1. Program 1:
    - Inputs:
        - UserMessage, starts off empty is filled by user text from the console
    - OutPuts:
        - FirstMessage, a simple asciiz string that outputs the message, “Hello, may I have your name, please?
        - SecondMessage, simple asciiz string that outputs the message, “Hello”.
        - UserMessage, after storing the user input second message and user message are output in that order to achieve the desired output and formating
    - Program 2:
        - Inputs:
            - User inputs four positive integers, a b c d, that are stored in registers $t0-$3 respectively
        - Outputs:
            - In the last step of function F, specifically ((a+b)-(c+d)) + (b+3), the add command is used to call the registers containing the two halves, ((a+b)-(c+d)) and (b+3), and the result is stored in $a0. This is then printed out with the commands li $v0, 1 and a syscall.
2. Program 1:

    The MIPS code prompts the user to enter their name. This is then stored in the user message data. Once captured the second message is output followed by the user message. There is no new line tag in the second message so when it is output the user input will be output on the same line.

    Program 2:

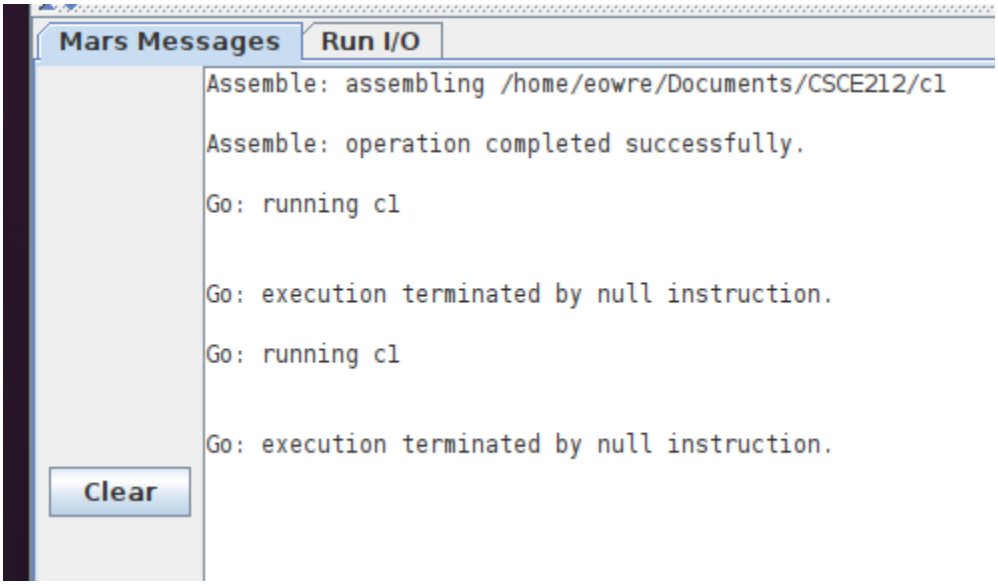
    The user is prompted to enter 4 positive integers, using four syscalls these integers are stored in registers $t0-$t3. Each step of function F is then followed using PEMDAS to ensure correct values are calculated. As each of these steps are taken the results are stored in new registers. Once the preliminary addition has been done, i.e. a+b c+d b+3, the subtract and add functions are used to complete the calculations. NOTE, the last function called is adding ((a+b)-(c+d)) and (b+3), this result is stored in $a0 to easily print out to console using li $v0, 1 and a syscall.
3. Program 1:

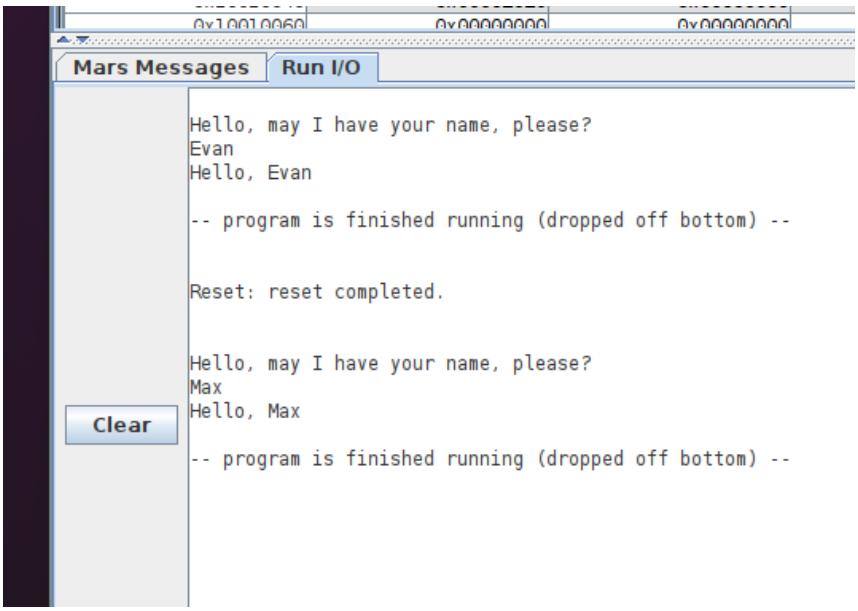
| | |
|---|---|
| $v0 4 | Syscall print message |
| $a0 firstmessage | Load message for syscall print |
| $v0 8 | Syscall read string |
| $a0 usermessage | Save user input |
| $a1 20 | String buffer length |
| $v0 4 | Syscall print message |
| $a0 secondmessage | Load message for syscall print |
| $a0 usermessage | Load message for syscall print |

Program 2:

