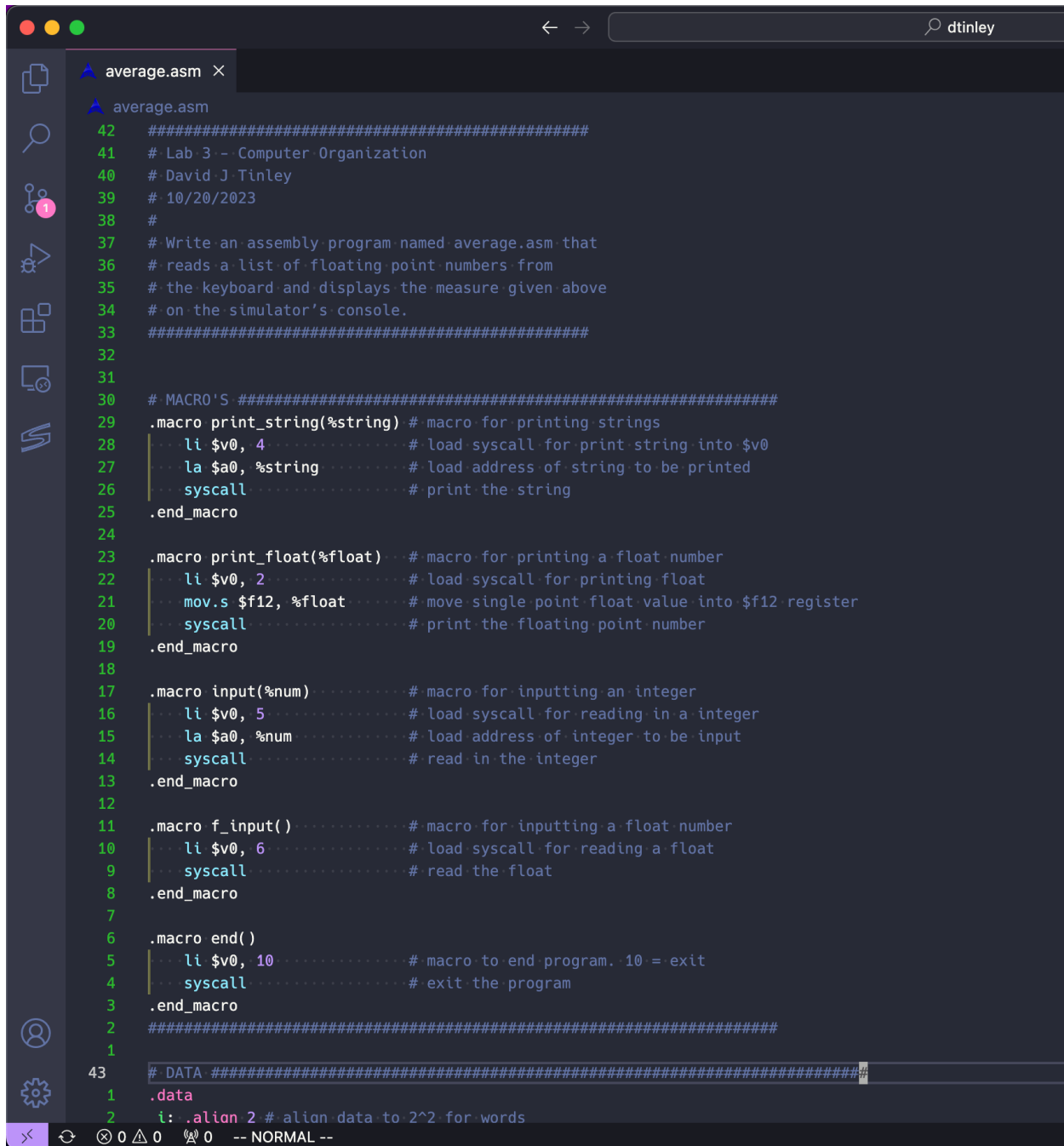


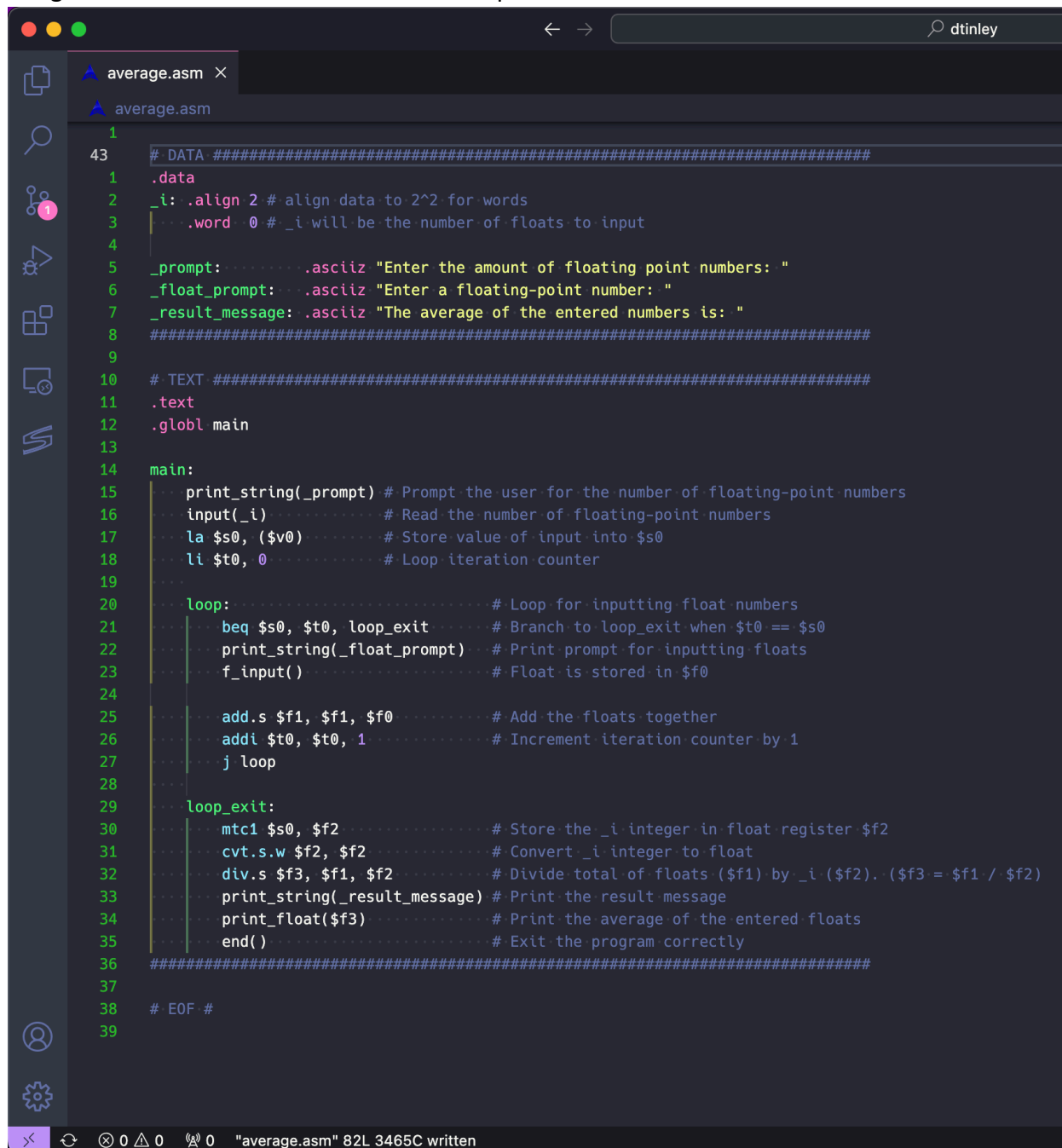
David J Tinley
10/22/2023
Lab 3 Report
Computer Organization

For lab 3 I again used the macros I have been implementing since lab1.



```
42 #####
41 # Lab 3 -- Computer Organization
40 # David J Tinley
39 # 10/20/2023
38 #
37 # Write an assembly program named average.asm that
36 # reads a list of floating point numbers from
35 # the keyboard and displays the measure given above
34 # on the simulator's console.
33 #####
32
31
30 # MACRO'S #####
29 .macro print_string(%string) # macro for printing strings
28     .li $v0, 4 # load syscall for print string into $v0
27     .la $a0, %string # load address of string to be printed
26     .syscall # print the string
25 .end_macro
24
23 .macro print_float(%float) # macro for printing a float number
22     .li $v0, 2 # load syscall for printing float
21     .mov.s $f12, %float # move single point float value into $f12 register
20     .syscall # print the floating point number
19 .end_macro
18
17 .macro input(%num) # macro for inputting an integer
16     .li $v0, 5 # load syscall for reading in a integer
15     .la $a0, %num # load address of integer to be input
14     .syscall # read in the integer
13 .end_macro
12
11 .macro f_input() # macro for inputting a float number
10     .li $v0, 6 # load syscall for reading a float
9     .syscall # read the float
8 .end_macro
7
6 .macro end()
5     .li $v0, 10 # macro to end program. 10 = exit
4     .syscall # exit the program
3 .end_macro
2 #####
1
43 # DATA #####
1 .data
2 .i: .align 2 # align data to 2^2 for words
```

Next, I declare the data to be used in the program. It was just three different strings and one integer value to be used as the number of inputs to use.



```
1
43 # DATA #####
1 .data
2 _i: .align 2 # align data to 2^2 for words
3     .word 0 # _i will be the number of floats to input
4
5 _prompt: .asciiz "Enter the amount of floating point numbers: "
6 _float_prompt: .asciiz "Enter a floating-point number: "
7 _result_message: .asciiz "The average of the entered numbers is: "
8 #####
9
10 # TEXT #####
11 .text
12 .globl main
13
14 main:
15     print_string(_prompt) # Prompt the user for the number of floating-point numbers
16     input(_i) # Read the number of floating-point numbers
17     la $s0, ($v0) # Store value of input into $s0
18     li $t0, 0 # Loop iteration counter
19
20     loop: # Loop for inputting float numbers
21         beq $s0, $t0, loop_exit # Branch to loop_exit when $t0 == $s0
22         print_string(_float_prompt) # Print prompt for inputting floats
23         f_input() # Float is stored in $f0
24
25         add.s $f1, $f1, $f0 # Add the floats together
26         addi $t0, $t0, 1 # Increment iteration counter by 1
27         j loop
28
29     loop_exit:
30         mtc1 $s0, $f2 # Store the _i integer in float register $f2
31         cvt.s.w $f2, $f2 # Convert _i integer to float
32         div.s $f3, $f1, $f2 # Divide total of floats ($f1) by _i ($f2). ($f3 = $f1 / $f2)
33         print_string(_result_message) # Print the result message
34         print_float($f3) # Print the average of the entered floats
35         end() # Exit the program correctly
36 #####
37
38 # EOF #
39
```

In the main section I first prompt the user for the number of floats they would like to enter and then have them enter those values. As the values are entered there are added together into a single register. Once the total numbers are input it jumps to the next section where the value for the total input numbers is converted into a float. The total of the floats is then divided by the converted number to get the average. The program then exits.

Users/djt/Desktop/dtinley/average.asm - MARS 4.5

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,\$0,4	58: <15> li \$v0, 4 # load syscall for print string into \$v0
	0x00400004	0x3c011001	lui \$1,4097	<16> la \$a0, _prompt # load address of string to be printed
	0x00400008	0x34240004	ori \$4,\$1,4	
	0x0040000c	0x0000000c	syscall	<17> syscall # print the string
	0x00400010	0x24020005	addiu \$2,\$0,5	59: <27> li \$v0, 5 # load syscall for reading in a integer
	0x00400014	0x3c011001	lui \$1,4097	<28> la \$a0, _i # load address of integer to be input
	0x00400018	0x34240000	ori \$4,\$1,0	
	0x0040001c	0x0000000c	syscall	<29> syscall # read in the integer
	0x00400020	0x20500000	addi \$16,\$2,0	60: la \$s0, (\$v0) # Store value of input into \$s0
	0x00400024	0x24000000	addiu \$8,\$0,0	61: li \$t0, 0 # Loop iteration counter
	0x00400028	0x12000009	beq \$16,\$8,9	64: beq \$s0, \$t0, loop_exit # Branch to loop_exit when \$t0 == \$s0
	0x0040002c	0x24020004	addiu \$2,\$0,4	65: <15> li \$v0, 4 # load syscall for print string into \$v0
	0x00400030	0x3c011001	lui \$1,4097	<16> la \$a0, _float_prompt # load address of string to be printed

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	1702129221	1752440946	1835081829	1953396079	543584032	1634692198	1735289204
0x10010004	1768910880	1847620718	1700949365	540701554	1953383680	1629516389	1869375008	1852404833
0x10010008	1869622631	544501353	1651340654	540701285	1701336064	1702256928	1701273970	543584032
0x1001000c	543516788	1702129253	543450482	1651340654	544436837	540701545	0	0
0x10010010	0	0	0	0	0	0	0	0
0x10010014	0	0	0	0	0	0	0	0
0x10010018	0	0	0	0	0	0	0	0
0x1001001c	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010024	0	0	0	0	0	0	0	0
0x10010028	0	0	0	0	0	0	0	0
0x1001002c	0	0	0	0	0	0	0	0
0x10010030	0	0	0	0	0	0	0	0

Registers Coproc 1 Coproc 0

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	268500992
\$v1	3	0
\$a0	4	268501073
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	2
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	2
\$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

Mars Messages Run I/O

Go: execution completed successfully.

Assemble: assembling /Users/djt/Desktop/dtinley/average.asm

Assemble: operation completed successfully.

Users/djt/Desktop/dtinley/average.asm - MARS 4.5

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,\$0,4	58: <15> li \$v0, 4 # load syscall for print string into \$v0
	0x00400004	0x3c011001	lui \$1,4097	<16> la \$a0, _prompt # load address of string to be printed
	0x00400008	0x34240004	ori \$4,\$1,4	
	0x0040000c	0x0000000c	syscall	<17> syscall # print the string
	0x00400010	0x24020005	addiu \$2,\$0,5	59: <27> li \$v0, 5 # load syscall for reading in a integer
	0x00400014	0x3c011001	lui \$1,4097	<28> la \$a0, _i # load address of integer to be input
	0x00400018	0x34240000	ori \$4,\$1,0	
	0x0040001c	0x0000000c	syscall	<29> syscall # read in the integer
	0x00400020	0x20500000	addi \$16,\$2,0	60: la \$s0, (\$v0) # Store value of input into \$s0
	0x00400024	0x24000000	addiu \$8,\$0,0	61: li \$t0, 0 # Loop iteration counter
	0x00400028	0x12000009	beq \$16,\$8,9	64: beq \$s0, \$t0, loop_exit # Branch to loop_exit when \$t0 == \$s0
	0x0040002c	0x24020004	addiu \$2,\$0,4	65: <15> li \$v0, 4 # load syscall for print string into \$v0
	0x00400030	0x3c011001	lui \$1,4097	<16> la \$a0, _float_prompt # load address of string to be printed

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	1702129221	1752440946	1835081829	1953396079	543584032	1634692198	1735289204
0x10010004	1768910880	1847620718	1700949365	540701554	1953383680	1629516389	1869375008	1852404833
0x10010008	1869622631	544501353	1651340654	540701285	1701336064	1702256928	1701273970	543584032
0x1001000c	543516788	1702129253	543450482	1651340654	544436837	540701545	0	0
0x10010010	0	0	0	0	0	0	0	0
0x10010014	0	0	0	0	0	0	0	0
0x10010018	0	0	0	0	0	0	0	0
0x1001001c	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010024	0	0	0	0	0	0	0	0
0x10010028	0	0	0	0	0	0	0	0
0x1001002c	0	0	0	0	0	0	0	0
0x10010030	0	0	0	0	0	0	0	0

Registers Coproc 1 Coproc 0

Name	Number	Value
\$zero	0	0
\$at	1	268500992
\$v0	2	10
\$v1	3	0
\$a0	4	268501073
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	2
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	2
\$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		4194432
hi		0
lo		0

Mars Messages Run I/O

Enter the amount of floating point numbers: 2

Enter a floating-point number: 1.2

Enter a floating-point number: 2.3

The average of the entered numbers is: 1.75

--- program is finished running ---

Overall, I did not experience any major difficulties. The main thing I learned was how mips differentiates between single point and double point floats and how they are stored in the co-processors.