        syscall
        li $v0, 4
        la $a0, a1
        syscall
        li $v0, 1
        move $a0, $s1
        syscall
        li $v0, 10              #terminate
        syscall

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
        j work
```

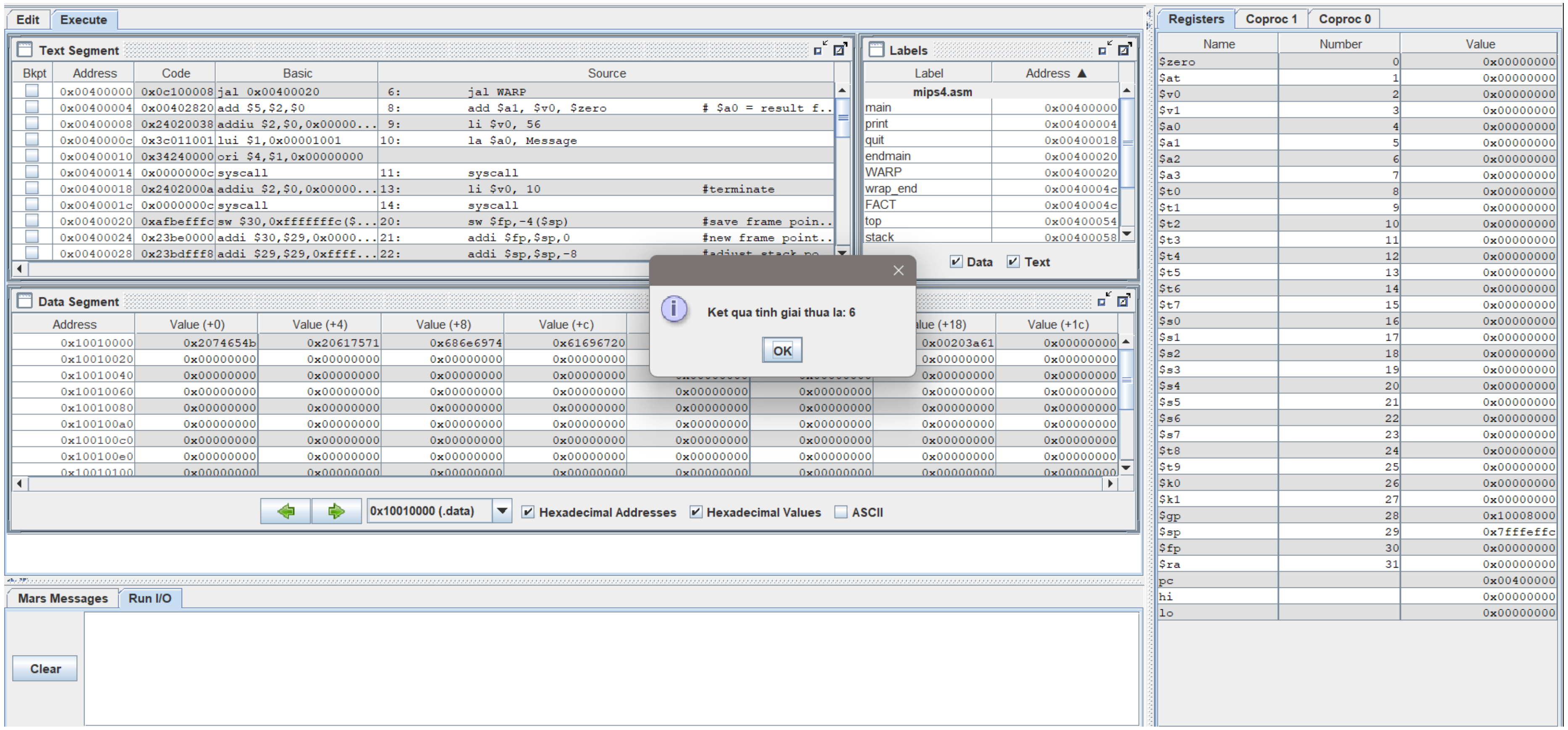


Mars Messages | Run I/O

```
6 2
-- program is finished running --
```

Clear

4. **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!).*



Edit | Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x0c100008	jal 0x00400020	6: jal WARP
	0x00400004	0x00402820	add $5,$2,$0	8: add $a1, $v0, $zero # $a0 = result f..
