Chris Carr

Manchester metropolitian university

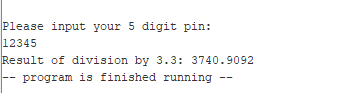
1cwk50 – part 3

# Floating Point Arithmetic in MIPS

In MIPS, floating point arithmetic is done in a separate component to the processor, called the co-processor. The arithmetic uses a standard for the arithmetic called IEEE 754 (latest revision IEEE 754-2008). This standard covers all the basis of arithmetic including: format, rounding rules, operations & exception handling.

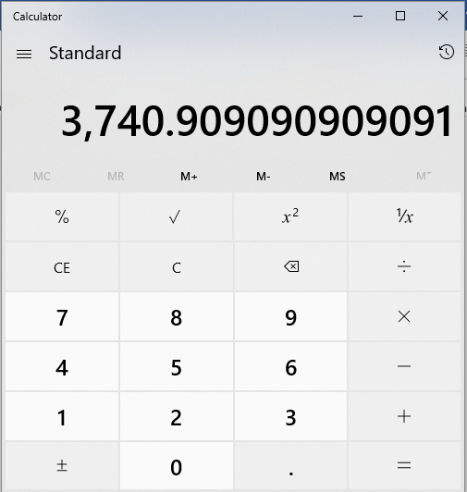
The co-processor includes a new set of registers, labelled from $f0-$f31. These registers come in pairs, as the second register is used to store the extra data for Doubles (covered later). Along with the new registers comes a new set of instructions to move values in and out of the co-processors’ registers, and instructions to perform arithmetic.

There are 2 data types accepted by the co-processor: Float and Double. Floats contain 32-bits of data, whereas doubles (conveniently named) contain double this – 64-bits of data. Doubles are extremely important for mathematical work where extreme precision is required.



***Figure A***

In my own work, I ran into a problem with using the 32-bit float data types. In Task C, I ran the division 12345/3.3, which is illustrated in *Figure A*. As you can see, the result of this is: 3740.9092. Next, to check that this was the correct result, I performed this addition on the windows calculator, which is illustrated in *Figure B*. The result of the division on the windows calculator was 3740.909090… . I noticed that this means there is a huge problem with the rounding of Single Precision Floats. The rounding of 90909 would never result in 9092; the rounding of 90909 should result in 9091 – in my program it becomes 9092. This is not an issue with the amount of data that is stored in Floats, rather it is an issue with the aforementioned IEEE 754 standards.



***Figure B***

(*Revision 1*) Cleaned up the comments. <https://github.com/jynxmagic/MIPS-Assembly-Language/commit/c5e8fd95381d9dcd8b74fa0cc75fab5fe701bb42>

# Flow Chart Diagram

Registers used: $a0, $v0, $a1

Instructions used: Load Address (la), Load Immediate (li), Syscall

Registers used: $f2, $f0, $f12

Instructions used: Load float 32 bit - single (l.s), Divide float 32 bit – single (div.s)

