ECE 486
Winter 2022
Hector Soto
3/14/2022

RISC-V Simulator Final Project Report

Introduction and Simulation Limits

A RISC-V simulator was created to simulate programs written in RISC-V assembly. This simulator can allocate up to 4-GiB of memory that instructions can write to. Up to 4-GiB of instructions can be loaded into this simulator. Only the RV32I base instruction set was implemented, excluding the FENCE, ECALL, and EBREAK instructions.

Extra Credit + Extra Features

GUI: A Graphical User Interface was implemented. You can see the ENTIRE state of the simulation. Your view at one time is of course limited but you could theoretically shift your view of memory to see all 4 GiB of memory between instruction steps.

Time-Stepping: You can step through instructions within the GUI

Time-Reversal: When you step forward in time, the previous state is saved so you can reverse the simulation. Reversing to a previous state also undoes any changes made to the 4-GiB of memory. This is possible because each saved state only needs to hold the status of the registers, and the data stored in a memory location before a write occurred if a write occurred.

Instruction Protection: By default instructions are loaded into the same memory that they can alter. So they have the capability of overwriting themselves. You can enable instruction protection, which will load instructions into their own separate simulated memory that they can't alter.

Simulation Trace File: You can redirect normal simulation output from the terminal into a trace file.

Memory Trace File/s: You can output all of accessed memory into trace files. By default the program uses 16 threads to write memory in 16 different trace files. You can only change the amount of threads used (and thus amount of separate files generated) before compilation time by altering a DEFINE.

Linux + Windows Compilation: Can compile and run simulator on both Linux and Windows.

Simulation Memory Quick Rundown

Memory is allocated when needed. The smallest amount of memory that can be allocated is 1KiB which was defined as a page element. Memory at its highest level is divided into 32 memory partitions.

Memory (4 GiB Max): An array of 32 Memory Partitions.
Memory Partition (128 MiB Max): An array of 64 Memory Partition Elements.
Memory Partition Element (2 MiB Max): An array of 64 Pages.
Page (32 KiB Max): An array of 32 Page Elements.
Page Element (1 KiB Total): An array of 256 4-byte elements (32-bit integers/instructions/data).

If all 4 GiB of memory is allocated with 64-bit pointers, an extra 32 MiBs would be used for pointers. If instruction protection is on, it's possible for the program to allocate more than 8 GiBs of memory.