	0x00400008	0x24020038	addiu $2,$0,0x00000...	9: li $v0, 56
	0x0040000c	0x3c011001	lui $1,0x00001001	10: la $a0, Message
	0x00400010	0x34240000	ori $4,$1,0x00000000	
	0x00400014	0x0000000c	syscall	11: syscall
	0x00400018	0x2402000a	addiu $2,$0,0x00000...	13: li $v0, 10 #terminate
	0x0040001c	0x0000000c	syscall	14: syscall
	0x00400020	0xafbefffc	sw $30,0xfffffffc($...	20: sw $fp,-4($sp) #save frame poin..
	0x00400024	0x23be0000	addi $30,$29,0x0000...	21: addi $fp,$sp,0 #new frame point..
	0x00400028	0x23bdfff8	addi $29,$29,0xffff...	22: addi $sp,$sp,-8

Labels

Label	Address ▲
mips4.asm	
main	0x00400000
print	0x00400004
quit	0x00400018
endmain	0x00400020
WARP	0x00400020
wrap_end	0x0040004c
FACT	0x0040004c
top	0x00400054
stack	0x00400058

✓ Data ✓ Text

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	[illegible]	[illegible]	lue (+18)	Value (+1c)
0x10010000	0x2074654b	0x20617571	0x686e6974	0x61696720	[illegible]	[illegible]	0x00203a61	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	[illegible]	[illegible]	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	[illegible]	[illegible]	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ✓ Hexadecimal Addresses ✓ Hexadecimal Values ☐ ASCII

Ket qua tinh giai thua la: 6

OK

Mars Messages | Run I/O

Clear

Registers | Coproc 1 | Coproc 0