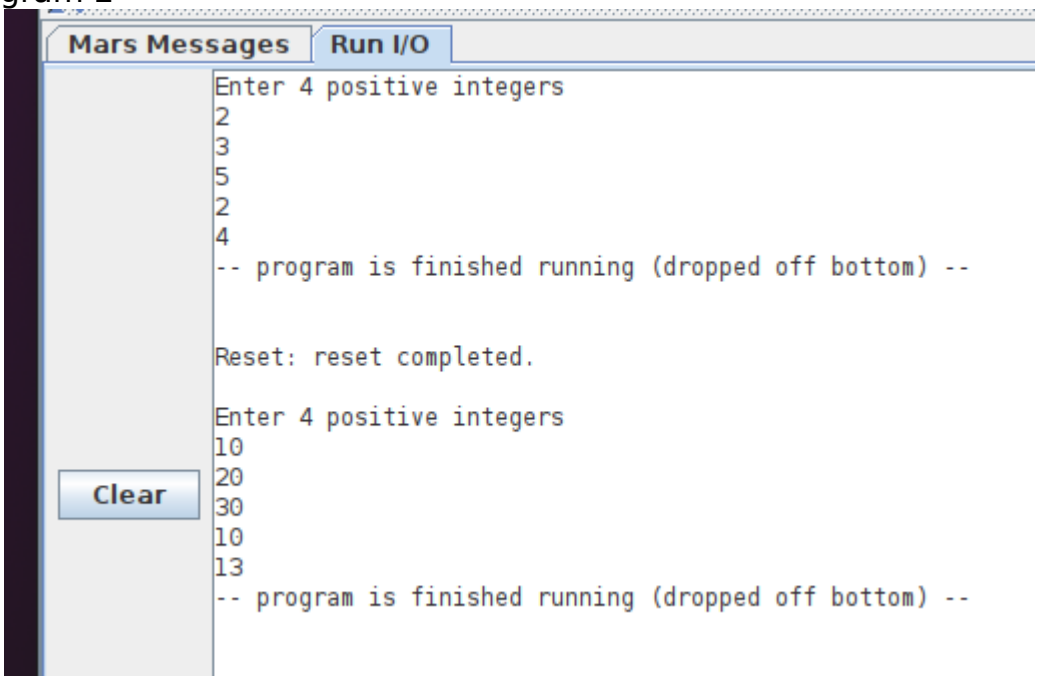
| | |
|---|---|
| $v0 5 | Syscall read integers |
| $t0 | Stores first int |
| $t1 | Stores second int |
| $t2 | Stores third int |
| $t3 | Stores fourth int |
| $t4 | Stores $t0+$t1 |
| $t5 | Stores $t2+$t3 |
| $t6 | Stores $t1+3 |
| $t7 | Stores $t4-$t5 |
| $a0 | Stores $t7+$t6 for syscall print |
| $v0 1 | Syscall print integers |

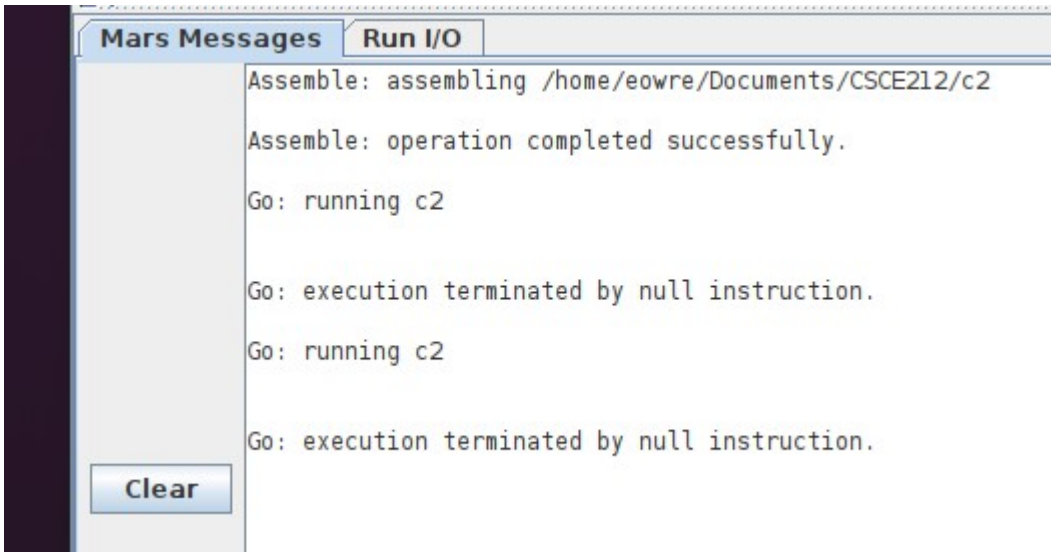
4. I learned
    1. How to create custom messages,
    2. How to create data to hold a user string
    3. How to store strings and integers from the user
    4. How Syscall works and what the different numbers mean for $v0
    5. How to step through steps while running MIPS in order to find bugs
5. Program 1



5. Program 2


Mars Messages
Run I/O
Enter 4 positive integers
2
3
5
2
4
-- program is finished running (dropped off bottom) --
Reset: reset completed.
Enter 4 positive integers
10
20
30
10
13
-- program is finished running (dropped off bottom) --
Clear


Mars Messages
Run I/O
Assemble: assembling /home/eowre/Documents/CSCE212/c2
Assemble: operation completed successfully.
Go: running c2
Go: execution terminated by null instruction.
Go: running c2
Go: execution terminated by null instruction.
Clear