Output prompt to screen

User input pin

Divide by 3.3

Output Result

Registers used: $v0, $f12

Instructions used: Load Immediate (li), syscall

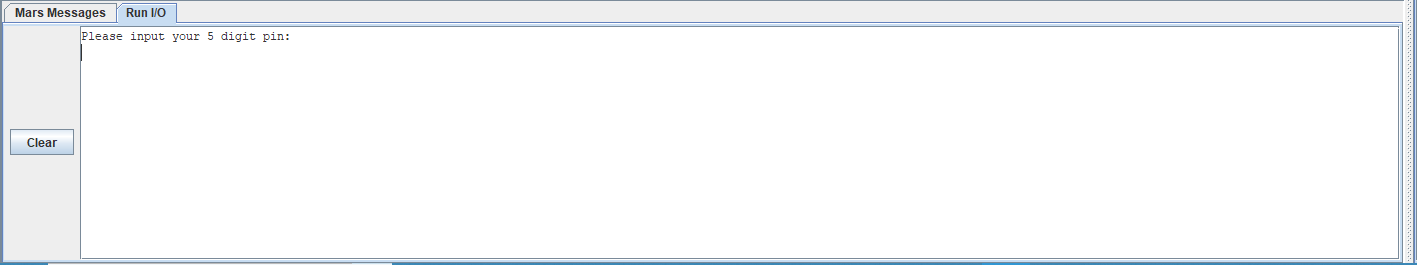
Registers used: $f12, $v0

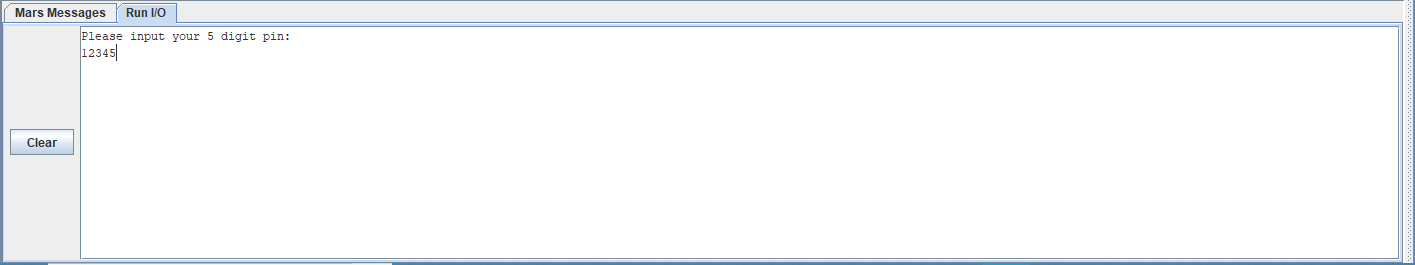
Instructions used: li, syscall

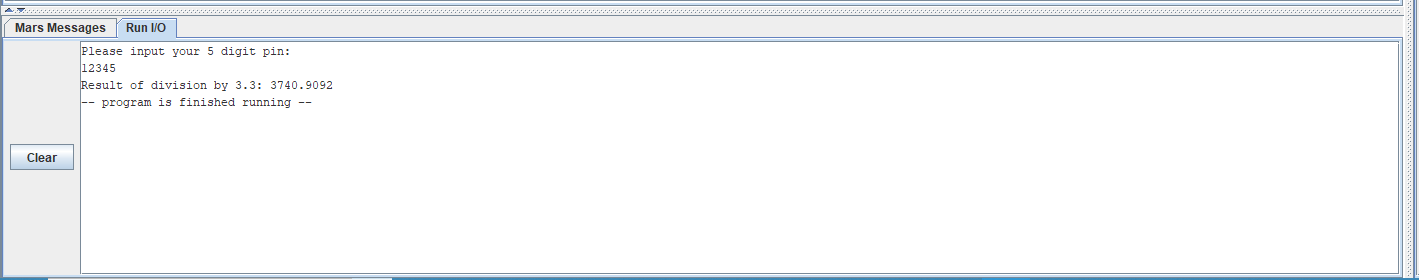
Registers used: $v0

Instructions used: $lli, syscall

# Program Screenshots







# Program Code

**.data**

**prompt:** **.asciiz** "Please input your 5 digit pin: \n"

**result:** **.asciiz** "Result of division by 3.3: "

**divisor:** **.float** 3.3

**.text**

## User input ##

#output text to screen

**la** $a0, **prompt**

**li** $v0, 4

**syscall**

#get user input

**li** $v0, 6

**syscall**

## Division ##

**l.s** **$f2**, **divisor**

#divide by 3.3 in co processor

**div.s** **$f12**, **$f0**, **$f2**

## Output ##

#output result string

**la** $a0, **result**

**li** $v0, 4

**syscall**

#output result

**li** $v0, 2

**syscall**

#end

**li** $v0, 10

**syscall**

Github commits: <https://github.com/jynxmagic/MIPS-Assembly-Language/commits/master/Task_C/Task_C.asm>