ESE 356

Project 1 Phase 2

Fengwei Zhang, Justin Fan

The goal of phase 2 is to design the controller for the mini-RISC processor. The design is a 5 stage pipeline. The 5 stages are instruction fetch(IF), read and decode(RD), execute(EXE), memory access(MEM), and write back(WB). The control signals and other signals that were needed at each stage of the pipeline were determined. The controller has 4 internal registers. The 1st level simply stores the instructions from the program counter. The 2nd level stores the control signals generated by the decoding logic and the data from the RF. The 3rd level stores the memory access control bits and Rdest and the output of the ALU. The 4th level contains the data needed to write to the RF.

The decoding logic in prc\_decode is done using nested switch statements. The first switch looks at the OpCode of the instruction and the inner switch statement looks at the extended OpCode. The process prc\_pipe sends the data required for the next pipeline register to the registers. The process prc\_update updates the outputs based on the new data in the pipeline registers.

Branch and jump are implemented by checking the PSR bits for the corresponding instruction in the decode logic. If the branch condition is met, the bool variable branch is set and the same is done for jump instructions. In prc\_update, the status of branch and jump are checked and the program counter is changed if the conditions were met.

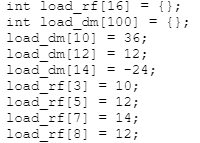
The ALU takes the control signals and performs the specified operation. This is done using a switch statement using alu\_op as the case.

The main program initializes the program memory, data memory, and register file and runs the simulation until all instructions are completed. To translate the instructions into binary, the instruction table in the project design requirements was used. The first program was translated into hexadecimal. The translated program is shown below.

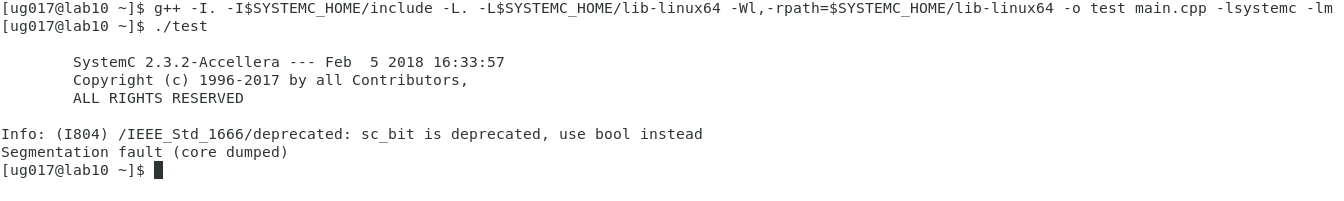


The nop instruction is put after memory access instructions. The ending is padded with zeros to prevent the program counter from accidentally going past the limit of the program memory.

The rf and dm are initialized by making an array and putting data where it is needed. This results in an array with zeros other than where the data has been put.



The controller and data memory modules are instantiated and connected. The clock is 50ns meaning it would take around 750ns to complete all instructions. However, when the main program runs and the simulation is started, an error occurs.



The segmentation fault means invalid memory was accessed. The source of this error could not be found and verification of the modules and system could not be completed.