

# CMPT350: Laboratory 3

---

*Exploring Floating Point values by computing cube roots in assembly*

5%

Due: November 14, 2022 @ 17:00

Computing roots may seem like a daunting task, and yet, every two dollar calculator under the sun has at least a square root function, and computing cube roots is no more difficult. How does such limited hardware compute such values? Using Newton's approximation method, we can quickly compute roots in Assembly.

## Laboratory Procedure:

Newton's approximation is an iterative method that approaches the root value of any polynomial as the number of iterations increases (but we'll focus just on cube roots for this assignment). To compute the cube root, the recurrence relation

$$x_{n+1} = x_n - \frac{(x_n)^3 - y}{3(x_n)^2}$$

is used, where  $y$  is the value whose root we wish to find, and  $x_0$  is a user-provided rough guess for the root value. This procedure can be repeated any number of times for increasingly high resolution, but for this assignment we will halt the process when an adjacent pair of values ( $x_{n+1}$ ,  $x_n$ ) share at least 20 most significant mantissa bits. We will claim  $x_{n+1}$  is the cube root of  $y$ .

All floating point values should be taken as single precision (aka a 32 bit representation) for simplicity.

On the course Moodle page, look for the Laboratory 3 assignment in the Laboratories section. Inside is a premade skeleton MIPS file, `laboratory3.s`. Save this somewhere handy on your machine. You may use both the command line version, `spim`, and the GUI version, `QtSpim` to run your program. As well, you will need a text editor to edit your programs.

## Assignment Requirements:

- The program should prompt the user for two integer value:  $y$ , the value whose cube root we will calculate, and  $x_0$ , the initial guess of the cube root of  $y$ .
- The program calls a function using  $x_0$  and  $y$  to compute the cube root of  $y$ .
- The function will iteratively compute successive  $x$  values as defined by the above recurrence relation until  $x_{n+1}$  and  $x_n$  differ by at most 3 mantissa bits.
- Since the  $x_i$  values will become floating point values quite quickly, arithmetic computing these values should be done in the floating point coprocessor (c1, as its referred to by MIPS).
- Since moving and converting a floating point value into a main processor register will be more work than it's worth, the function should return the resulting  $x_n$  value in the \$f31 floating point register.
- In the main function, after the cube root function has been called, the program should print out the cube root value. This should not be done in the cube root function itself in your final submission.

\*\*\* NOTE: Your code **must properly interpret and execute** within the **laboratory environment**. Failure to do so will result in a **mark of ZERO** for the program. \*\*\*

## Creating a trace file

As part of all demo and laboratory submissions, you must submit what's called a trace file. This is a text file you'll create that in essence replicates what your terminal looks like when you demonstrate your programs. To assist with this, a tool called `script2` has been provided for you. To create a trace file, simply run `script2` in your terminal, and then proceed to use the terminal as normal to navigate to and run your programs. When you have demonstrated everything, press CTRL+D to signal that you are done, and `script2` will exit, creating a file called `typescript` in the directory you ran `script2` from. This is your trace file. **Please rename your tracefile to include your name, e.g. baker\_typescript**

## Submission:

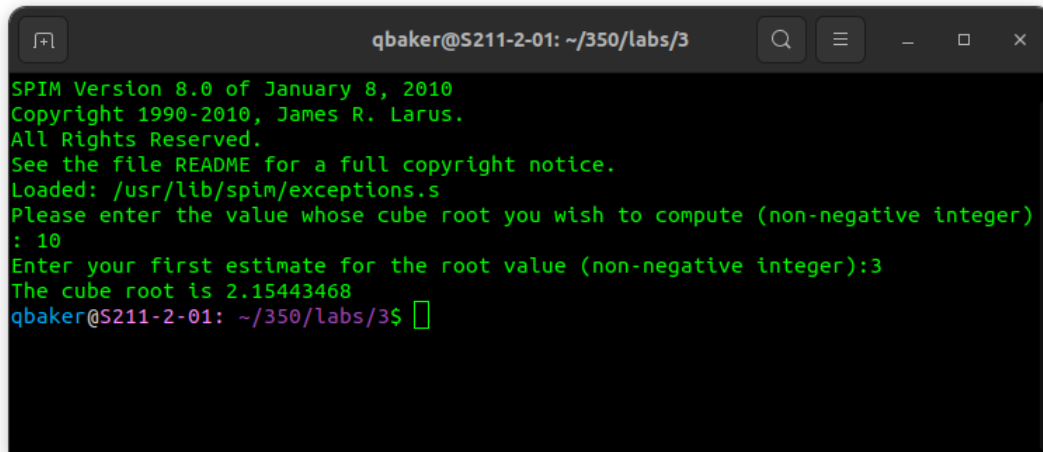
Your programs should be named using your name and end in a `.s` extension. For example, `BakerLaboratory2.s`

When you feel your programs meet all of the above requirements, create a zip archive with your code submission and trace file(s) included. An easy way to do this is with the `zip` command. For example, if your solution file and trace file are in a folder called `baker`, then the command `zip -r baker.zip baker/` will create a zip file containing your submission files called `baker.zip` (Replace this with your name when creating your zip file). Upload this zip file to Moodle.

### Hints:

- Use a while loop to repeatedly compute the root approximation values using the formula given on the first page.
- Recall Demo 5 and Demo 6's bitwise manipulation and the structure of the floating point when thinking about how to check the similarity of  $x_{n+1}$  and  $x_n$ .

### Example Output:



```
qbaker@S211-2-01: ~/350/labs/3
SPIM Version 8.0 of January 8, 2010
Copyright 1990-2010, James R. Larus.
All Rights Reserved.
See the file README for a full copyright notice.
Loaded: /usr/lib/spim/exceptions.s
Please enter the value whose cube root you wish to compute (non-negative integer)
: 10
Enter your first estimate for the root value (non-negative integer):3
The cube root is 2.15443468
qbaker@S211-2-01: ~/350/labs/3$
```