

Nadine Nam
student ID: 811151197

The Code File:

- Nam - average.asm
- Nam - average.pdf

A brief Summary of Project Implementation

Regarding the project 3 implementation, this is learning how to store and calculate floating point numbers by using instructions in MIPS. An user enters numbers to average the sum, and they input decimal numbers according to the number of average values. Then divide the sum by floating points that are converted from integer.

Results showing the working code via screen prints

ENTIRE SCREEN

The screenshot displays a MIPS simulator interface with the following components:

- File Edit Run Settings Tools Help** menu bar.
- Run speed at max (no interaction)** status bar.
- Text Segment** window showing assembly code:

Bkpt	Address	Code	Basic	Source
	4194304	0x24090000	addiu \$9,\$0,0	25: li \$t1, 0 # counter i++
	4194308	0x3c011001	lui \$1,4097	36: la \$a0, myName
	4194312	0x34240000	ori \$4,\$1,0	
	4194316	0x24020004	addiu \$2,\$0,4	37: li \$v0, 4
	4194320	0x0000000c	syscall	38: syscall
	4194324	0x3c011001	lui \$1,4097	43: la \$a0, prompt1
	4194328	0x3424002c	ori \$4,\$1,44	
	4194332	0x24020004	addiu \$2,\$0,4	44: li \$v0, 4
	4194336	0x0000000c	syscall	45: syscall
	4194340	0x24020005	addiu \$2,\$0,5	48: li \$v0, 5 # read int
	4194344	0x0000000c	syscall	49: syscall # assign int
	4194348	0x00024021	addu \$8,\$0,\$2	50: move \$t0, \$v0 # move int number (n) in \$v0 to \$t0
- Data Segment** window showing memory values:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	1936287828	544434464	1768186190	1310745966	1881173345	1702061426	1852404846	1986076775
268501024	1734439525	1935748709	667245	544698184	2037277037	1836412448	1936876898	1970239264
268501056	2032165996	1814066543	543517545	1629515636	1634887030	977233255	32	1634036816
268501088	1696621939	1919251566	857760032	1734960160	1679848553	1835623269	1679846497	979657774
268501120	32	1953384714	1919248229	544175136	1634692166	1735289204	1768910880	540701806
268501152	1968376320	2112109	1750338058	1983979621	1734439525	1936269413	8250	0
268501184	0	0	0	0	0	0	0	0
268501216	0	0	0	0	0	0	0	0
268501248	0	0	0	0	0	0	0	0
- Registers** window showing register values:

Name	Number	Value
\$zero	0	0
\$at	1	268500992
\$v0	2	10
\$v1	3	0
\$a0	4	268501160
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	3
\$t1	9	3
\$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		4194500
hi		0
lo		0
- Mars Messages** window showing program output:

```
This is Nadine Nam presenting average.asm.  
How many numbers would you like to average?: 3  
Please enter a 3 digit decimal d.dd: 3.14  
Please enter a 3 digit decimal d.dd: 4.12  
Please enter a 3 digit decimal d.dd: 1.29  
  
Sum: 8.55  
Integer to Floating point: 3.0  
  
The Average is: 2.8500001  
--- program is finished running ---
```

RUN I/O



The Conclusion listing the lessons learned and problems faced

My project 3 has three procedures. They are main(), Loop(), and Return(). In the main(), it prints out prompt messages to input integer number.

For the Loop() procedure, it repeats looping to get decimal numbers from the user until the end of a for-statement. Once the counter i met the number of n, the process moves on to the next procedure, the Return().

For the Return(), it prints out the sum of decimals values, converts integer to the floating points, and calculates the average of decimals numbers.

Problems I faced

1. At the first time, I couldn't print out the float numbers, but learned that the float type contents should be moved to register \$f12 for the output.
2. The most complicated problem I faced in this project was converting the data type from integer to floating points.

Solution:

1.
 - A) To input the decimal numbers, the user should do system call 6 for reading a float.

```
# < Read - Input Decimal Numbers from User >
    li $v0, 6                # system call 6 for reading a float is
required in $v0
    syscall                  # the user's input is placed in register
$f0
```

1. • B) To print out the floating points, I need to move the stored contents of FP to the register \$f12

```
# < Print the number as floating point on the terminal >
mov.s $f12, $f4
li $v0, 2
syscall
```

2. To convert int to float in MIPS, I should use data movement and conversion instructions

```
# < CONVERT int to floating point >
mtc1 $t0, $f3          # move int stored in $t0 to floating point
register $f3
cvt.s.w $f4, $f3        # convert (from word to single precision)
$f3 to $f4
# cvt.s.w $f12, $f12 is also
acceptable
```

Lessons

In the project 3, I learned

- How to input output floating points
- How to convert integer to the floating points in MIPS
- Using MIPS floating point instructions
- Types of FP instructions, such as Arithmetic Instructions and Data movement/conversion instructions