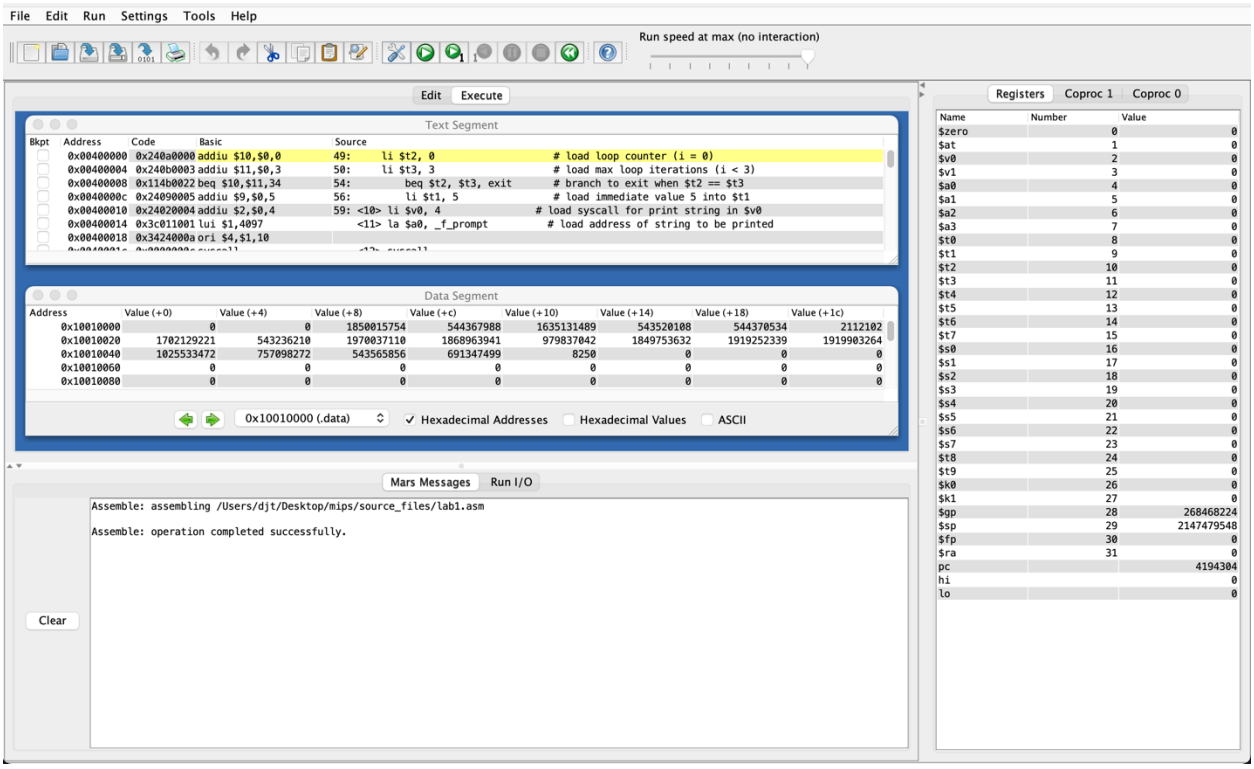
David J Tinley
09/24/2023
Lab 1 Report
Computer Organization

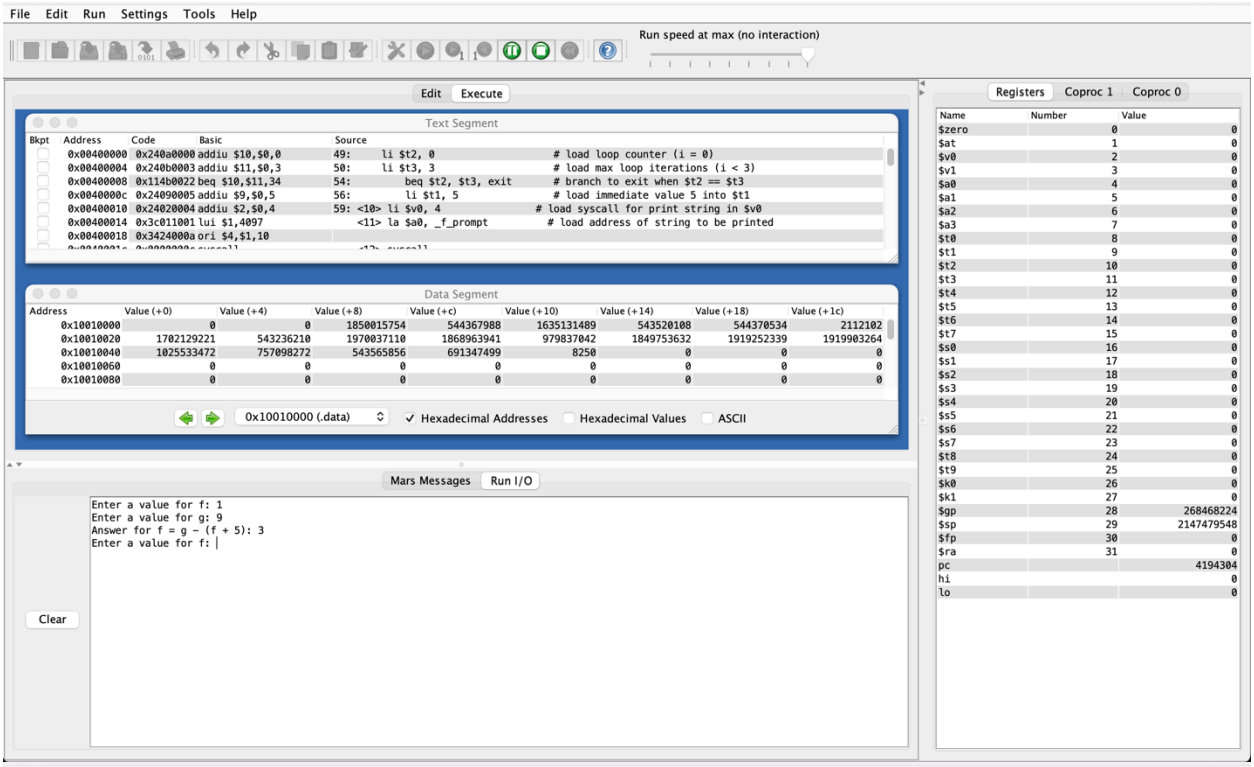
For this lab assignment I was able to utilize macro's that I had learned about online. They are essentially functions that you can write instead of having to write each line of assembly code over and over for things like print or read in the terminal. The second concept I used was a loop. The MIPS loops were like using for loops in C/C++. A big difference being you must declare your loop iterator and maximum loop counter outside of the loop. You also must explicitly tell the program where to branch to after the loop is finished.

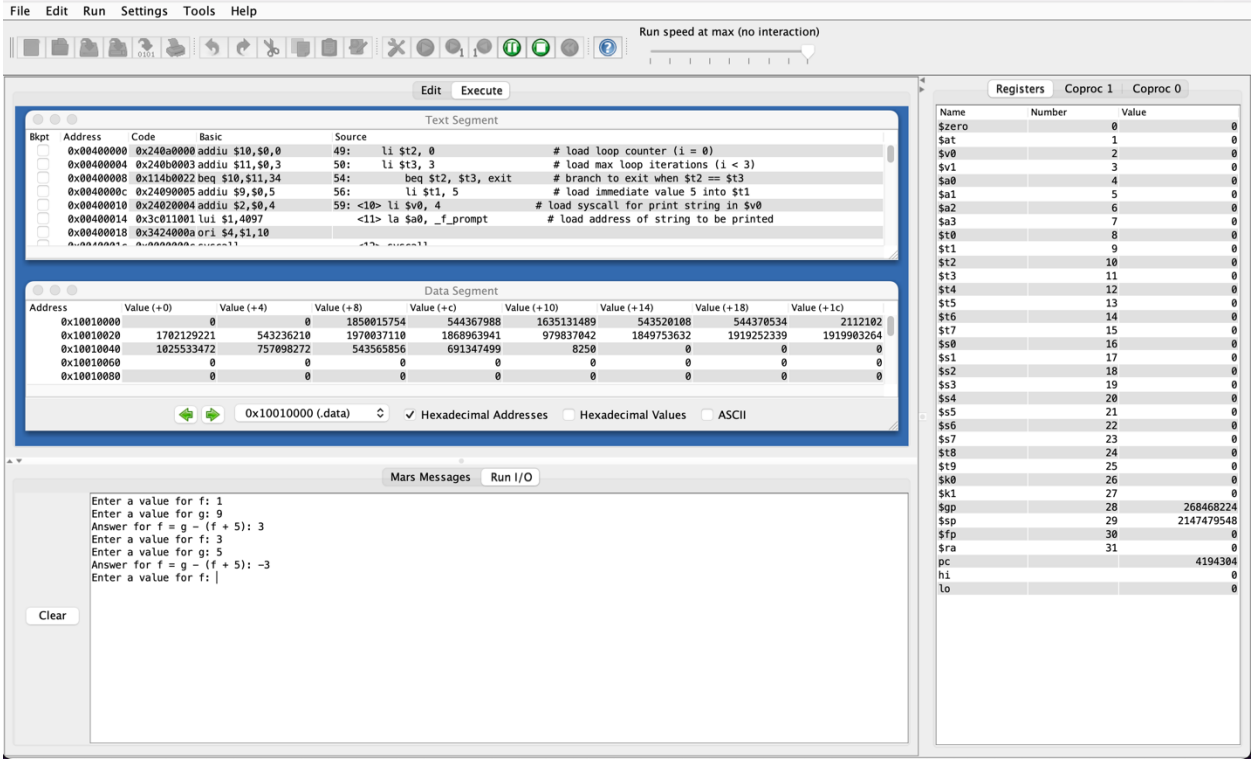
```
#
# Lab 1 - Computer Organization
# David J Tinley
# 09/22/2023
# Write a MIPS program to compute f = g - (f + 5)
#

# MACROS ###########################################################################
.macro print_string(%string) # macro for printing string parameter
   li $v0, 4              # load syscall for print string in $v0
   la $a0, %string        # load address of string to be printed
   syscall
.end_macro

.macro input(%num)         # macro for inputting integer
   li $v0, 5              # load syscall for reading integer
   la $a0, %num           # load address of integer to be input
   syscall                # value stored in $v0
.end_macro

.macro print_result(%num)   # macro for printing result
   li $v0, 1              # load syscall for printing integer
   la $a0, (%num)         # load address of integer to be printed
   syscall
.end_macro
###################################################################################


# DATA #############################################################################
.data
```

