Week 7

Assignment 1

Source code and explanation:

```
#Laboratory Week 7, Assignment 1

.text

main:
	li $a0, -45 #init input
	jal abs		#jump and link to procedure (assign $ra to current
program counter value)
	nop

abs:
	sub $t0, $zero, $a0	#inverse $a0 value
	bltz $a0, done		#if $a0 < 0 then go to 'done'
	nop
	add $t0, $a0, $zero	#else set $t0 to $a0

done:
	jr $ra			#jump back to after jal opcode
```

Run results:

Case 1:

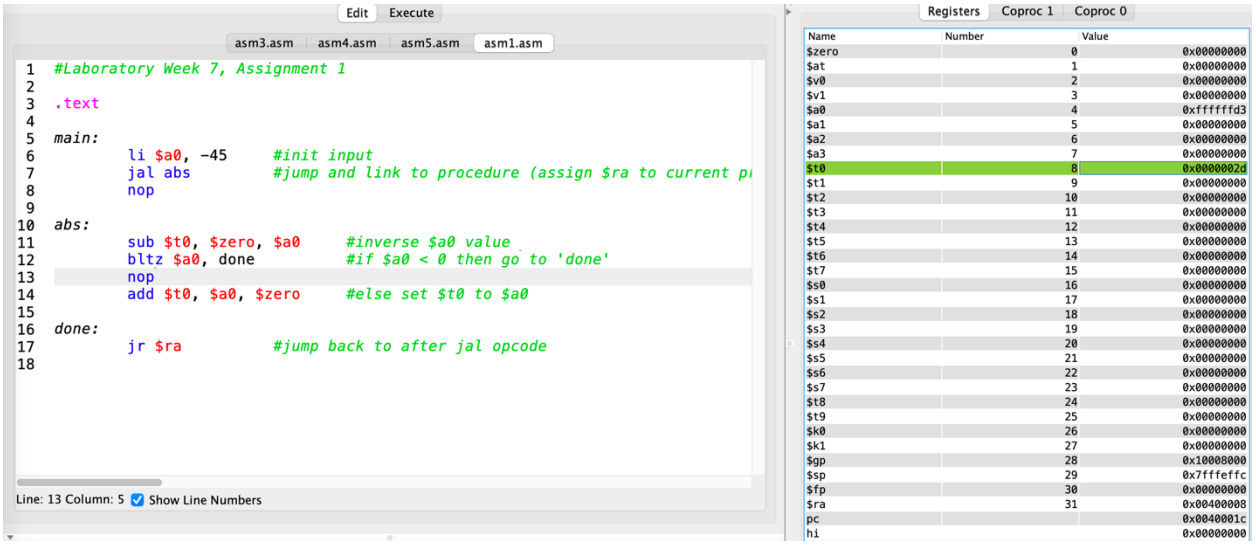
Input: -45 (0xfffffd3)

