CMSC 411 Project Documentation

Group members: Justin Do, Rahsaan Hall, Benjamin Nace, Alexander Rochford

Our project implements the CORDIC algorithm for cosh(t), sinh(t), and e^t in ARM assembly language. The CORDIC algorithm involves a series of rotations of increasingly smaller pre-determined angles, in order to approximate a target angle. In the hyperbolic implementation, those angles are arctanh(0.5^i), where i is the rotation number which goes from 1 to the maximum number of rotations. For each rotation, new x and y values are calculated using the formulas:

$$x' = x + d * y/2^i$$

$$y' = y + d * x/2^i$$

In the formulas, d indicates the direction of rotation, 1 for clockwise and -1 for counterclockwise. For the hyperbolic implementation of the CORDIC algorithm, the y value starts at 0 and the x value starts at 1.207534. Ordinarily, the x value would start at 1, but the CORDIC algorithm requires the x and y values to be multiplied by a k value at the end, which is approximately 1.207534. To avoid this multiplication, we factor it in at the beginning in the initial x and y values.

When all of the rotations are done, x is approximately the value of cosh(t) and y is approximately the value of sinh(t). The value for e^t can be found by adding cosh(t) and sinh(t), so the resulting x and y are added together to get e^t.

The following are some statistics about our program, calculated for different clock speeds:

Statistic	**32kHz clock**	**1MHz clock**	**1GHz clock**
CPI	$\frac{32{,}000 \text{ cycles/sec}}{6135.2 \text{ instructions/sec}} = \mathbf{5.215\ CPI}$	$\frac{1{,}000{,}000 \text{ cycles/sec}}{6135.2 \text{ instructions/sec}} = \mathbf{162.99\ CPI}$	$\frac{1{,}000{,}000{,}000 \text{ cycles/sec}}{6135.2 \text{ instructions/sec}} = \mathbf{162{,}993.87\ CPI}$
Total computer cycles	$5.215 \frac{\text{cycles}}{\text{instruction}} \times 574 \text{ instructions} = \mathbf{2{,}993\ cycles}$	$162.99 \frac{\text{cycles}}{\text{instruction}} \times 574 \text{ instructions} = \mathbf{93{,}556\ cycles}$	$162{,}993.87 \frac{\text{cycles}}{\text{instruction}} \times 574 \text{ instructions} = \mathbf{93{,}558{,}481\ cycles}$
Total processing time	$2{,}993 \text{ cycles} \times \frac{1 \text{ second}}{32{,}000 \text{ cycles}} = \mathbf{0.0935\ seconds}$	$93{,}556 \text{ cycles} \times \frac{1 \text{ second}}{1{,}000{,}00 \text{ cycles}} = \mathbf{0.0936\ seconds}$	$93{,}558{,}481 \text{ cycles} \times \frac{1 \text{ second}}{1{,}000{,}000{,}000 \text{ cycles}} = \mathbf{0.0936\ seconds}$

Notes:

5 runs of the program were done on ARMSim, which gave instructions per second numbers for each run.

The 5-run average is: $\frac{6808+5696+5697+5696+6779}{5} = 6135.2 \frac{\text{instructions}}{\text{second}}$

According to ARMSim there are 574 instructions executed when the program is run.

Sample test data:

x	**cosh(x)**	**sinh(x)**	**e^x**
0	1.003	0.04332	1.046
0.88	1.416	1	2.416
π/4	1.327	0.870	2.198
- π/4	1.327	-0.8704	0.4568
-1	1.546	-1.178	0.3688

Project Division:

- Rahsaan Hall worked on the assembly code
- Benjamin Nace worked on the documentation
- Alexander Rochford and Justin Do worked on the presentation

Project Issues

- The hardest part about this project was completing the Assembly code because the algorithm was only documented well for sin and cos online. There was very little information about how to change the Cordic algorithm to compute cosh, sinh, and e^x.
- The easiest part of the project was completing the documentation because we had all of the necessary information and completed code to complete the documentation.

Project Conclusions

The Cordic algorithm is a very efficient algorithm to compute various elementary functions in a very short amount of time. It takes a very small amount of cycles and power to compute these functions and can be used in low powered computers.

Resources

C Code to reference when computing sinh, cosh, and e^x:

http://www.voidware.com/cordic.htm