Lab 2 Report

COMP3350
Adam Biggs – 9/19/2024

The purpose of this lab was to translate functions from C to MIPS Assembly.
All code is original and based off of MIPS documentation

Task A:

Write a MIPS assembly program that prompts the user to enter a string less than 40 characters (no need to test with invalid inputs). The program should then print the entered string back to the console.

This task was very simple and can be broken down into variables, prompt, input, print.
First I establish my variables for the message string and the user response (which I cap at 40 char). Then I print my prompt and listen for input. Following the input i print the input string and end. I had no issues writing this code and did it quickly.

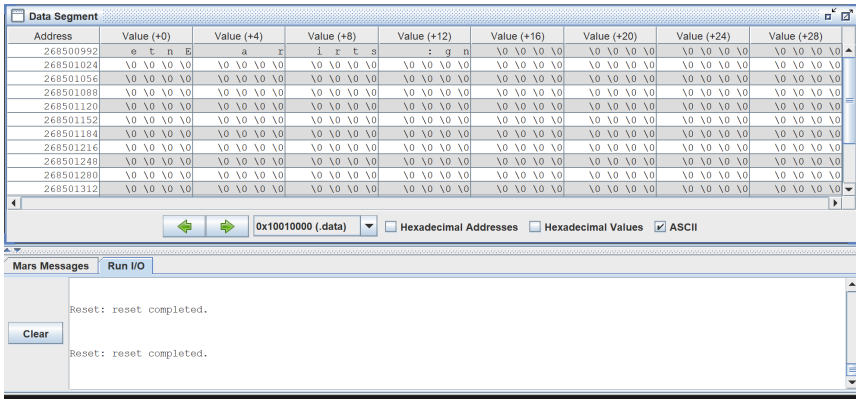
```
#Lab 2a - Adam Biggs ----------------------------------

.data
#variables
prompt:      .asciiz "Enter a string: " #prompt message
userString:  .space 40        #length of maximum response is 40

.text
.globl main
#main method ------------------------------------------
main:
  #print prompt
  li $v0, 4 #4 = print_string
  la $a0, prompt
  syscall

  #input detection
  li $v0, 8 #8 = read_string
  la $a0, userString
  li $a1, 40
  syscall

  #print input
  li $v0, 4 #4 = print_string
  la $a0, userString
  syscall

  #end
  li $v0, 10 #10 = exit
  Syscall
```

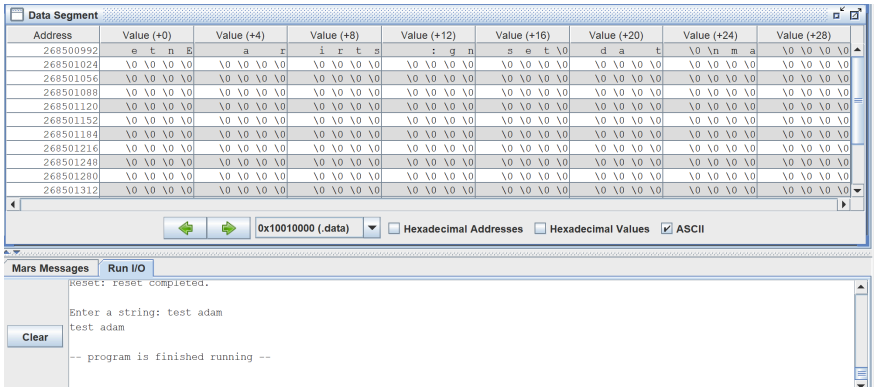
Before Run:

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	0
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194304
hi		0
lo		0

After Run:

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	10
$v1	3	0
$a0	4	268501009
$a1	5	40
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	0
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194364
hi		0
lo		0





