CISC360 - Project 3

James Feister - jfeister@udel.edu

Introduction

Project 3 required we expand even further upon our current Gemini implementations.

Part 1: A pipelining architecture within the CPU

- Consists of [Fetch|Decode|eXecute|Store] stages.
- Separate each stage into its own thread

Part 2: Branch prediction;

- Based on past results predict future branching within fetch stage

Pipelining

The pipelining is separated into 4 running threads. In order to maintain simple cross platform support, making sure not having to mess with compiler flags, I went with the implementation of QThreads provided by the Qt library. Messages to update the GUI utilized the easy to use Qt provided with a SLOT and SIGNAL interface. The hardest part of this whole thing was getting the synchronization between the threads to happen correctly.

Branch prediction

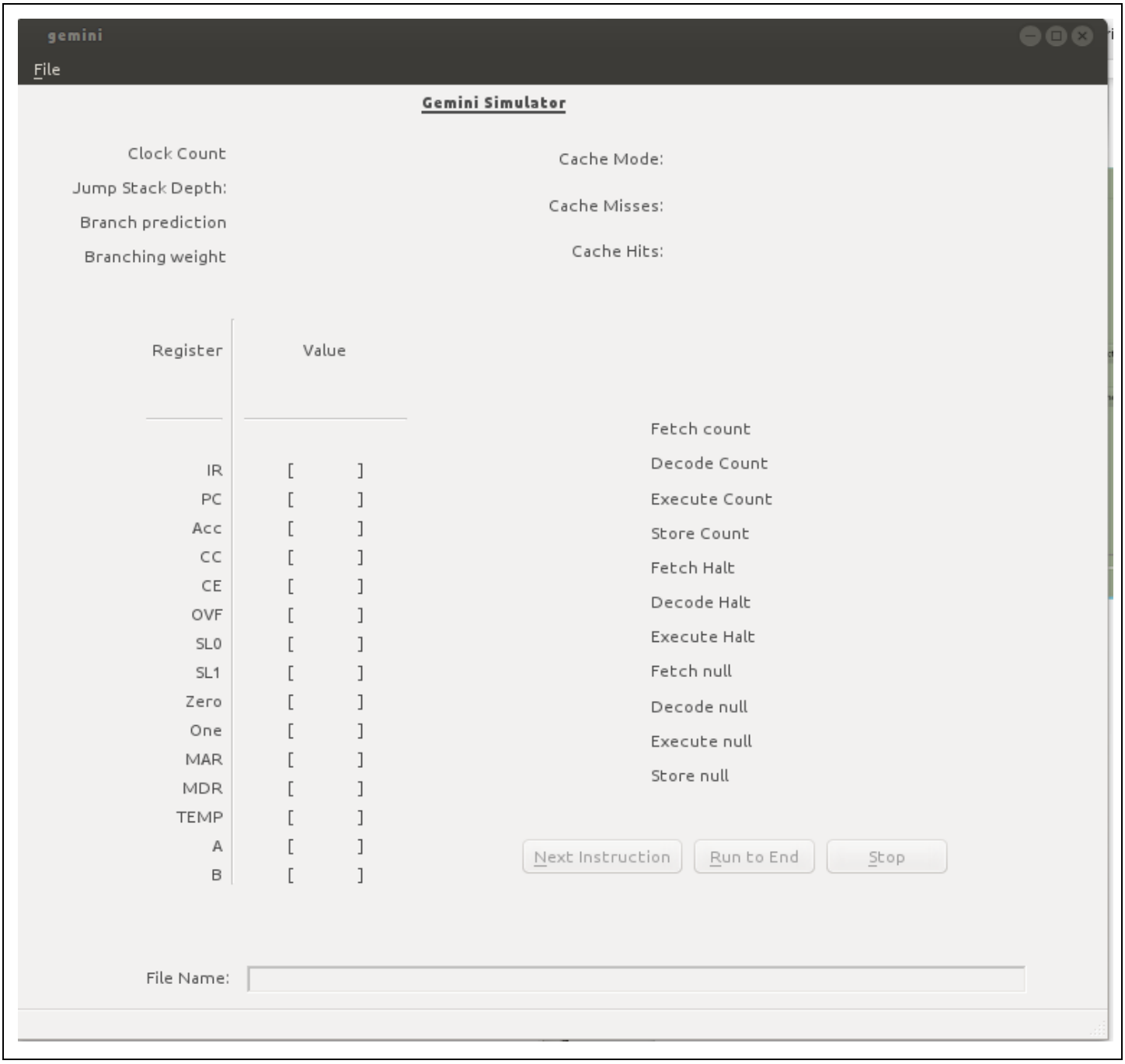
As an enhancement I performed the branch prediction. I do not have any immediate comments on the results of the implementation. The count on the **branch_ways** var goes +1 for successful branch and -1 for unsuccessful prediction. The prediction portion will then evaluate if **branch_ways** is less then 0, which it will then use a path taken, or greater then 1 which it will then keep the next instruction.