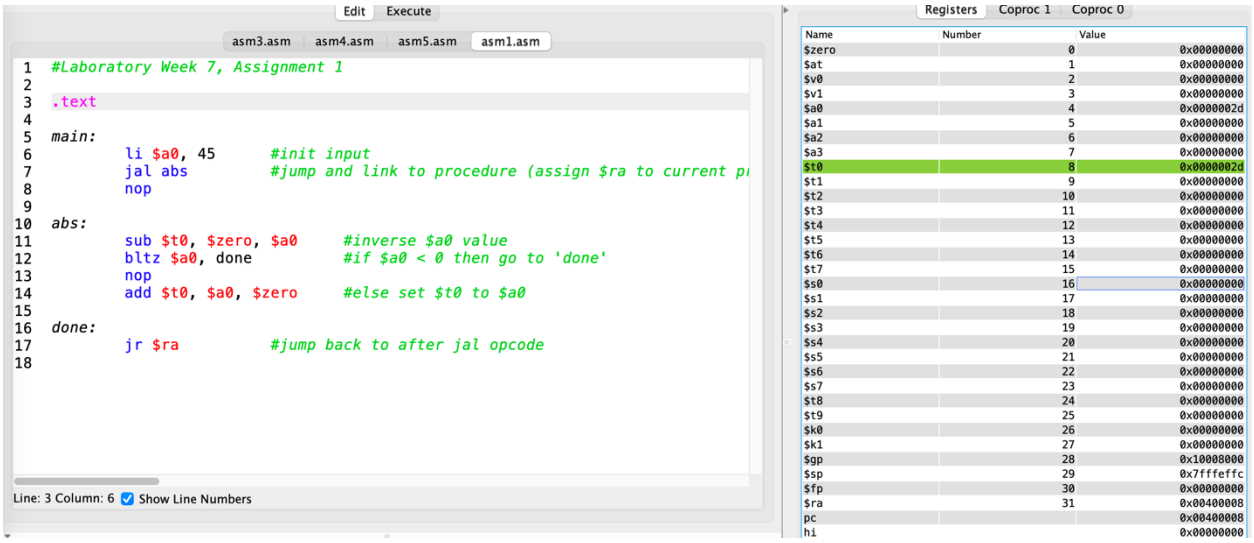
Output: 45 (0x0000002d)

Case 2:

Input: 45 (0x0000002d)

Output: 45 (0x0000002d)



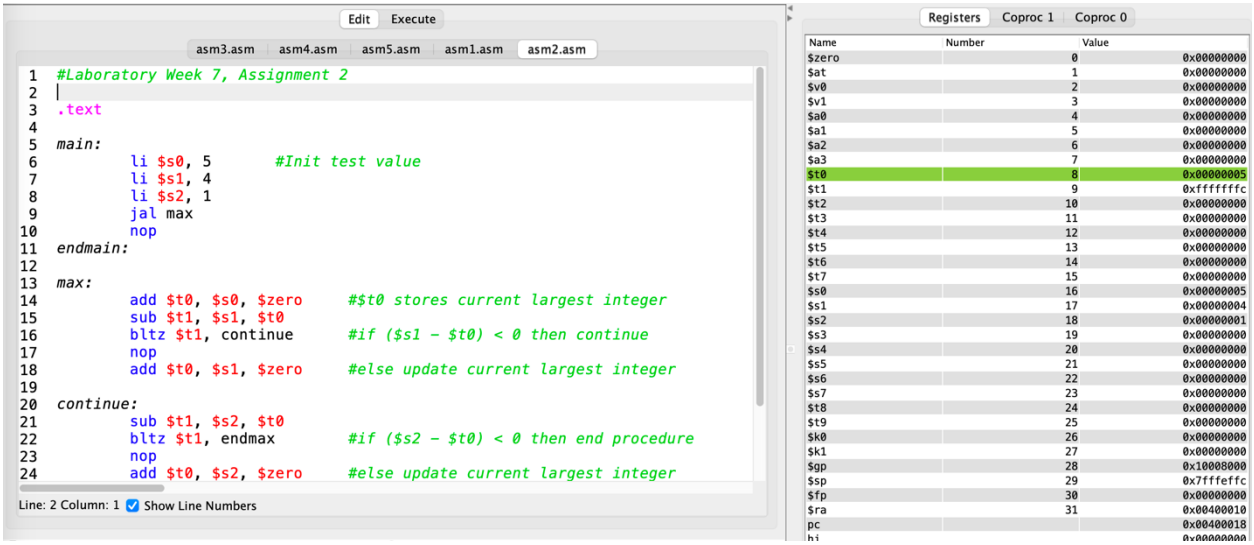


Assignment 2

Source code and explanation

```
#Laboratory Week 7, Assignment 2

.text

main:
      li $s0, 5   #Init test value
      li $s1, 4
      li $s2, 1
      jal max
      nop
endmain:

max:
      add $t0, $s0, $zero     #$t0 stores current largest integer
      sub $t1, $s1, $t0
      bltz $t1, continue      #if ($s1 - $t0) < 0 then continue
      nop
      add $t0, $s1, $zero     #else update current largest integer

continue:
      sub $t1, $s2, $t0
      bltz $t1, endmax #if ($s2 - $t0) < 0 then end procedure
      nop
      add $t0, $s2, $zero     #else update current largest integer

endmax:
      jr $ra            #return back to main
```