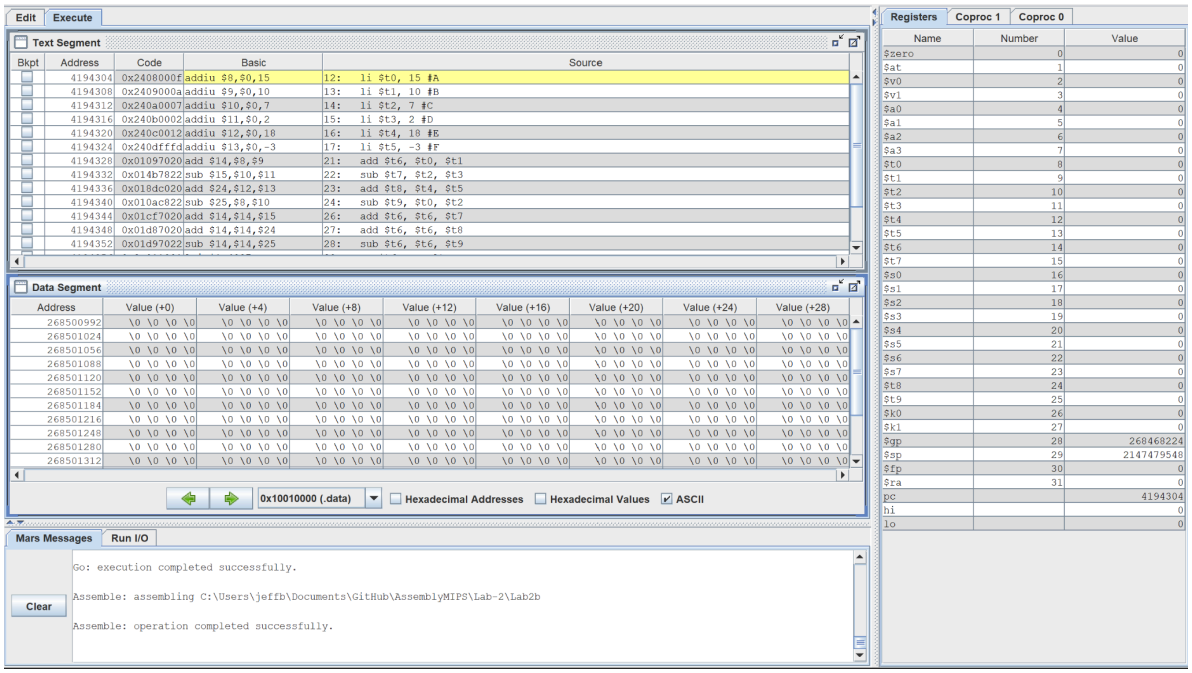
Task B:

Write a complete MIPS program that translates and calculates the equation in C shown below.

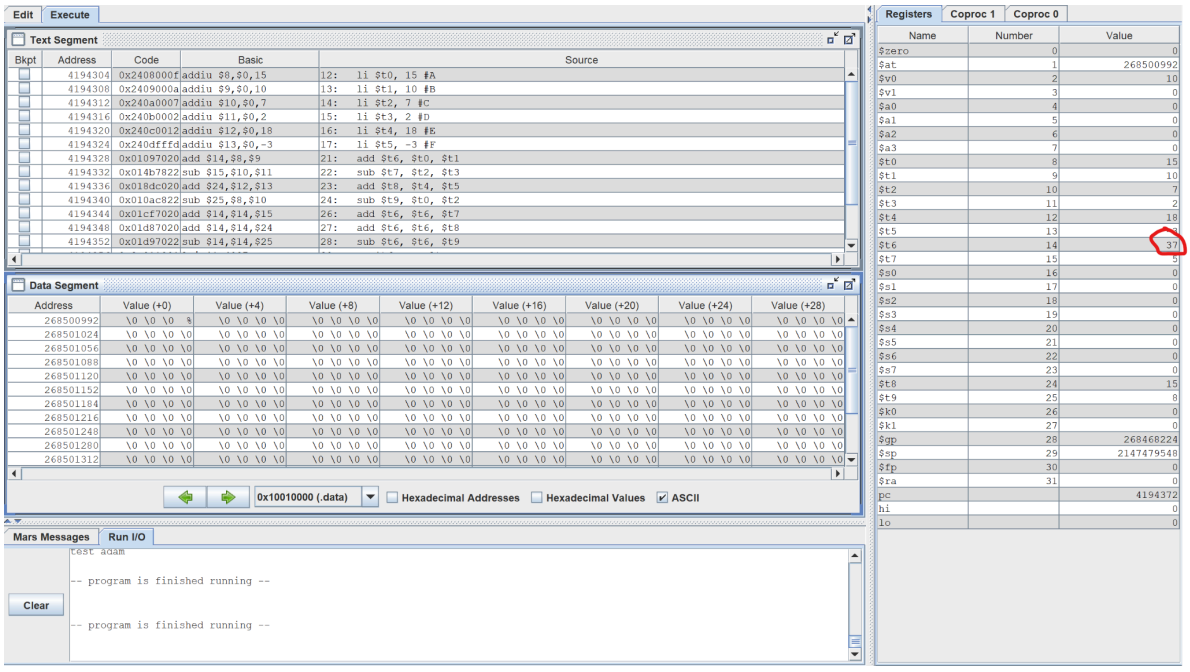
This task was also very simple. First I initialized my result Z to 0 (for use later). Then I set up each of my int vars. Then I perform the add and sub calculations. Finally I store the result in Z. This took me a few attempts, i kept mixing up the $t temp registers.

```
#Lab 2b - Adam Biggs ---------------------------------------

.data
#initialize var
resultZ: .word 0 #Z

.text
.globl main

main:
  #var setup
  li $t0, 15 #A
  li $t1, 10 #B
  li $t2, 7 #C
  li $t3, 2 #D
  li $t4, 18 #E
  li $t5, -3 #F

#basic operations
# Z = (A+B) + (C-D) + (E+F) - (A-C);
  add $t6, $t0, $t1
  sub $t7, $t2, $t3
  add $t8, $t4, $t5
  sub $t9, $t0, $t2

  add $t6, $t6, $t7
  add $t6, $t6, $t8
  sub $t6, $t6, $t9

  sw $t6, resultZ

  li $v0, 10
  Syscall
```