GUI Enhancements

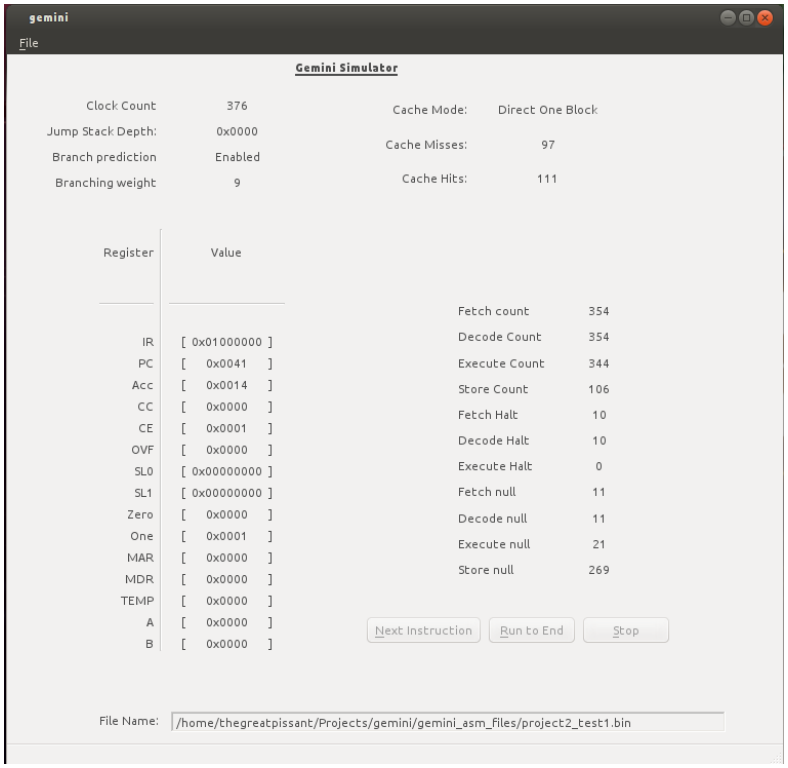
The '**Run to End'** button is now tied to a QTimer object that will emit SIGNALS to the gemini gui for executing the next instruction button. This allows for a stop button as well that will stop execution of the timer and allow for the user to restart at any time.

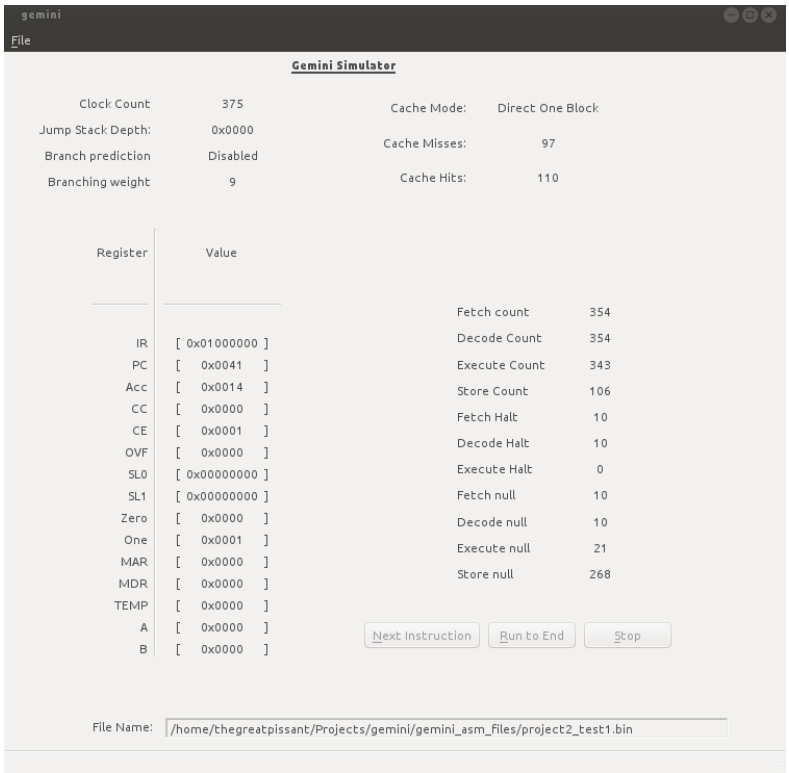
Counts have been added to show each stage execution count, halt count, and null (nop) count.