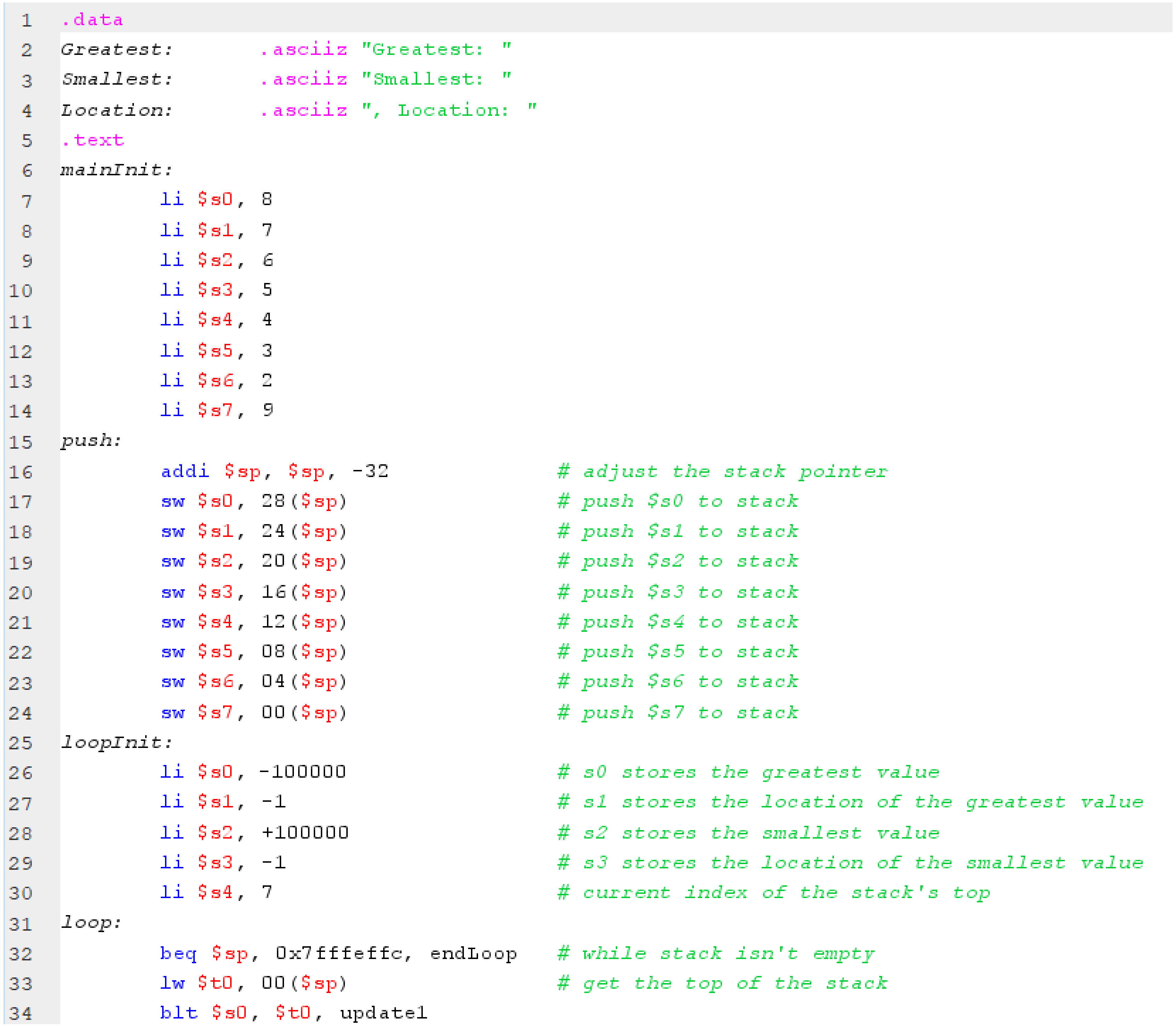
Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	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		0x00400000
hi		0x00000000
lo		0x00000000

Lúc bắt đầu WRAP, $sp = 0x7fffeffc

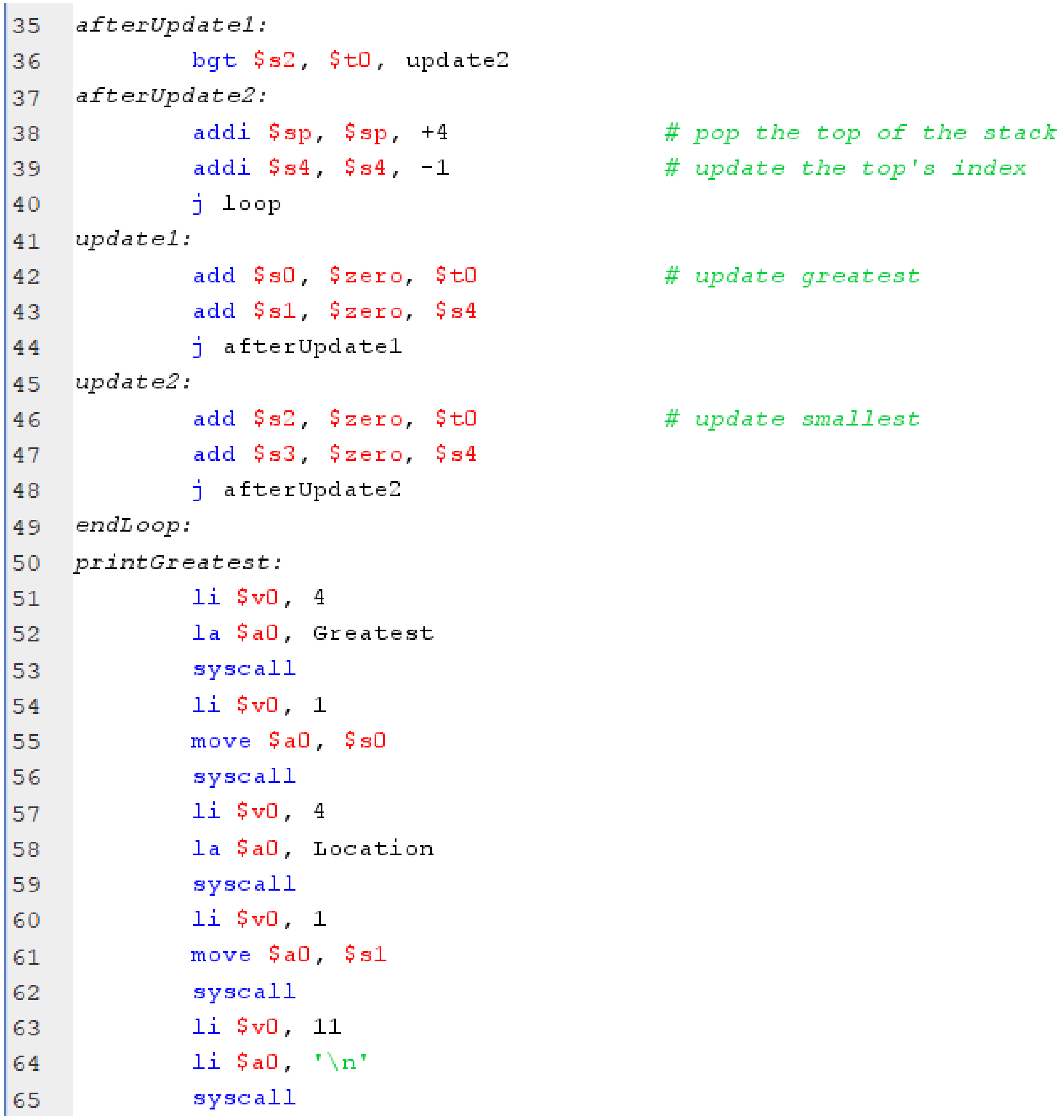
$a0 = 1	0x7fffefd0 <-new $sp (addi $sp,$sp,-12)	return
$ra = 0x00400080	0x7fffefd4	sw $ra,4($sp)
$fp = 0x7fffefe8 sw $fp,-4($sp)	0x7fffefd8	
$a0 = 2	0x7fffefdc <-new $sp (addi $sp,$sp,-12)	sw $a0,0($sp)
$ra = 0x00400080	0x7fffefe0	sw $ra,4($sp)
$fp = 0x7fffeff4 sw $fp,-4($sp)	0x7fffefe4	addi $fp,$sp,0 -> update $fp = 0x7fffefe8
$a0 = 3	0x7fffefe8 <-new $sp (addi $sp,$sp,-12)	sw $a0,0($sp)

$ra = 0x00400038	0x7fffefec	sw $ra,4($sp)
$fp = 0x7fffeffc sw $fp,-4($sp)	0x7fffeff0	addi $fp,$sp,0 -> update $fp = 0x7fffeff4
$ra = 0x00400004	0x7fffeff4 <- new $sp (addi $sp,$sp,-8)	sw $ra,0($sp)
$fp = 0x00000000 (init value) sw $fp,-4($sp)	0x7fffeff8	addi $fp,$sp,0 -> update $fp = 0x7fffeffc
$sp = 0x7fffeffc (init $sp)	0x7fffeffc	

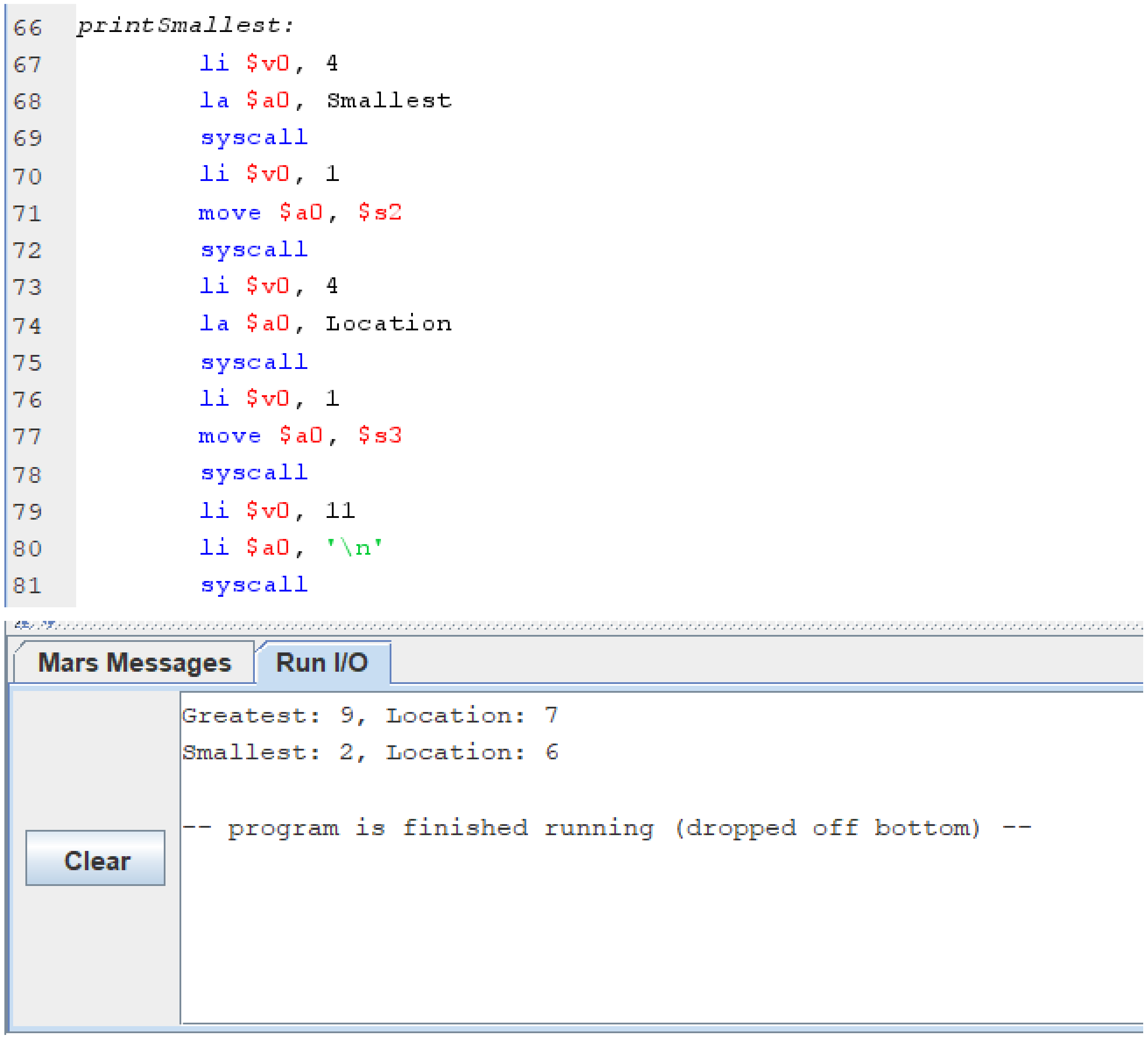
5. **Assignment 5:** *Write a procedure to find the largest, the smallest and these positions in a list of 8 elements that are stored in regsiters $s0 through $s7. For example: Largest: 9,3 => The largest element is stored in $s3, largest value is 9 Smallest: -3,6 => The smallest element is stored in $s6, smallest value is -3 Tips: using stack to pass arguments and return results.*



