

Matthew Michael Sherlin
Dr. Augustine Samba
Computer Organization
October 29, 2020

Assembly Code File part 2:

```
.data
    prompt1: .asciiz "Enter two values into each register. The program will swap and print results."
    prompt2: .asciiz "\nEnter a value for x: "
    prompt3: .asciiz "Enter a value for y: "
    message1: .asciiz "\nSwapping x and y."
    message2: .asciiz "\n\nThe value of x is now: "
    message3: .asciiz "\n\nThe value of y is now: "

.text

li $v0, 4      #reading the first prompt
la $a0, prompt1
syscall

li $v0, 4      #reading the second prompt
la $a0, prompt2
syscall

li $v0, 5      #entering user input
syscall

add $s0, $zero, $v0    #setting value to s0 without move

li $v0, 4      #reading the third prompt
la $a0, prompt3
syscall

li $v0, 5      #entering user input
syscall

add $s1, $zero, $v0    #setting value to s1 without move

li $v0, 4      #reading the first message
la $a0, message1
syscall

add $t0, $zero, $s0    #swap s0 into temporary register
add $s0, $zero, $s1    #swap s1 into s0
add $s1, $zero, $t0    #retrieve s0 original value from temporary and place into s1
```

```

li $v0, 4          #reading the second message
la $a0, message2
syscall

li $v0, 1          #displaying final answer pt1
add $a0, $zero, $s0
syscall

li $v0, 4          #reading the final message
la $a0, message3
syscall

li $v0, 1          #displaying final answer pt2
add $a0, $zero, $s1
syscall

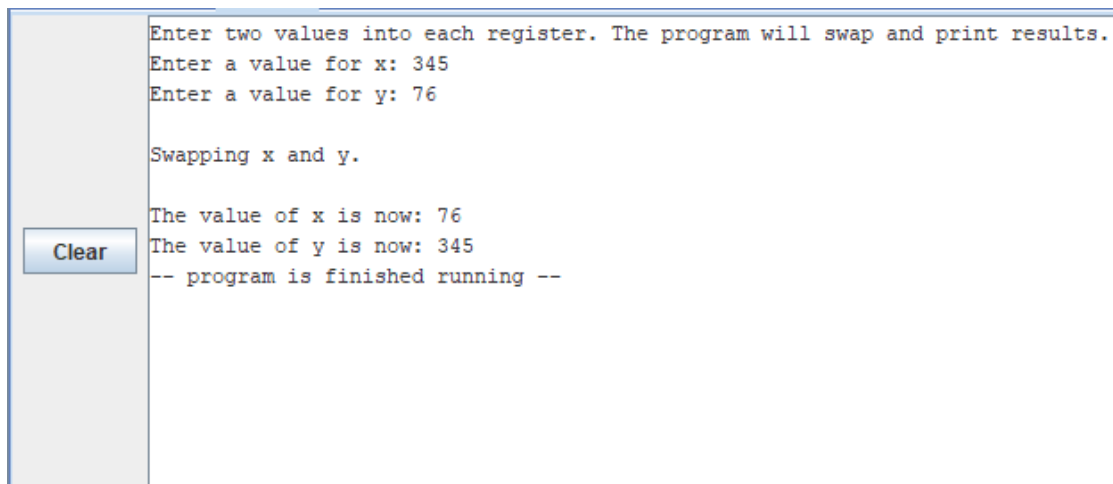
li $v0, 10         #terminate program run and exit
syscall

```

Project Implementation:

To begin this project, I used basic calls to read the prompts and retrieve the inputs from the console. As per Dr. Samba, I was not allowed to use 'move' for the second section of this lab, so I simply used the add instruction to swap my values using a single temporary register. First I added the value of s0 into a temporary register with \$zero, next, I added s1 into s0 using \$zero again, and finally I added the value stored in the temporary register back into s1 using \$zero also. This all together swapped the values in s0 and s1 without using move instruction. Finally, I used some basic syscall, li and la instructions to read the final messages and print the results of both registers. I also used add instruction here to store the values in the saved registers to \$a0 to print them out.

Working Code Screen Print:



```

Enter two values into each register. The program will swap and print results.
Enter a value for x: 345
Enter a value for y: 76

Swapping x and y.

The value of x is now: 76
The value of y is now: 345
-- program is finished running --

```

Transcription:

Enter two values into each register. The program will swap and print results.

Enter a value for x: 345

Enter a value for y: 76

Swapping x and y.

The value of x is now: 76

The value of y is now: 345

-- program is finished running --

Conclusion:

To conclude, I enjoyed this lab using MIPS also. I did not really learn much besides nailing down how to move values between registers using add instead of move. Besides that, I did not have really any issues at all during this lab. I had a fun time implementing this!