Gemini simulator update GUI

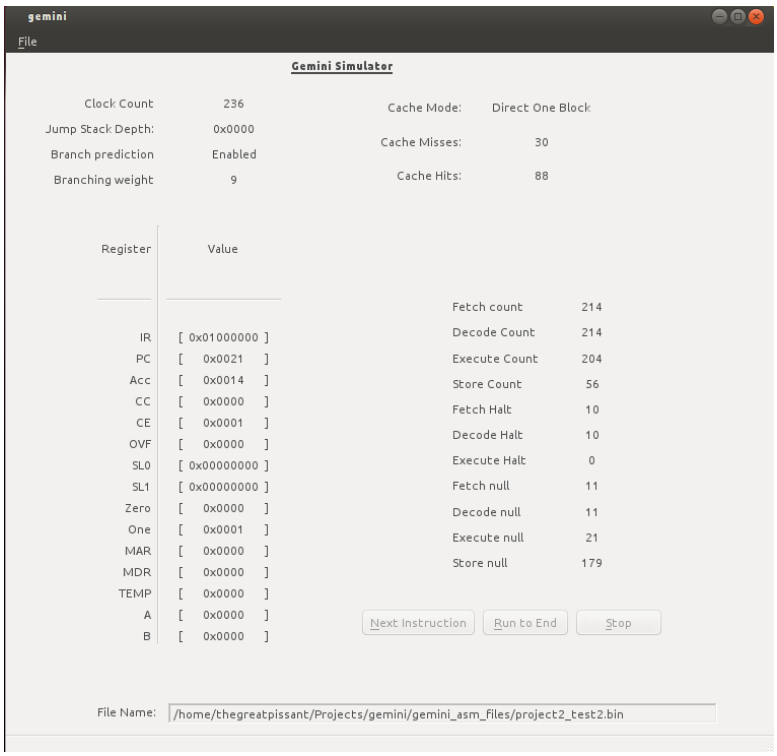


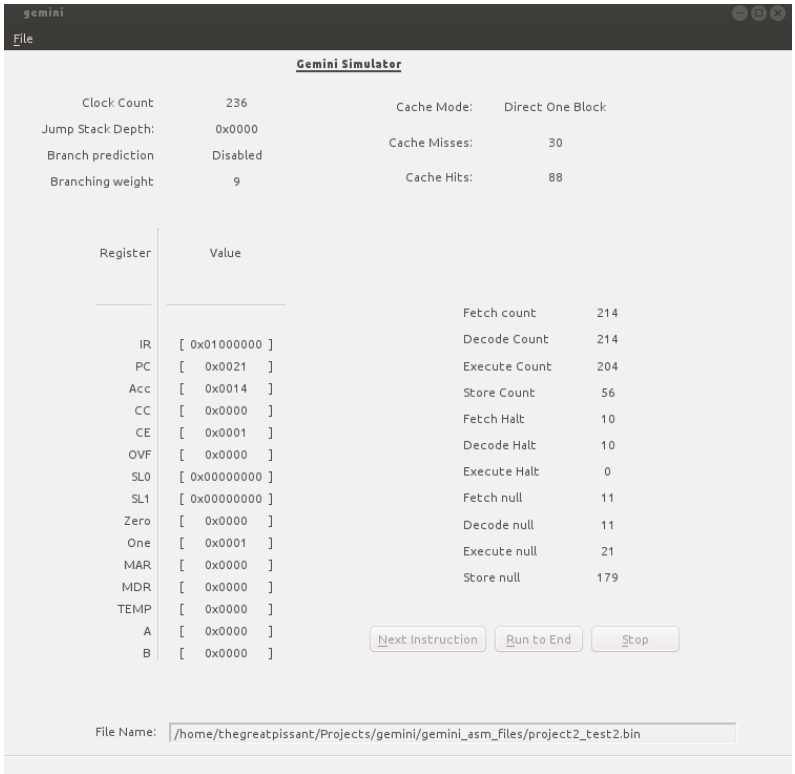
Project2_test1.s 4 Branch Prediction runs



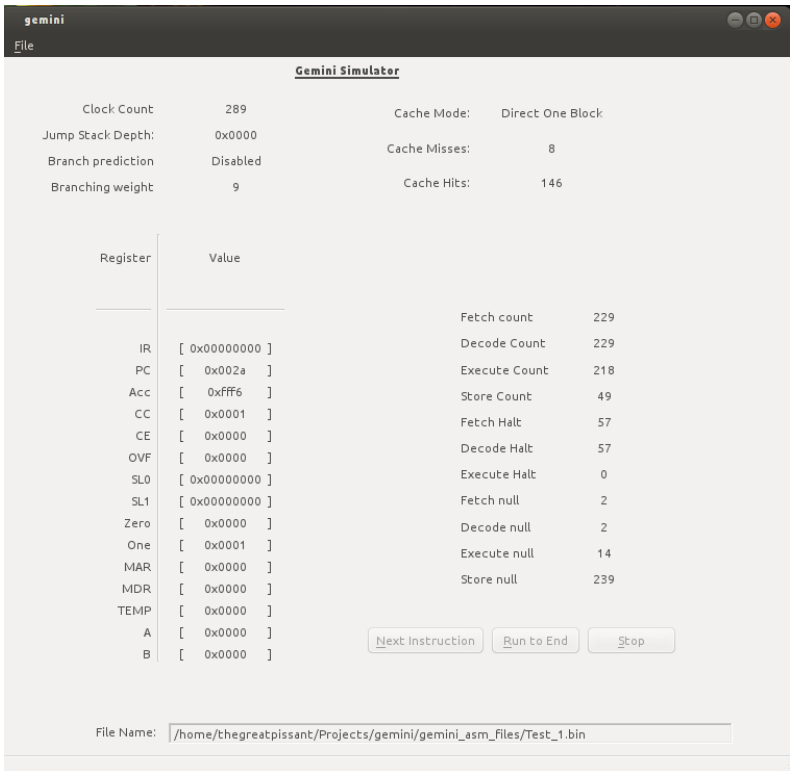
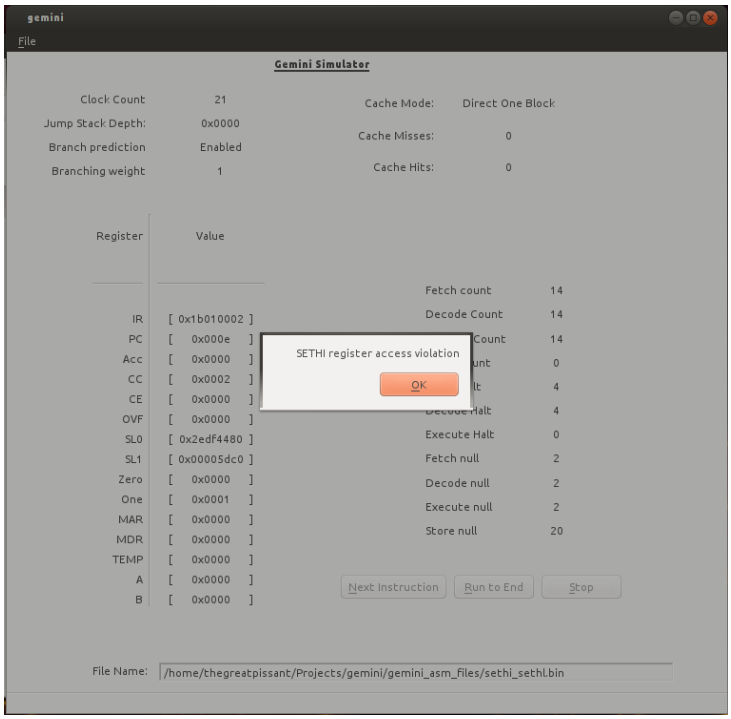


Project2_test2.s 4 Branch Prediction runs





Test_1.s 4 Branch Prediction runs



sethi_sethl.s runs