nItebHa', Doch cha'

By: Lucian Hymer, Irsyad Hanif, Emily Heinrichs

***Abstract*—This assignment required us to take the previously designed pipelined processor implementation and add superscalar execution capable of executing more than one instruction per clock cycle under specific circumstances.**

I. Introduction

The goal for this assignment is to modify the pipelined loQ Don processor and add superscalar and out of order execution. Since most of the processor was already built, we focused efforts on managing how and when these multiple instructions would be executed. The processor would first fetch two instructions. Then, it checks if the first instruction is an ALU operation and if the second operation is a store operation. If both of these conditions are satisfied, the processor enables a secondary pipe to execute the store operation and passes the ALU operation through to the primary pipeline.

For out of order execution, the processor checks if the next two instructions are ALU operations and the two instructions after that are store instructions. If both of these conditions are satisfied, then the processor reorders the instructions to be interleaved with each other.

II. Overall Design

Since most of the processor was already built, we only needed to write the Verilog for the control system for both the superscalar and out of order execution.

The superscalar execution is implemented by fetching two 16-bit instructions as a 32-bit word per clock cycle. The processor then checks if the opcode of the first fetched instruction equates to an ALU opcode. It also checks if the opcode of the second fetched instruction equates to the store opcode. If both of these are true, the processor enables the secondary pipeline to execute the store instruction in parallel with the ALU operation. Otherwise, the processor continues as normal and executes the 32-bit word sequentially, starting with the high instruction.

The out of order execution is implemented by additionally prefetching two more 16-bit instructions and adding them to the 32-bit word, creating a 64-bit word. The processor then checks if the first two instructions are ALU instructions and the next two operations are store instructions. If both of these are true, the processor executes them out of order, executing the first ALU instruction, the first store instruction, the second ALU instruction, and the second store instruction in that order.

Our design speculates that a jump is not taken, squashing the instruction if it turns out that the jump was taken.

III. Testing

To test if both superscalar and out of order execution work, two loQ Don assembly functions were created. The function testing the superscalar execution has a ALU operation right before the store operation. If the superscalar execution is performed successfully, the processor outputs an indicator showing that superscalar execution is being performed. The function testing the out of order execution has two ALU operations right before two store operations. If the out of order execution is performed sucessfully, the processor outputs an indicator showing that out of order execution is being performed. The processor also outputs the values of each important variable for each clock cycle. You can see that when out of order execution is being performed, the program counter isn't sequential - it skips.

IV. Issues

The test The processor was designed successfully and works as intended. The pc value is not stored in register

0

properly.

V. Conclusion

We were able to sucessfully design a superscalar processor with out of order execution under certain circumstances.

References

[1] Dietz, Hank. "An Introduction to Verilog", University of Kentucky, 15 September 2016 http://aggregate.org/EE480/slidesF16v0.pdf.