Run results:



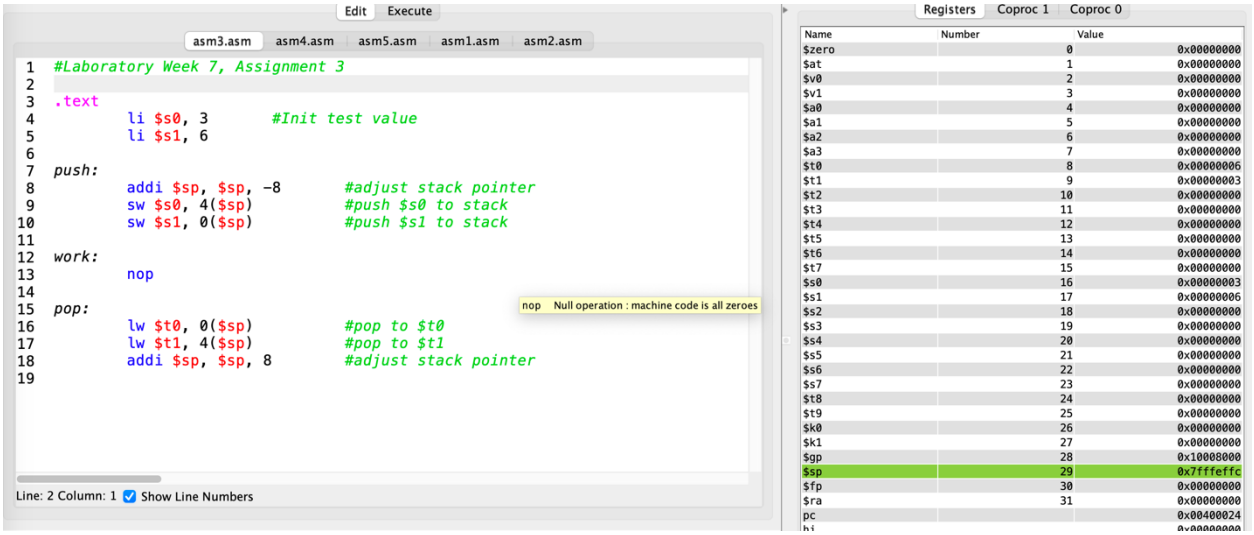
Result is stored in $t0 register.

Assignment 3

Source code and explanation:

```
#Laboratory Week 7, Assignment 3

.text
      li $s0, 3   #Init test value
      li $s1, 6

push:
      addi $sp, $sp, -8#adjust stack pointer
      sw $s0, 4($sp)          #push $s0 to stack
      sw $s1, 0($sp)          #push $s1 to stack

work:
      nop

pop:
      lw $t0, 0($sp)          #pop to $t0
      lw $t1, 4($sp)          #pop to $t1
      addi $sp, $sp, 8 #adjust stack pointer
```

Run results: Stack's values are popped to $t0 and $t1 register and follow the 'First in last out' rule



