Lab 1 Extra Credit

Daniel Diamont UT EID: dd28977

Robert Noe UT EID:

Observations (code was run on the simulator)

Version 1: C floating point

Average execution time over 10 trials: 0.01073189 s

Version 2: C fixed point

Average execution time over 10 trials: 0.00103049 s

Version 3: Assembly floating point

Average execution time over 10 trials: 0.00072358 s

Version 4: Assembly fixed point

Average execution time over 10 trials: 0.00077489 s

Notes:

The difference in execution time between the C floating point and C fixed point can probably be attributed to the fact that integer division is faster because it can be performed inside the CPU with the ALU and the CPU registers. In contrast, floating point is slower because the Floating-Point Unit does not have direct access to the CPU registers, thus data must be transferred through the peripheral bus. The difference between C code and assembly code can likely be attributed to weak compiler optimization settings in comparison to well written assembly code that takes less bus cycles to execute the calculation. Lastly the difference seen between the ASM floating-point and ASM fixed point goes against our original intuition. It makes sense that the FPU must load data from the CPU because it cannot directly access CPU registers. Additionally, data for the calculations must be pushed and pulled from the stack. In total, it makes sense that the floating-point calculation in assembly will take longer than the fixed-point version. For fixed-point, the processor can handle the operations with shift instructions in the CPU that can take 2 bus cycles or less because of hardware dividers as well as pipelined operations. Perhaps, the discrepancy in our results between Version 3 and Version 4 can be attributed to the execution of instructions in the Keil Simulator.