AmdaZulo

A Superscalar LC-3b Processor

By Steve Hanna
Tom Hughes
Mark Murphy

AmdaZulo

Amdahl's Law

\+ Toma**sulo**'s algorithm

AmdaZulo

Superscalar Design

- Four-Wide fetch/decode/issue
  - Fetch/Predict/issue only 1 control instruction per cycle
- All Caches Fully Associative
  - Exact LRU for each cache
- Out of Order Memory Queue
  - Stores serialize, Loads OOE
- Register Renaming
- Speculation

Design Decisions

- Cost was never a consideration, performance was our only goal.
- Utilized cost-wise impractical methods to achieve maximum performance.
- Attempted to simplify some design areas due to the extreme complexity of our processor.
- Read architecture papers to learn about various designs.

Pipelines

- 2 ALU and 2 Memory pipelines
  - Done to decrease amount of arbitration logic necessary to dispatch instructions.
  - Each set of pipelines has its own set of reservation stations (16).
  - The ALU pipleline is one stage shorter than the memory pipeline.

Caches

- All caches are fully associative
- Caches include:
  - L1 Data Cache(512B)
  - L1 I-Cache(4kB)
  - L2 D-Cache(4kB)
  - Speculative Cache(32B)
  - BTB(128-entry cache)
- Arbiter controls DRAM access between L2 D-Cache and L1-ICache
- All caches have exact LRU
  - (Impractical amount of logic)

Memory Queue

- Loads between stores can be executed out of order.
- Speculative stores are sent down the pipeline twice.
- Special speculation mechanism.

Register Renaming

- 32 Physical Registers.
- Register Alias Table.
- Reference counts to determine free registers.

Speculation

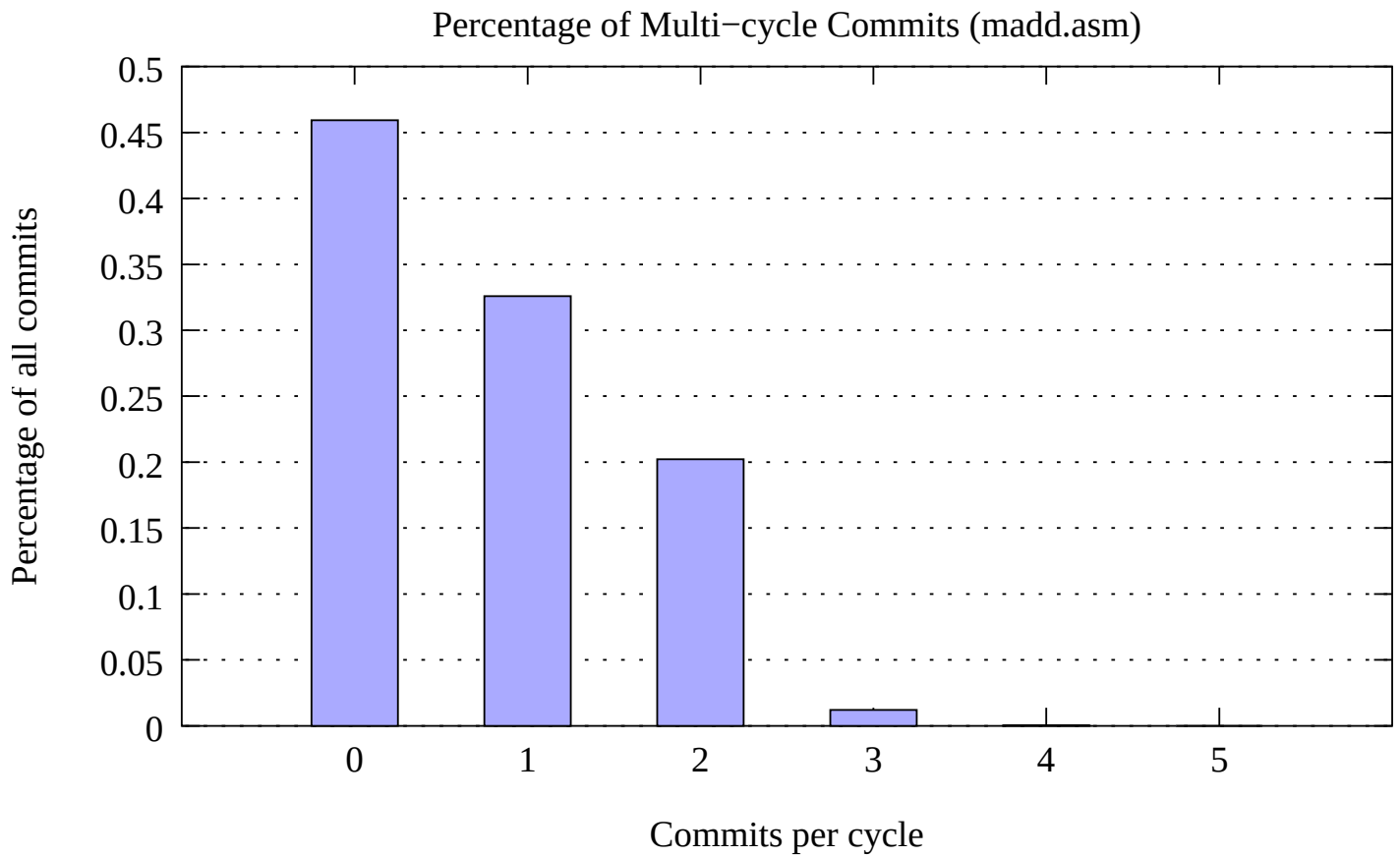
- Superscalar without speculation impedes upon performance.
- Out of order commit, no reorder buffer!
- “Turn on a dime recovery”
- Maximum of two speculative control instructions.
- Tournament Branch Predictor