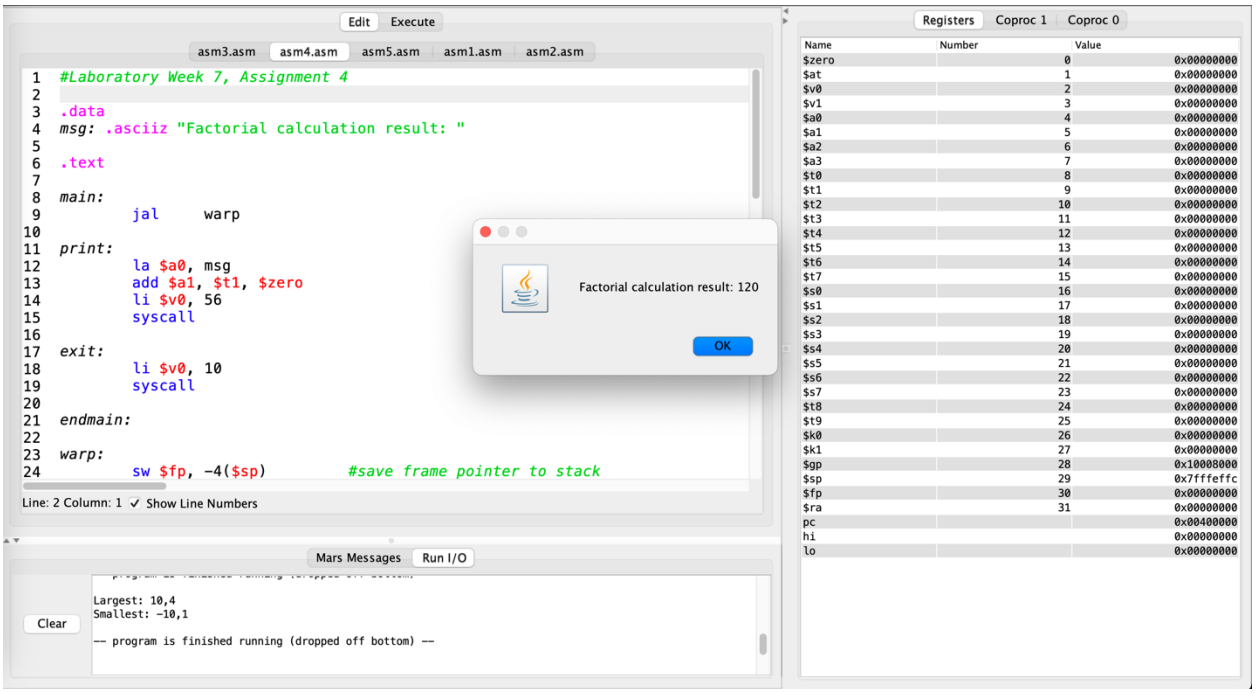
Assignment 4

Source code and explanation:

```
#Laboratory Week 7, Assignment 4

.data
msg: .asciiz "Factorial calculation result: "

.text

main:
      jal   warp

print:
      la $a0, msg
      add $a1, $t1, $zero
      li $v0, 56
      syscall

exit:
      li $v0, 10
      syscall

endmain:

warp:
      sw $fp, -4($sp)         #save frame pointer to stack
      addi $fp, $sp, 0 #save stack's top address to frame pointer
      addi $sp, $sp, -8#adjust stack pointer to make space
      sw $ra, 0($sp)          #save return address to stack

      li $s0, 5   #Init test input N
      jal fact    #jump to procudure factorial
      nop
      lw $ra, 0($sp)    #retrieve return address
```

```
      addi $sp, $fp, 0 #return stack pointer
      lw $fp, -4($sp)        #return frame pointer
      jr $ra

warpend:

fact:
      sw $fp, -4($sp)        #save frame pointer to stack
      addi $fp, $sp, 0 #save stack's top address to frame pointer

top:
      addi $sp, $sp, -12     #adjust stack pointer to make space for $fp,
$ra, $s0

stack:
      sw $ra, 4($sp)         #save return address to stack
      sw $s0, 0($sp)         #save N to stack

      slti $t0, $s0, 2 #if N < 2 false
      beqz $t0, recursive    #then continue recursive
      nop
      li $t1, 1        #else resultN! = 1
      j done
      nop

recursive:
      addi $s0, $s0, -1#N Decrement
      jal fact         #recursive call
      nop
      lw $t2, 0($sp)         #Load current N
      mult $t2, $t1          #Perform multiplication
      mflo $t1

done:
      lw $ra, 4($sp)         #retrieve return address
      lw $s0, 0($sp)         #retrieve initial N
      addi $sp, $fp, 0 #retrieve stack pointer
      lw $fp, -4($sp)        #retrieve frame pointer
      jr $ra                 #jump back

factend:
```

