Topic: Branch Prediction.

Aimed for: Enhancing Performance *Author: Abdullah Matar*

Objective

Enhancing performance by minimizing the delays caused by control hazards during instruction execution in a pipelined processor.

Motivation

In pipelined architectures, branches introduce uncertainty about the next instruction to fetch, as the outcome of the branch may not be known immediately. Branch prediction techniques aim to guess the direction (taken or not taken) and target address of a branch instruction to ensure the pipeline remains filled with the correct sequence of instructions. Accurate branch prediction reduces pipeline stalls and improves overall processor throughput, enabling faster execution of programs.

Types of Branch Prediction

- **Static Branch Prediction**: Assumes a fixed outcome for branches, for example, always "taken" or "not taken", or based on reasoning like backward branches being "taken" and forward branches "not taken". The following figures shows the pipeline when the assumption is “branch not taken”.

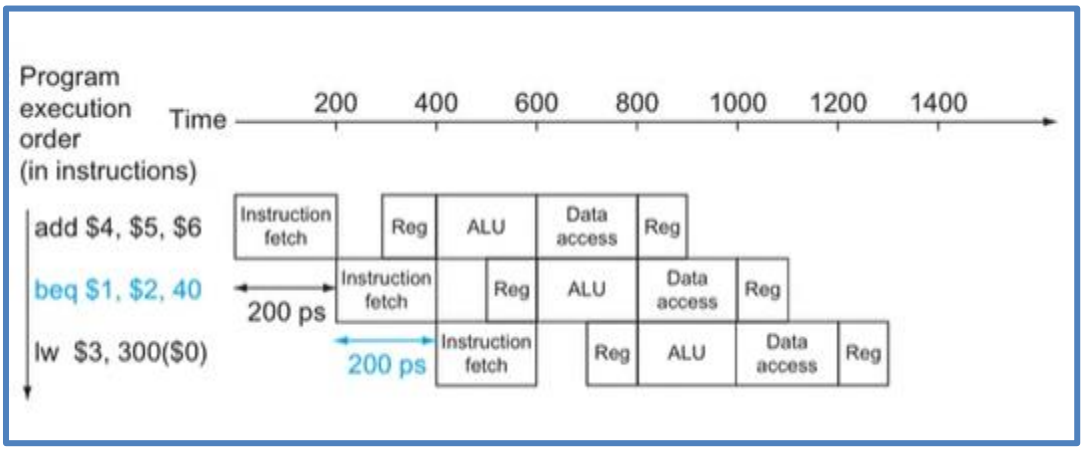

Figure 1: Branch not taken (doesn't cause a stall)

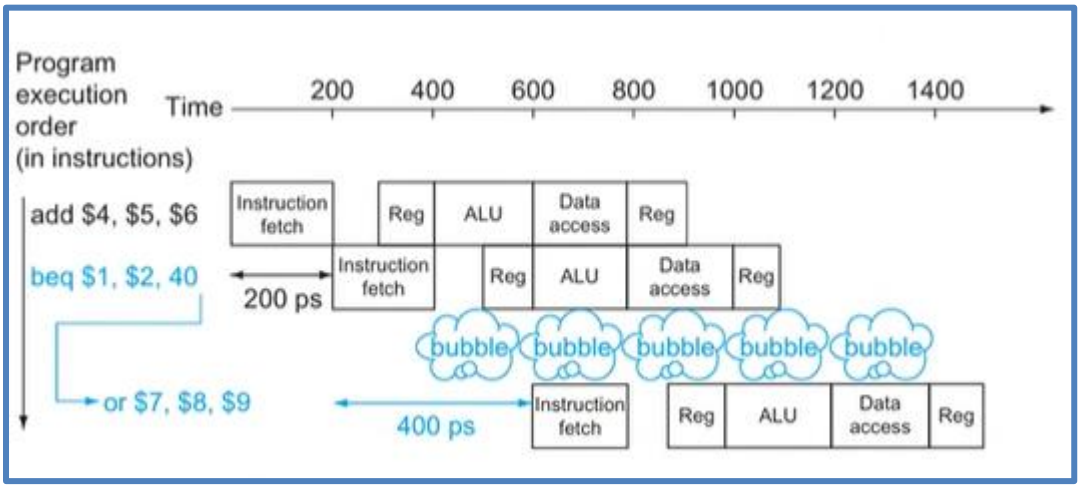

Figure 2: Branch taken (caused a stall)

- **2-Bit Branch Prediction**: A dynamic technique that uses a 2-bit counter to track branch behavior, requiring two mispredictions to change the prediction direction, making it more resistant to single anomalies in branch behavior.

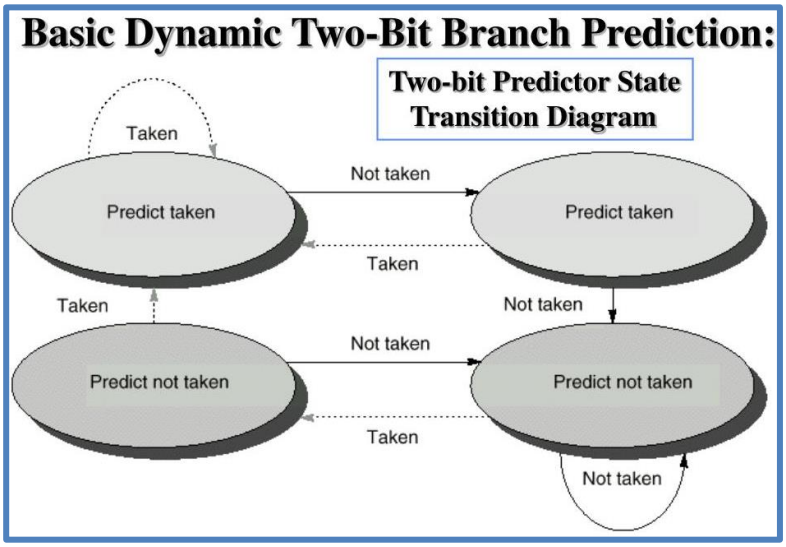

Figure 3: 2-bit dynamic prediction workflow

- There are several different dynamic branch prediction techniques, each with its own approach to improving prediction accuracy and processor performance. These include methods like local history prediction, global history prediction, and hybrid predictors, among others. You are free to use whichever technique you prefer, as long as it is implemented correctly and effectively enhances the overall performance of your pipelined processor.

Workflow

- **Understand the Pipeline**: Identify where and how prediction will fit.
- **Choose a Technique**: Select static or dynamic prediction based on goals.
- **Design Predictor**: Create logic (e.g., tables, counters) for predictions.
- **Integrate with Pipeline**: Connect predictor to instruction fetch.
- **Handle Mispredictions**: Implement pipeline flush and recovery.
- **Test & optimize**.

Conclusion

Branch prediction plays a critical role in enhancing the performance of pipelined processors by addressing control hazards and ensuring efficient instruction execution. By selecting and correctly implementing appropriate prediction techniques, whether static or dynamic, and integrating them effectively within the pipeline, you will be able to minimize stalls and maximize throughput.