```
.data
Greatest:       .asciiz "Greatest: "
Smallest:       .asciiz "Smallest: "
Location:       .asciiz ", Location: "
.text
mainInit:
        li $s0, 8
        li $s1, 7
        li $s2, 6
        li $s3, 5
        li $s4, 4
        li $s5, 3
        li $s6, 2
        li $s7, 9
push:
        addi $sp, $sp, -32              # adjust the stack pointer
        sw $s0, 28($sp)                 # push $s0 to stack
        sw $s1, 24($sp)                 # push $s1 to stack
        sw $s2, 20($sp)                 # push $s2 to stack
        sw $s3, 16($sp)                 # push $s3 to stack
        sw $s4, 12($sp)                 # push $s4 to stack
        sw $s5, 08($sp)                 # push $s5 to stack
        sw $s6, 04($sp)                 # push $s6 to stack
        sw $s7, 00($sp)                 # push $s7 to stack
loopInit:
        li $s0, -100000                 # s0 stores the greatest value
        li $s1, -1                      # s1 stores the location of the greatest value
        li $s2, +100000                 # s2 stores the smallest value
        li $s3, -1                      # s3 stores the location of the smallest value
        li $s4, 7                       # current index of the stack's top
loop:
        beq $sp, 0x7fffeffc, endLoop    # while stack isn't empty
        lw $t0, 00($sp)                 # get the top of the stack
        blt $s0, $t0, update1
```



```
afterUpdate1:
        bgt $s2, $t0, update2
afterUpdate2:
        addi $sp, $sp, +4              # pop the top of the stack
        addi $s4, $s4, -1              # update the top's index
        j loop
update1:
        add $s0, $zero, $t0            # update greatest
        add $s1, $zero, $s4
        j afterUpdate1
update2:
        add $s2, $zero, $t0            # update smallest
        add $s3, $zero, $s4
        j afterUpdate2
endLoop:
printGreatest:
        li $v0, 4
        la $a0, Greatest
        syscall
        li $v0, 1
        move $a0, $s0
        syscall
        li $v0, 4
        la $a0, Location
        syscall
        li $v0, 1
        move $a0, $s1
        syscall
        li $v0, 11
        li $a0, '\n'
        syscall
```



```
printSmallest:
        li $v0, 4
        la $a0, Smallest
        syscall
        li $v0, 1
        move $a0, $s2
        syscall
        li $v0, 4
        la $a0, Location
        syscall
        li $v0, 1
        move $a0, $s3
        syscall
        li $v0, 11
        li $a0, '\n'
        syscall
```

```
Greatest: 9, Location: 7
Smallest: 2, Location: 6

-- program is finished running (dropped off bottom) --
```