Run result: With test value N = 5



Assignment 5

Source code and explanation:

```
#Laboratory Week 7, Assignment 5

.data
max_message: .asciiz "Largest: "
min_message: .asciiz "Smallest: "
comma: .asciiz ","
newline: .asciiz "\n"

.text
main:
      li $s0, -2
      li $s1, -10
      li $s2, 5
      li $s3, 6
      li $s4, 10
      li $s5, 2
      li $s6, -3
      li $s7, 9
      addi $sp, $sp, -32      #Allocate space in stack

push:
      sw $s7, 0($sp)          #Push to stack
      addi $sp, $sp, 4
      sw $s6, 0($sp)
      addi $sp, $sp, 4
      sw $s5, 0($sp)
      addi $sp, $sp, 4
```

```
      sw $s4, 0($sp)
      addi $sp, $sp, 4
      sw $s3, 0($sp)
      addi $sp, $sp, 4
      sw $s2, 0($sp)
      addi $sp, $sp, 4
      sw $s1, 0($sp)
      addi $sp, $sp, 4
      sw $s0, 0($sp)
      addi $sp, $sp, 4

find_min_max:
      li $t3, 0   #$t3 = 0 (i)
      li $t4, 8   #Loop condition
      lw $k0, -4($sp)  #$k0 = $s0 => initial value
      lw $k1, -4($sp)  #$k1 = $s0
                  #$k0 = max, $k1 = min, $t1 = index of max, $t0 = index of
min, $a0 = value, $a1 = value position

load:
      addi $t3, $t3, 1 #$t3 = $t3 + 1 = i + 1
      add $t5, $t3, $t3#$t5 = 2*$t3 = 2*i
      add $t5, $t5, $t5#$t5 = 4*$t3 = 4*i
      sub $t5, $zero, $t5    #$t1 = -4*i
      add $t2, $t5, $sp#$t2 = $sp - 4*$t1 = &A[i]
      lw $a0, 0($t2)         #$a0 = *$t2 = A[i]
      addi $a1, $t3, -1#$a1 = $t0 - 1 = position of A[i]
      beq $a1, $t4, print    #if $a1 == $t4, jump to print
      nop

check:
      slt $a2, $k0, $a0#if $k0 < $a0 => $a1 = 1 ($a0 <= $k0)
      beqz $a2, min          #if $a1 == 0 ($a0 <= $k0 => no change), jump
to min
      nop
      addi $k0, $a0, 0 #$k0 = $a0
      addi $t0, $a1, 0 #$t0 = $a1 (new max position)

min:
    slt $a2, $a0, $k1  #if $k1 <= $a0 => jump, if $a0 < $k1 => $a1 = 1
    beqz $a2, load     #if $a1 == 0 ($k1 <= $a0 => no change), jump to
load (new loop)
    nop
    addi $k1, $a0, 0   #$k1 = $a0
    addi $t1, $a1, 0   #$t1 = $a1 (new min position)
    j load

print:
    li $v0, 4          #Print max result
    la $a0, max_message
    syscall
    li $v0, 1
    addi $a0, $k0, 0
    syscall
```

```
    li $v0, 4
    la $a0, comma
    syscall
    li $v0, 1
    addi $a0, $t0, 0
    syscall
    li $v0, 4
    la $a0, newline
    syscall

    li $v0, 4           #Print min result
    la $a0, min_message
    syscall
    li $v0, 1
    addi $a0, $k1, 0
    syscall
    li $v0, 4
    la $a0, comma
    syscall
    li $v0, 1
    addi $a0, $t1, 0
    syscall
    li $v0, 4
    la $a0, newline
    syscall
```

Run Results: