Ana Camba
Fabiana Gonzalez
6.192

Proposal Outline for Superscalar Processor and Cache Design Comparison

Title: Enhancing Processor Performance through Superscalar Design and Optimized Cache Implementations

Team Members:

Ana Camba:
- Email: anacamba@mit.edu
- GitHub username: anacambag

Fabiana Gonzalez:
- Email: fabianag@mit.edu
- GitHub username: fabianaagonzalez

Project Overview:
Our project aims to enhance the processing capabilities of a baseline processor built in Lab 4 by integrating a superscalar architecture and comparing the performance impact of direct-mapped and 2-way set-associative cache designs.

Objectives:
- Implement a superscalar processor capable of fetching and executing two instructions simultaneously.
- Compare the performance implications of a direct-mapped cache with a 2-way set-associative cache in the context of the superscalar architecture.

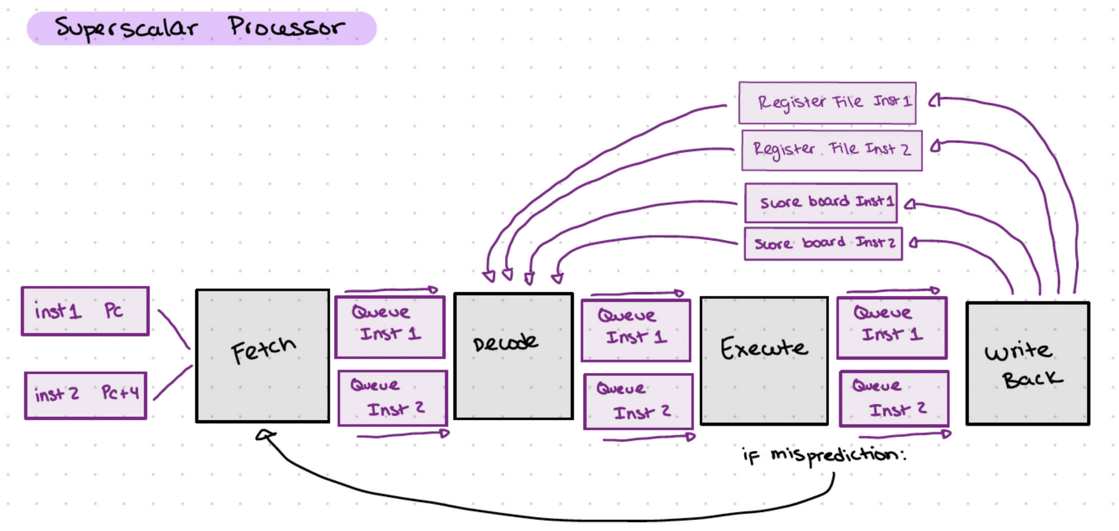
Weekly Objectives:
- Week 1: Complete the design specifications for the superscalar enhancements, including modifications to the fetch, decode, execute, and writeback stages.
- Week 2: Implement the superscalar processor modifications in simulation.
- Week 3: Develop and implement the 2-way set-associative cache and integrate it with the superscalar processor.
- Week 4: Benchmark performance differences between the direct-mapped and 2-way set-associative caches using test cases designed in C/assembly.

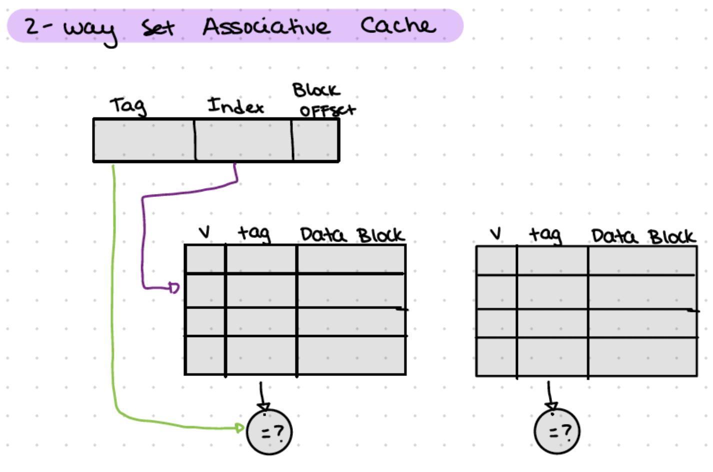
Superscalar Enhancements:
- Fetch: Modify the instruction fetch mechanism to simultaneously retrieve two instructions. Implement boundary checks for cache line crossings.
- Decode: Duplicate the decoding logic to handle two instructions concurrently. Integrate a complex scoreboard system to manage dependencies and handle stalls dynamically.
- Execute: Explore options for duplicating ALUs or other functional units to allow parallel execution of independent instructions.

- Writeback: Increase the number of register file ports to accommodate simultaneous writebacks.

Cache Design Comparison:
- Direct-mapped cache: Currently implemented; baseline for performance comparison.
- 2-way set-associative cache: Implement and test a 2-way set-associative cache to evaluate performance improvements, particularly in reducing conflict misses. Implement LRU (Least Recently Used) as the replacement policy.

Challenges and Solutions:
- Handling the program counter (PC) issues where PC+4 is not in the same cache line, leading to potential fetch problems. Solution: Implement logic to handle single instruction fetch if boundary crossed.
- Dependency resolution between two simultaneously fetched instructions, potentially leading to stalls. Solution: Implement stall logic in the decode stage.
- Address writeback conflicts where two instructions might attempt to write to the same register. Solution: Extend the existing register file to handle multiple write accesses.


Superscalar Processor
Register File Inst 1
Register File Inst 2
Score board Inst 1
Score board Inst 2
inst 1 PC
inst 2 PC+4
Fetch
Queue Inst 1
Queue Inst 2
Decode
Queue Inst 1
Queue Inst 2
Execute
Queue Inst 1
Queue Inst 2
Write Back
if misprediction:


2-way Set Associative Cache
Tag
Index
Block offset
V
tag
Data Block
V
tag
Data Block
=?
=?