Matthew Michael Sherlin
Dr. Augustine Samba
Computer Organization
October 22, 2020

Assembly Code File:

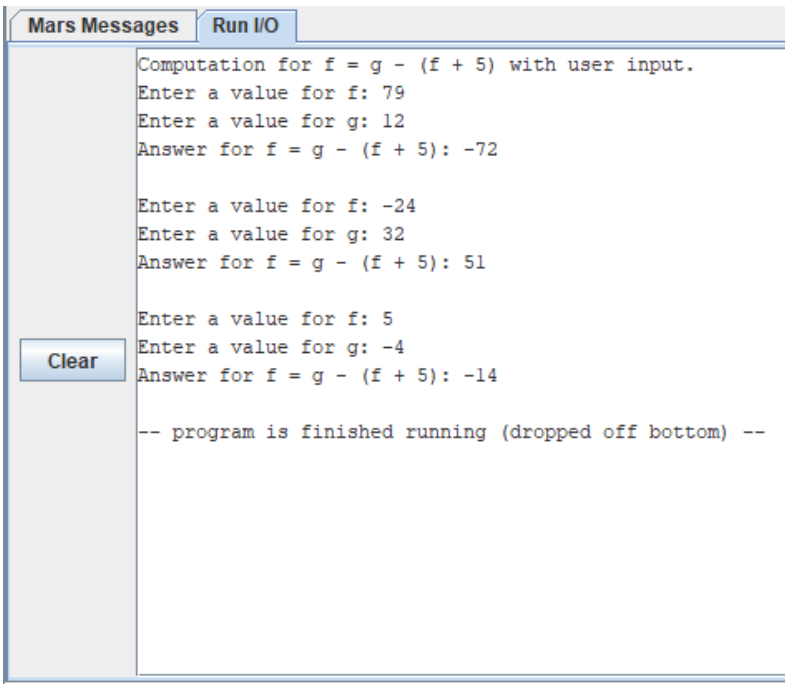
```
.data
        prompt1: .asciiz "Computation for f = g - (f + 5) with user input."
        prompt2: .asciiz "\nEnter a value for f: "
        prompt3: .asciiz "Enter a value for g: "
        message: .asciiz "Answer for f = g - (f + 5): "
        space: .asciiz "\n"

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

li $s3, 3               # s3 is the 3 to end loop
li $t6, 0               # t6 is the counter (i)
loop:
beq $t6, $s3, end       # if t6 == 3 (s3) we are done

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

li $v0, 5               #entering user input
syscall

move $t0, $v0           #moving user input into temporary register

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

li $v0, 5               #entering user input
syscall

move $s1, $v0           #moving user input into saved register

addi $t3, $t0, 5        #temp for (f+5)
sub $t4, $s1, $t3       #subtracting g - (f+5)

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

```
li $v0, 1                #displaying final answer
move $a0, $t4
syscall

li $v0, 4                #adding spacing for cleanliness of text
la $a0, space
syscall


addi $t6, $t6, 1         # add 1 to t1
j loop                   # jump back to the top
end:
```

Project Implementation:

In order to get this program to work, I first looked to create the loop for the program to run 3 separate times. I put the initial message outside of the loop to eliminate unnecessary repetition, then I placed everything else inside of the loop. I used branch if equal statement (beq) and a value in temp 6 that is incremented each time to create the loop. Inside of the loop, I allowed the prompts to be read, then I took user input for both value f and g and placed them into registers. After the values are stored, I did simple arithmetic with an add and subtract to get the final value which is stored in $t4. This message is shown, then the loop is incremented, then there is a jump back up to the top of the loop. Finally, there is the end: statement that coincides with the beq statement. This is the entire project summed up.

Working Code Screen Print:



Transcription:

Computation for f = g - (f + 5) with user input.
Enter a value for f: 79
Enter a value for g: 12
Answer for f = g - (f + 5): -72

Enter a value for f: -24
Enter a value for g: 32
Answer for f = g - (f + 5): 51

Enter a value for f: 5
Enter a value for g: -4
Answer for f = g - (f + 5): -14

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

Conclusion:

To conclude, I learned a lot during this project, while not facing very many problems at all. I really learned that using MIPS is not too difficult if you think about the problem logically and take it slow to get it down. I learned how to display a message and receive a user input really well. The arithmetic was not too difficult, but it was the first time I actually coded it in MIPS. The main only issue that I faced was not a very dire issue, however figuring out the spacing of the text in the output gave me a little bit of an issue. I kept getting double spacing between prompts which took me a second to figure out, however it was only an issue of where I had place '\n' within my prompts. I really enjoyed this project actually, and I am excited to do more coding in MIPS.