Noah Perryman

Patrick Lyons

Effect of Having Two Pipelines and of Having a Branch Predictor

In part one of the project we implemented a branch predictor into the simulated pipeline. Including a branch predictor into the pipeline is designed to try and improve the flow of the pipeline. In this project the affect was not seen entirely due to the simple simulation of the pipeline. However, when actual instructions are executed there is a clear improvement. With branches, if the branch is predicted correctly, the instruction doesn’t have to be executed in the sense that its branch address is already known. This allows the instruction to be carried out while reducing latency. This does not affect the overall number of cycles when the branch is predicted correctly but rather increases the efficiency of a specific branch due to the address of the next instruction being known. However, if the branch is predicted incorrectly the instructions preceding the branch are negatively effected. In this project the instructions were squashed and disposed of, which would hurt the data path, as those instructions are important. Overall, branch prediction aims to lower the cost of fetching the next instruction through its address after a branch.

In part two of the project we implemented a very simple version of a 2-issue architecture with two 5-stage pipelines. The effect of this on the pipeline in general is that it improves the throughput of the pipeline. While including two pipelines increases the complexity of the design the overall throughput is increased as shown in the output from the trace files when compared to the single pipeline implementation. While there are situations where data hazards introduce no-ops to both pipelines there is a still a definite increase in throughput from splitting into two pipelines, as the number of cycles is essentially cut in half, as each cycle outputs two completed instructions.