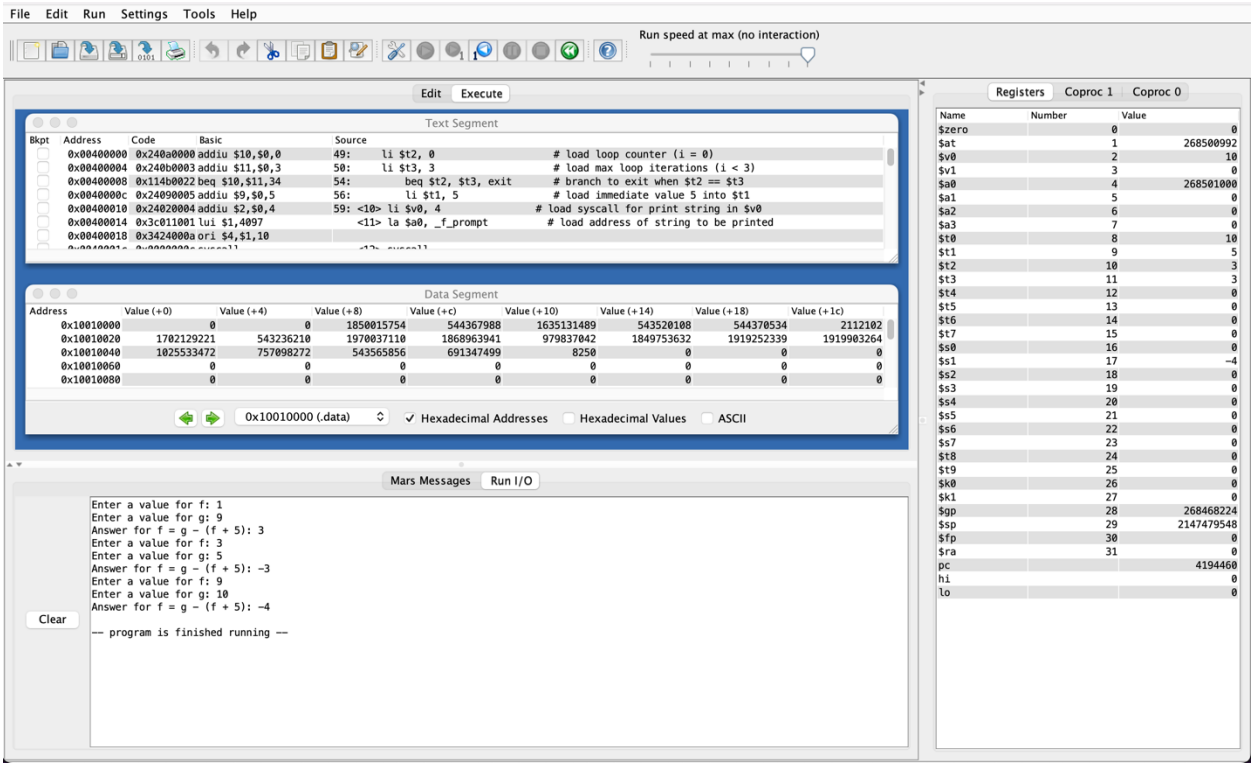
```
    .align 2            # align memory to 2^2, so 4 for word
                        # alignment must be declared before .word???
                        # declaration must also be literally aligned
                        # with .word???

_f: .word 0             # 32 bit integer for f
_g: .word 0             # 32 bit integer for g

_new_line: .asciiz "\n"
_f_prompt: .asciiz "Enter a value for f: "
_g_prompt: .asciiz "Enter a value for g: "
_answer:   .asciiz "Answer for f = g - (f + 5): "
################################################################################


# TEXT ########################################################################
.text

li $t2, 0               # load loop counter (i = 0)
li $t3, 3               # load max loop iterations (i < 3)

loop:                   # loop through 3 times total

    beq $t2, $t3, exit      # branch to exit when $t2 == $t3

    li $t1, 5           # load immediate value 5 into $t1
                        # used for equation (_f + 5)

    print_string(_f_prompt) # print prompt for _f input

    input(_f)           # input value for f variable

    la $s1, ($v0)       # store value of _f into $s1

    print_string(_g_prompt) # print prompt for _g input

    input(_g)           # input value for g

    la $t0, ($v0)       # store value of _g into $t0

    add $s1, $t1, $s1       # $s1 = (_f + 5)
```

```
    sub $s1, $t0, $s1       # $s1 = _g - _f

    print_string(_answer)   # print string "Answer for f = g - (f + 5):"

    print_result($s1)       # print result

    print_string(_new_line) # print new line character

    addi $t2, $t2, 1        # increment $t2 by 1 (++i)

    j loop                  # jump back to top of loop


# EXIT SYSCALL ##########################################################
exit:
    li  $v0, 10             # load syscall for program exit
    syscall

#########################################################################
```




File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x240a0000 addiu $10,$0,0 49: li $t2, 0 # load loop counter (i = 0)
0x00400004 0x240b0003 addiu $11,$0,3 50: li $t3, 3 # load max loop iterations (i < 3)
0x00400008 0x114b0022 beq $10,$11,34 54: beq $t2, $t3, exit # branch to exit when $t2 == $t3
0x0040000c 0x24090005 addiu $9,$0,5 56: li $t1, 5 # load immediate value 5 into $t1
0x00400010 0x24020004 addiu $2,$0,4 59: <10> li $v0, 4 # load syscall for print string in $v0
0x00400014 0x3c011001 lui $1,4097 <11> la $a0, _f_prompt # load address of string to be printed
0x00400018 0x3424000a ori $4,$1,10
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 0 0 1850015754 544367988 1635131489 543520108 544370534 2112102
0x10010020 1702129221 543236210 1970037110 1868963941 979837042 1849753632 1919252339 1919903264
0x10010040 1025533472 757098272 543565856 691347499 8250 0 0 0
0x10010060 0 0 0 0 0 0 0 0
0x10010080 0 0 0 0 0 0 0 0
0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII
Mars Messages Run I/O
Enter a value for f: 1
Enter a value for g: 9
Answer for f = g − (f + 5): 3
Enter a value for f:
Clear
Registers Coproc 1 Coproc 0
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


File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x240a0000 addiu $10,$0,0 49: li $t2, 0 # load loop counter (i = 0)
0x00400004 0x240b0003 addiu $11,$0,3 50: li $t3, 3 # load max loop iterations (i < 3)
0x00400008 0x114b0022 beq $10,$11,34 54: beq $t2, $t3, exit # branch to exit when $t2 == $t3
0x0040000c 0x24090005 addiu $9,$0,5 56: li $t1, 5 # load immediate value 5 into $t1
0x00400010 0x24020004 addiu $2,$0,4 59: <10> li $v0, 4 # load syscall for print string in $v0
0x00400014 0x3c011001 lui $1,4097 <11> la $a0, _f_prompt # load address of string to be printed
0x00400018 0x3424000a ori $4,$1,10
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 0 0 1850015754 544367988 1635131489 543520108 544370534 2112102
0x10010020 1702129221 543236210 1970037110 1868963941 979837042 1849753632 1919252339 1919903264
0x10010040 1025533472 757098272 543565856 691347499 8250 0 0 0
0x10010060 0 0 0 0 0 0 0 0
0x10010080 0 0 0 0 0 0 0 0
0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII
Mars Messages Run I/O
Enter a value for f: 1
Enter a value for g: 9
Answer for f = g − (f + 5): 3
Enter a value for f: 3
Enter a value for g: 5
Answer for f = g − (f + 5): -3
Enter a value for f:
Clear
Registers Coproc 1 Coproc 0
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




In conclusion, I think the project went well. The biggest issue I faced was understanding how memory alignment works within the code. I was eventually able to figure how the memory must be aligned differently for words, half-words, and so on. I do not fully understand why the alignment call had to be positioned exactly where I had put it in the code though.