Project 1 Lab Report

1.0 Program Input/Output

The first program we created for the project was dramatically simpler then the second one. To start the first program, we first outputted a greeting to the user along with asking the user for their name. The user then inputted their name. We then returned an output that recognized the user's input and reprinted it back out. That ended the first program. The second program began with outputting asking the user for a positive integer, a. We then stored this value and collected positive integers for variables b, c, and d. We then output the result of the function that we were given using the values we gathered from the user.

2.0 Program Design

To begin the first program, we first established two written messages and one other data storage that was used to store user input. We began by prompting the user with a greeting followed by asking them to enter their name. We did this by calling one of our pre-established written messages. The user then input their name, which was stored in our location for user input. We then printed out a generic welcome back out to the user. Directly following the generic “welcome,” we followed it with the user's unique name. To start the second program, we first established 5 messages in data. 4 of the messages were created to prompt the user to input integers for their variables. The last was created for when the final answer would be printed out. After establishing our text, we began the program by asking the user for each individual variable's value. We then took each individual value and stored them in different instances of memory. After collecting all four variables, we conducted multiple math operations.We first added a and b together, and c and d, and b and 3 together. We took these new values and stored these in memory. We then combined b + 3 and a and b together into a new unit in memory. Next, we took this new sum and subtract it by the sum of c + d in memory. This became our final answer. After completing the math operations, we then printed out the final answer back out to the user. We did this by first printing out our

established message in text. Then, we pulled our calculated value out of memory and printed it out to the user.