Before:



After:



Task C:

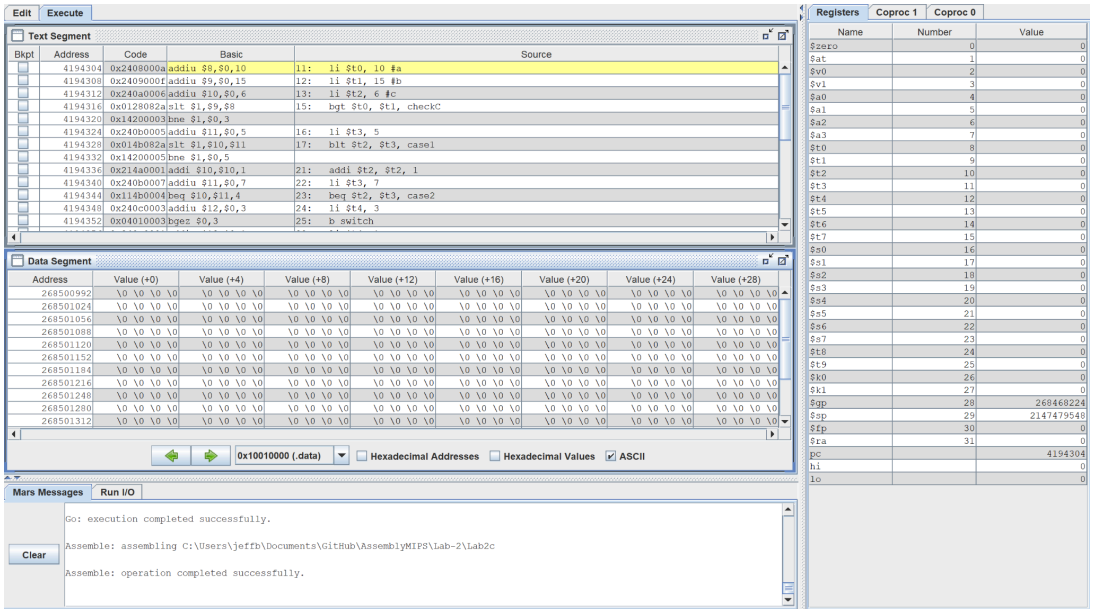
Write a complete MIPS program that implements the same algorithm (in C) shown below.

This task was a lot more time consuming. First I once again init the result var resultZ. Then I set up my 3 main vars. Then i start comparing the vars (If a>b || c<5) and cycling through the cases. For this one I had to reference documentation on how to do case switches. Turns out to be way easier than I initially thought, it's just very tedious in assembly. The biggest thing I learned from these 3 labs was how important syscall works and how setting $v0 effects the type of call.

```
#Lab 2C - Adam Biggs ----------------------------------

.data
#initialize var
resultZ: .word 0

.text
.globl main

main:
  li $t0, 10 #a
  li $t1, 15 #b
  li $t2, 6 #c

  bgt $t0, $t1, checkC
  li $t3, 5
  blt $t2, $t3, case1

#parameter
checkC:
  addi $t2, $t2, 1
  li $t3, 7
  beq $t2, $t3, case2
  li $t4, 3
  b switch

#case to check against 1
case1:
  li $t4, 1
  b switch

#case to check against 2
case2:
  li $t4, 2

switch:
  beq $t4, 1, set1
  beq $t4, 2, set2
  li $t4, 0
  b store

set1:
  li $t4, -1
```

```
  b store

set2:
  li $t4, -2

store: #store final result
  sw $t4, resultZ

  li $v0, 10
  Syscall
```

Before:



After: