MIPS Floating Point Addition

Raphael Spoerri

Introduction

This project consists of a simple MIPS program that adds two floating point numbers. For simplicity, I've used hard-coded floats and MARS' system call to print the floating-point sum, and I've omitted the stack frame. A more complex program would require implementing these.

Methods

This program uses hard-coded floating-point literals, statically initialized in the data section like this:

```
.data
float_anon1: .float 1
float_anon2: .float 2
```

The main program loads them into floating point registers, and then adds them into `$f12` (which is the same register that the system call uses for printing floats):

```
.text
main:
      lwc1 $f1, float_anon1
      lwc1 $f2, float_anon2
      add.s $f12, $f1, $f2
```

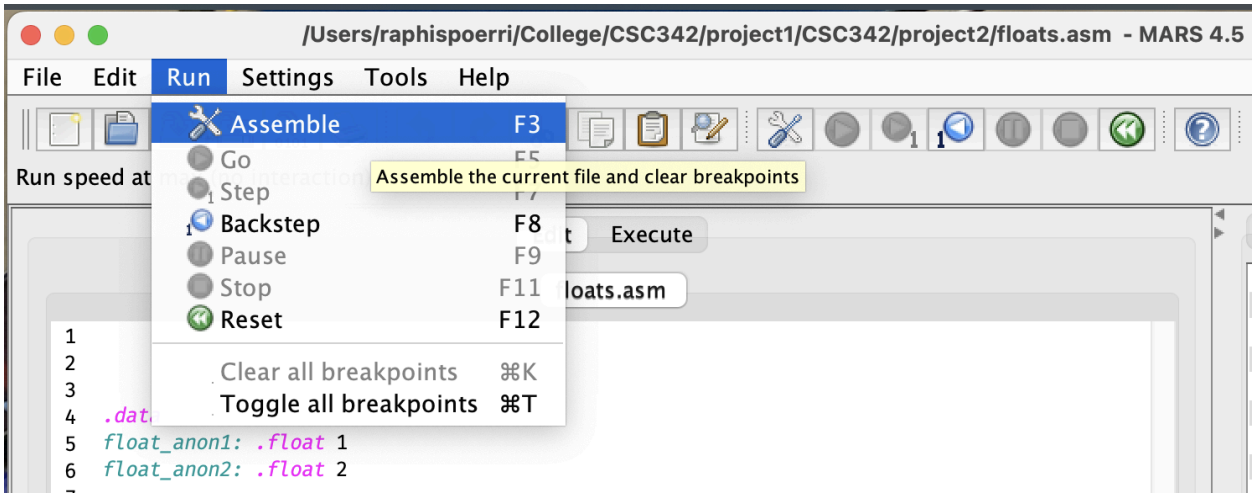
Finally, it performs a system call to print the float, and a system call to exit:

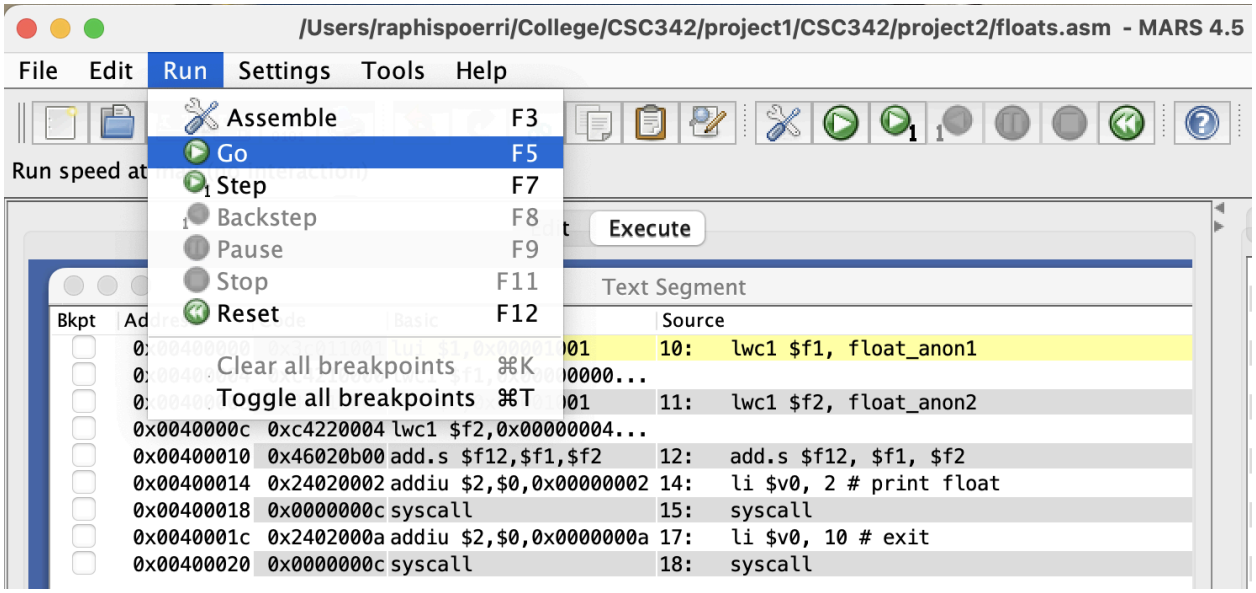
```
      li $v0, 2 # print float
      syscall

      li $v0, 10 # exit
      syscall
```

The full program can be found in the file `floats.asm` in this directory.

We can run it either by opening up the MARS editor, opening up `floats.asm`, pressing Run->Assemble, and then Run->Go (or via the icon buttons):





Or via the command line like this (Linux/Mac):

```
$ java -jar ~/Downloads/Mars*.jar floats.asm
```

(Replacing `~/Downloads/` with the directory where MARS is installed.)

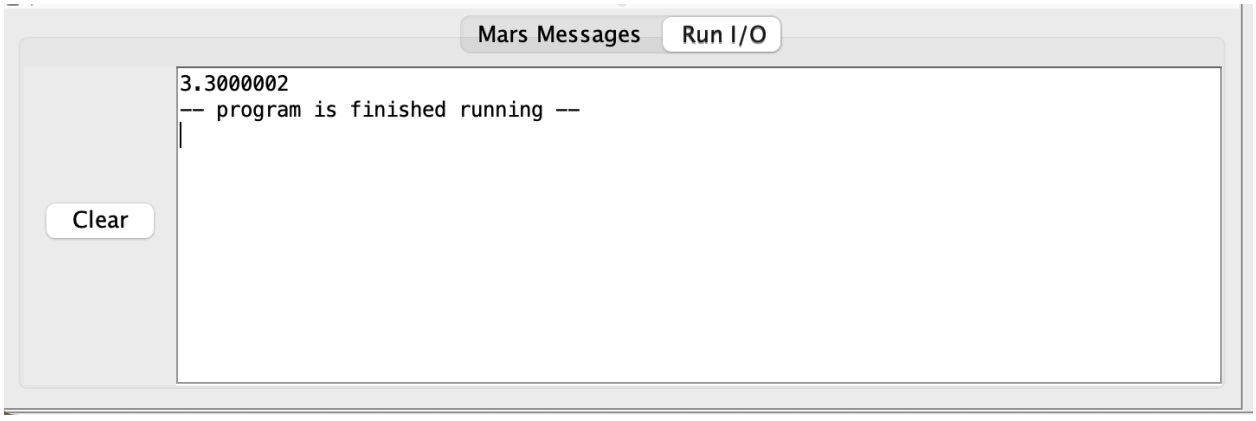
Results

The program successfully prints the sum of the two numbers:

It works for all inputs. As usual with floating point math, the results won't be exact, since binary floating point cannot represent all real decimal numbers exactly. For example, if we change the inputs:

```
float_anon1: .float 1.1
float_anon2: .float 2.2
```

We get:



but the output is nonetheless correct.

With more work, the program can be modified to read the inputs from the standard input or command-line arguments to make testing easier.

Conclusion

There are a lot of ways to improve this program, but for simplicity, I stuck to the basic requirements: The program adds two floating points and prints out their sum. The next step (if I remember correctly) involves timing the program, so it presents an opportunity for further improvement to this program.