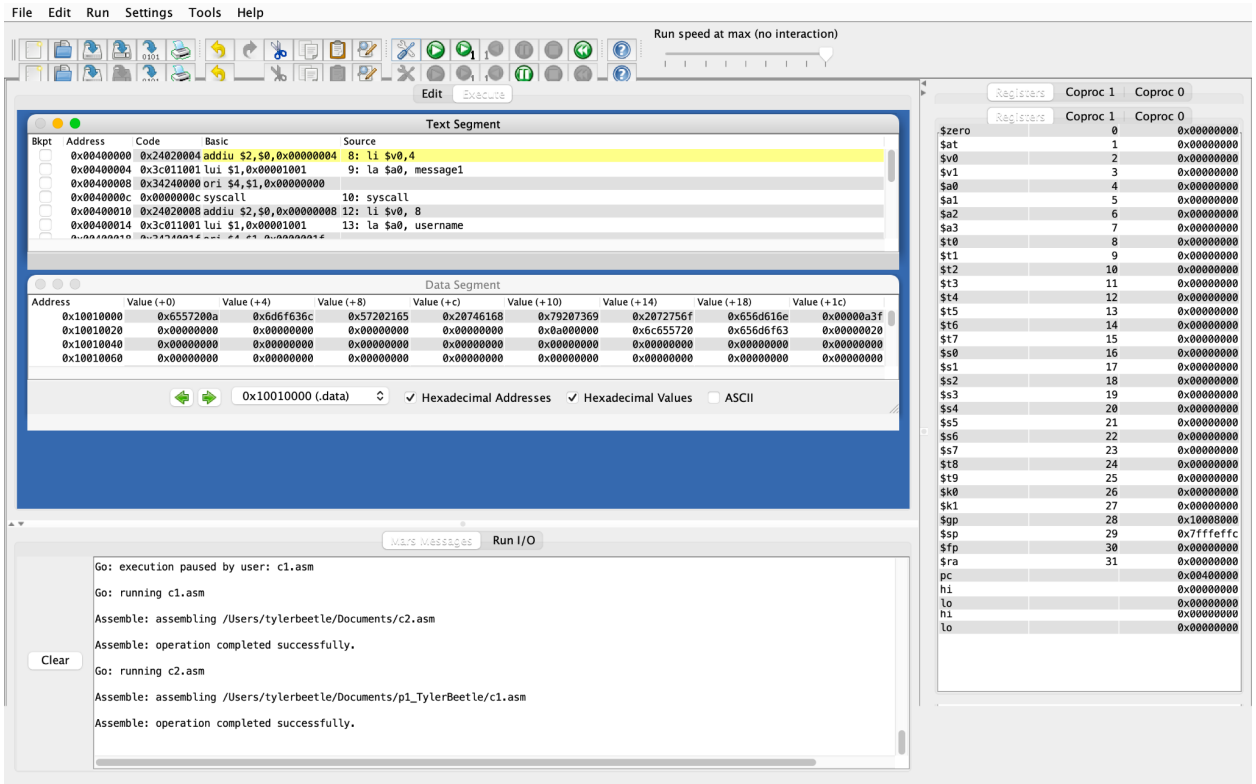
3.0 Symbol Table

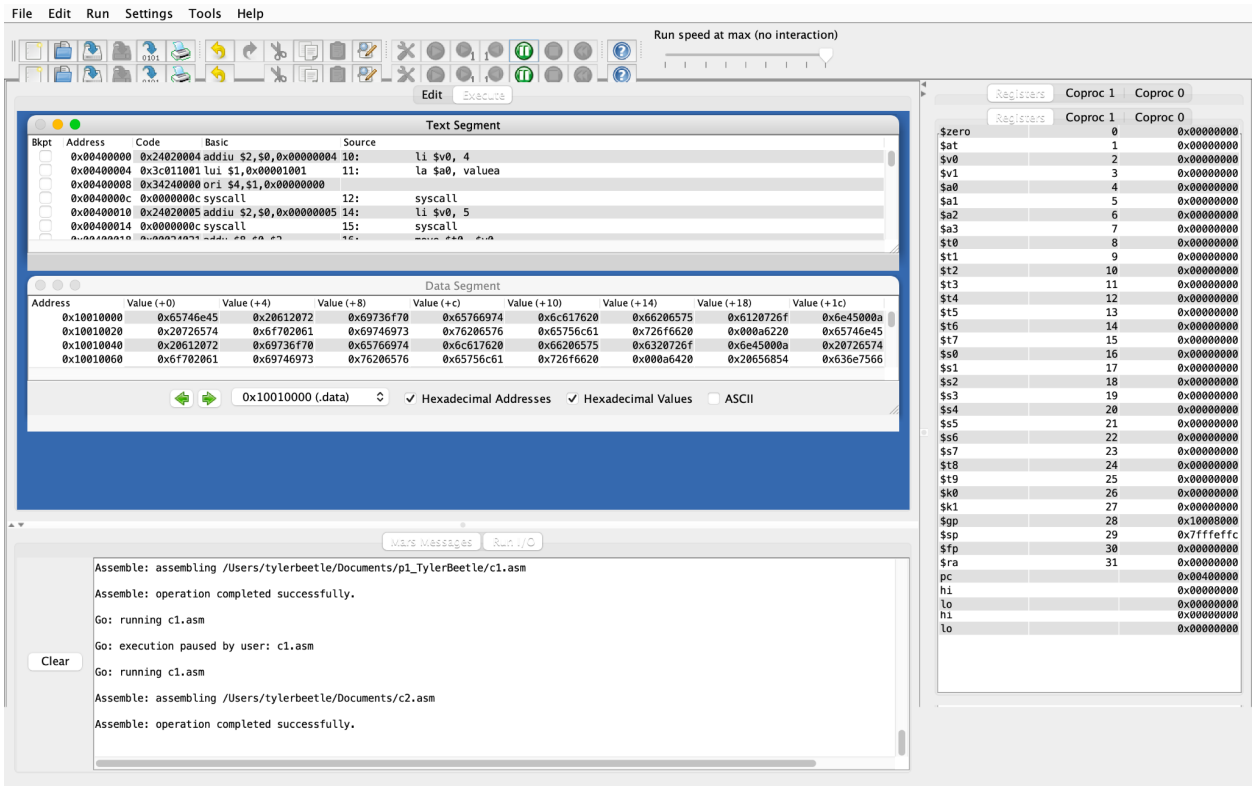
Registers	Purpose & Labels
$a0	-Stores the opening message in program 1 -Stores the inputted user name -stores the second message
$t0	-Stores the value of a in the second program
$t1	-Stores the value of b in the second program
$t2	-Stores the value of c in the second program
$t3	-Stores the value of d in the second program
$s0	-Stores the sum of a+b
$s1	-Stores the sum of c+d
$s2	-Stores the sum of b+3
$s3	-$s0 + $s2 - $s1
$v0	System Call
$a0	-Provides the output to the console

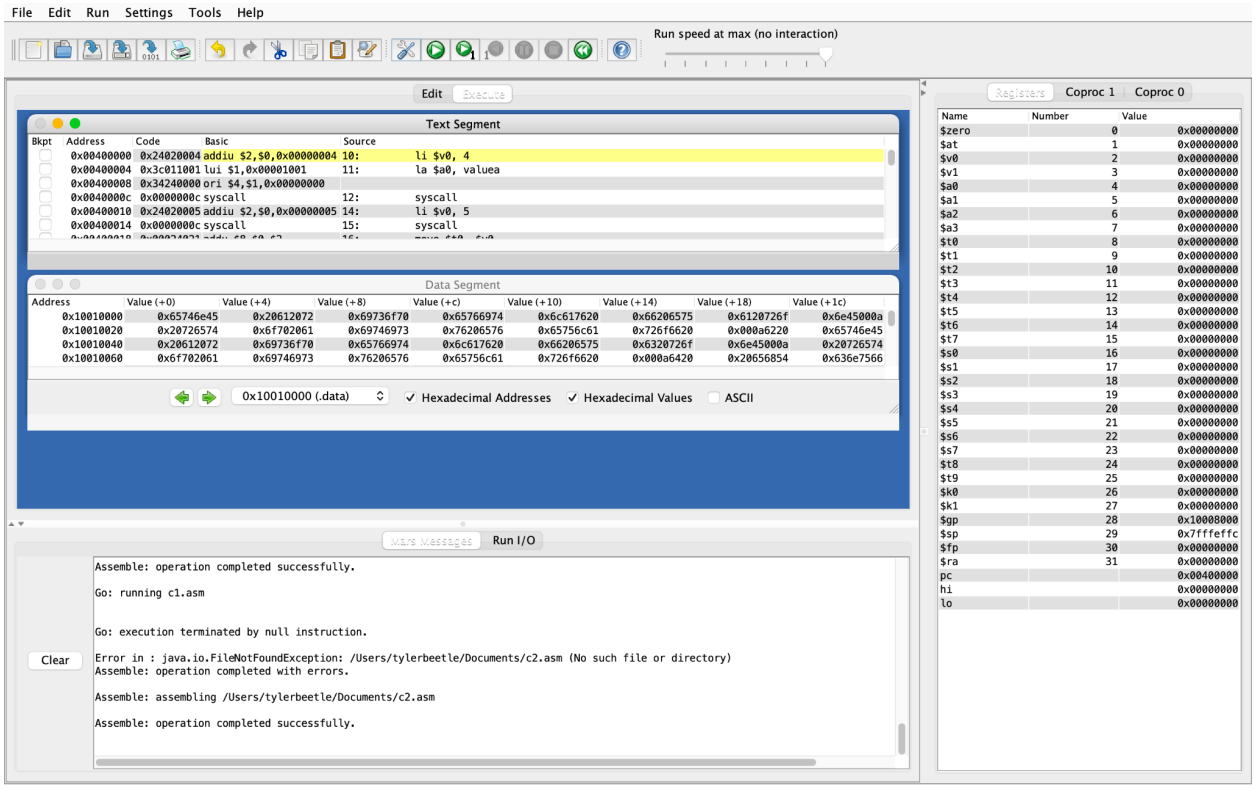
4.0 Learning Coverage

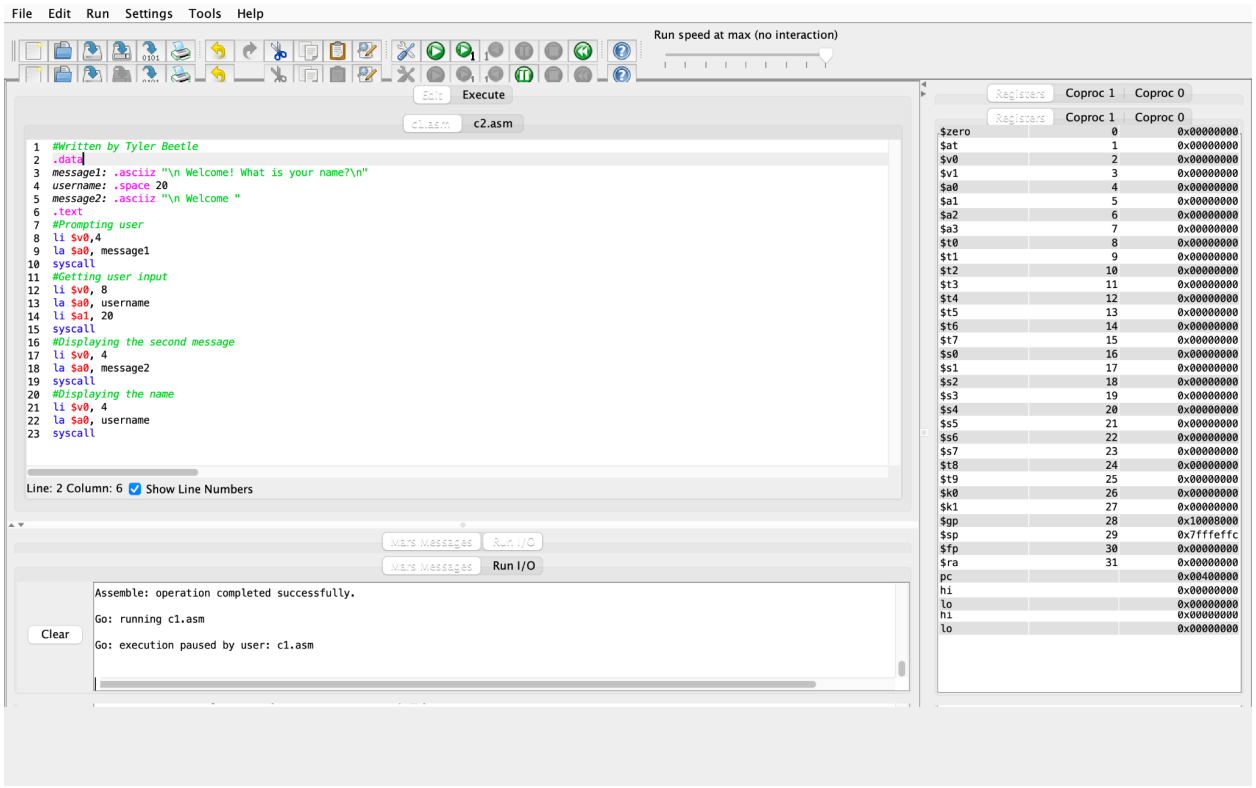
1. How to combine multiple registers with identical traits into one register.
2. How to reserve a register for later user input in the program.
3. The ability to organize code by generating concise comments that highlight your code
4. When a output comes from a constant established message in text and when an output is dynamic
5. Registers can be reused as you work further down in the program

