Project 4: Integrated Processor

1 Overview

In this project, you will integrate your prior simulators (P1, P2, P3) into a single simulator.

2 Logistics

This is a partner project. You are permitted to share code only with your partner. You may use C or C++ for this assignment, and you should only rely on standard libraries.

You are extending the processor/ component into your own superscalar processor simulator. You may also need to make updates to the other components. Your components should work with the reference, so do not change the interface.

3 Simulator Specifications

The traces will now contain the branch and memory operations from before. These operations have been extended to have an optional register dependency. Each is to be executed as an ALU operation. Memory operations have a latency determined by the cache simulator. Only the explicit memory operations (L and S) need to be modeled. You do not need to model an instruction cache.

Branches are also executed as an ALU operation. If a branch is correctly predicted, then the trace is already providing the simulator with the correct operations to execute. If the branch is mispredicted, then the next operation in the trace is the one executed after resolving the misprediction and therefore fetch should be stalled. Branches update the fetch unit via a common data bus during state update.

4 Evaluation

Evaluation of your submission is based on the correctness of the simulator. It can be difficult to compare the results for longer traces. The reference processor will print out for each stage the cycle for when the instruction first runs that stage. For longer traces, you may want to redirect this output to a file and use diff to compare the results.

Your report for this project should integrate your reports from the cache and branch predictor simulators (with any updates you wish to make), as well as a new third section exploring their integration. Possible analyses include how a superscalar pipeline handles cache misses and branch mispredictions. What resources ameliorate the impact of these stalls?

Your report should also answer the following questions:

1) Why does the cache use a callback function when the request is complete rather than returning the number of cycles to wait?
2) On a branch misprediction, the simulator does not simulate fetching any operations. Why is that reasonable, even given the superscalar pipeline?

5 Handin Instructions

Please submit a tar of your repos contents to Autolab.

6 Hints

Below are some hints to guide you through this project:

- We recommend you first work out by hand what should occur in each unit at each cycle for an example set of instructions before you start writing your program. (e.g., traces/ls-proc.trace)
- Keep a counter for each line of the tracefile to use for Tomasulo tags.
- Operand renaming is not necessary for this assignment.
- You may find it useful to make a separate function for each pipeline stage.
- Execute the procedures in reverse order from the flow of instructions.
- It may be helpful to have a data structure for all of the pipeline latches, e.g. a two dimensional array, one dimension for the number of execution units, another dimension for the number of pipeline stages of the execution units.

7 Conclusion

Please don't hesitate to reach out regarding clarifications or questions. Since this class is new, we would love to act on any feedback you can provide about the assignments.

Good luck on your integrated simulator!