5.0 Test Results




File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x24020004 addiu $2,$0,0x00000004 10: li $v0, 4
0x00400004 0x3c011001 lui $1,0x00001001 11: la $a0, valuea
0x00400008 0x34240000 ori $4,$1,0x00000000
0x0040000c 0x0000000c syscall 12: syscall
0x00400010 0x24020005 addiu $2,$0,0x00000005 14: li $v0, 5
0x00400014 0x0000000c syscall 15: syscall
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 0x65746e45 0x20612072 0x69736f70 0x65766974 0x6c617620 0x66206575 0x6120726f 0x6e45000a
0x10010020 0x20726574 0x6f702061 0x69746973 0x76206576 0x65756c61 0x726f6620 0x000a6220 0x65746e45
0x10010040 0x20612072 0x69736f70 0x65766974 0x6c617620 0x66206575 0x6320726f 0x6e45000a 0x20726574
0x10010060 0x6f702061 0x69746973 0x76206576 0x65756c61 0x726f6620 0x000a6420 0x20656854 0x636e7566
0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII
Mars Messages Run I/O
Assemble: assembling /Users/tylerbeetle/Documents/p1_TylerBeetle/c1.asm
Assemble: operation completed successfully.
Go: running c1.asm
Go: execution paused by user: c1.asm
Go: running c1.asm
Assemble: assembling /Users/tylerbeetle/Documents/c2.asm
Assemble: operation completed successfully.
Clear
Registers Coproc 1 Coproc 0
$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
hi 0x00000000
lo 0x00000000


File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x24020004 addiu $2,$0,0x00000004 10: li $v0, 4
0x00400004 0x3c011001 lui $1,0x00001001 11: la $a0, valuea
0x00400008 0x34240000 ori $4,$1,0x00000000
0x0040000c 0x0000000c syscall 12: syscall
0x00400010 0x24020005 addiu $2,$0,0x00000005 14: li $v0, 5
0x00400014 0x0000000c syscall 15: syscall
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 0x65746e45 0x20612072 0x69736f70 0x65766974 0x6c617620 0x66206575 0x6120726f 0x6e45000a
0x10010020 0x20726574 0x6f702061 0x69746973 0x76206576 0x65756c61 0x726f6620 0x000a6220 0x65746e45
0x10010040 0x20612072 0x69736f70 0x65766974 0x6c617620 0x66206575 0x6320726f 0x6e45000a 0x20726574
0x10010060 0x6f702061 0x69746973 0x76206576 0x65756c61 0x726f6620 0x000a6420 0x20656854 0x636e7566
0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII
Mars Messages Run I/O
Assemble: operation completed successfully.
Go: running c1.asm
Go: execution terminated by null instruction.
Error in : java.io.FileNotFoundException: /Users/tylerbeetle/Documents/c2.asm (No such file or directory)
Assemble: operation completed with errors.
Assemble: assembling /Users/tylerbeetle/Documents/c2.asm
Assemble: operation completed successfully.
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


File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
c1.asm c2.asm
1 #Written by Tyler Beetle
2 .data
3 message1: .asciiz "\n Welcome! What is your name?\n"
4 username: .space 20
5 message2: .asciiz "\n Welcome "
6 .text
7 #Prompting user
8 li $v0,4
9 la $a0, message1
10 syscall
11 #Getting user input
12 li $v0, 8
13 la $a0, username
14 li $a1, 20
15 syscall
16 #Displaying the second message
17 li $v0, 4
18 la $a0, message2
19 syscall
20 #Displaying the name
21 li $v0, 4
22 la $a0, username
23 syscall
Line: 2 Column: 6 Show Line Numbers
Mars Messages Run I/O
Clear
Assemble: operation completed successfully.
Go: running c1.asm
Go: execution paused by user: c1.asm
Registers Coproc 1 Coproc 0
$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
hi 0x00000000
lo 0x00000000