Cost Analysis

- Impractical cost, hardware would be incredibly expensive.
- Checkpoint reference counts, RAT and control state (~.5kB of state) backed up.
- LRU and Fully Associative Caches would be very expensive.

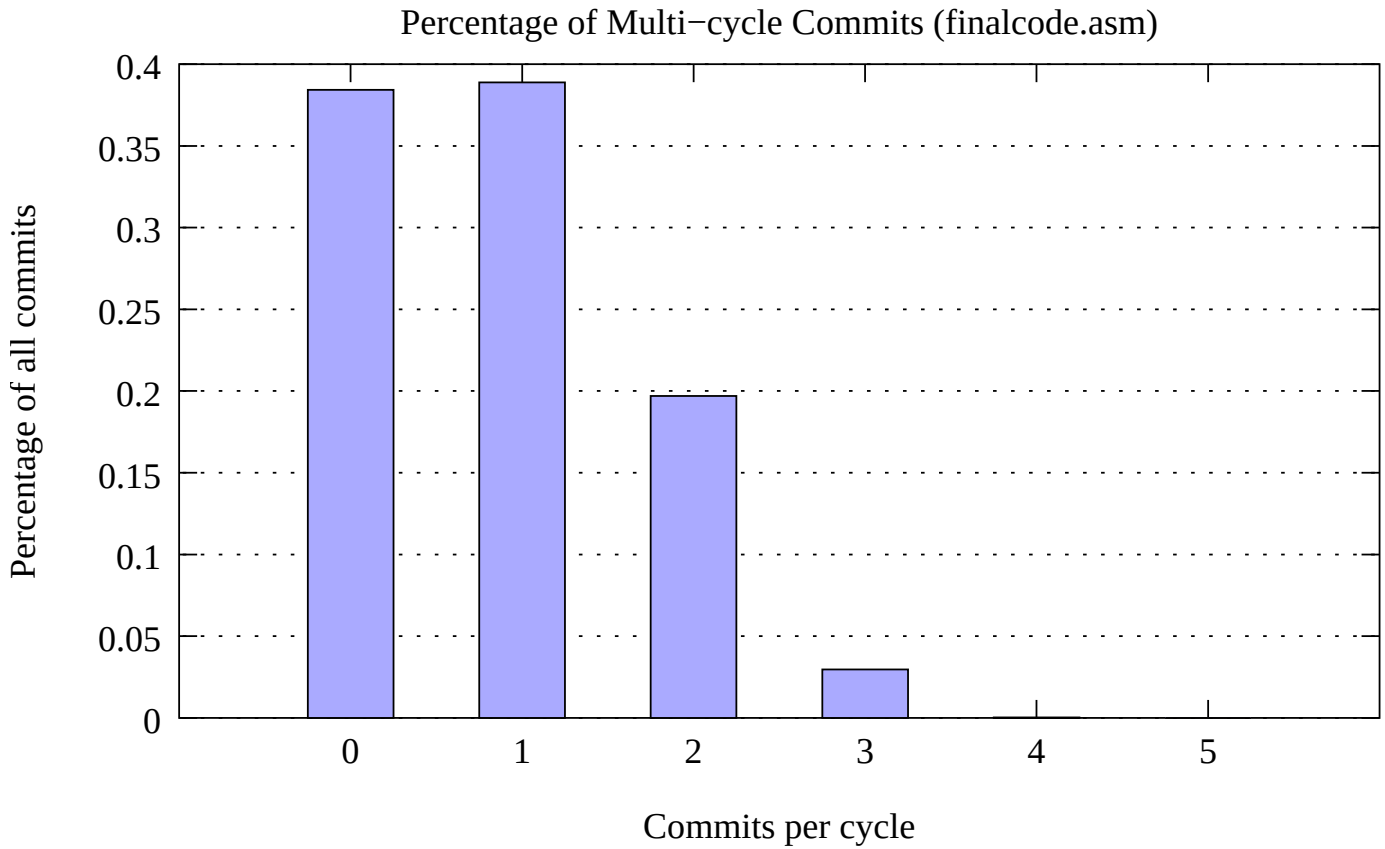
Performance Analysis

- Vector Adding



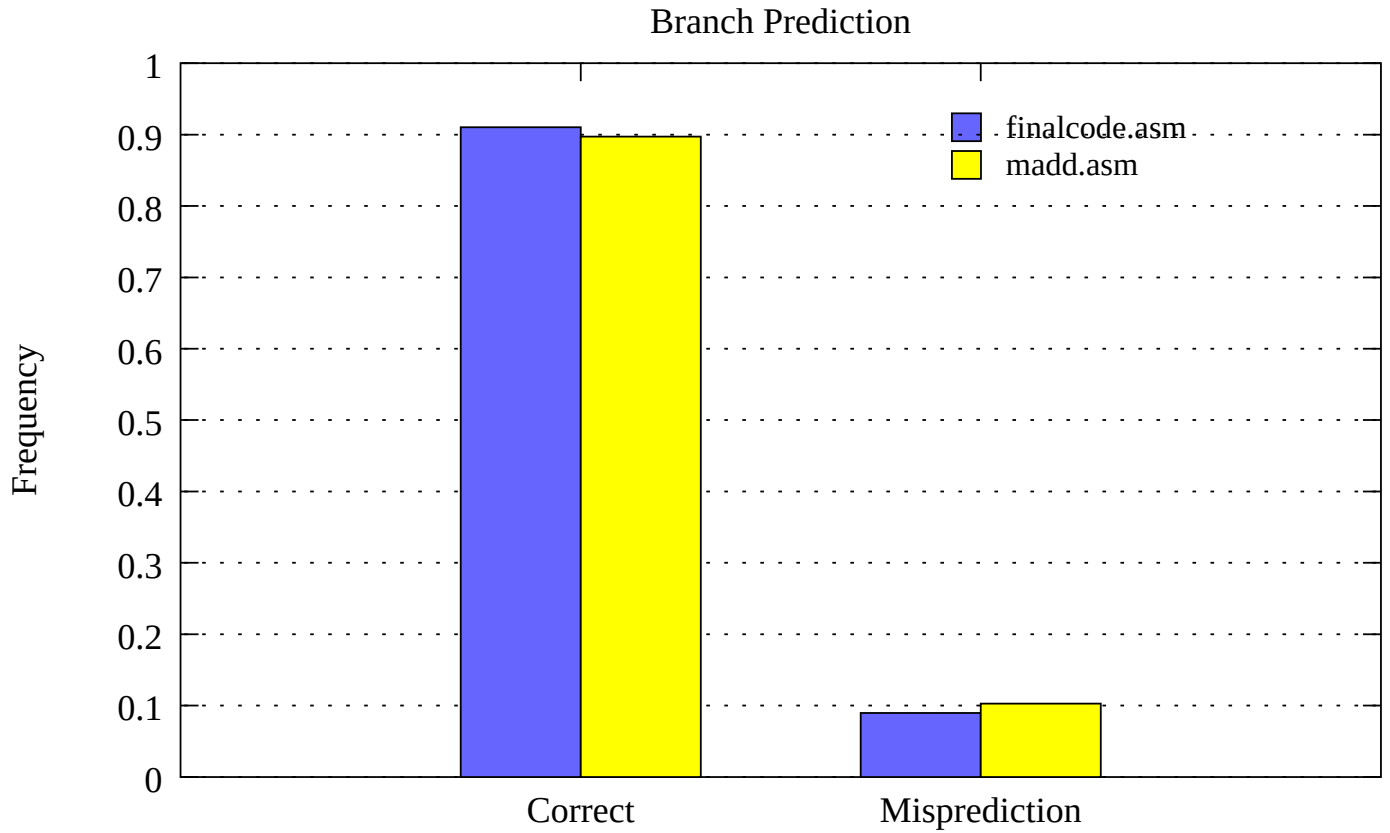
Performance Analysis

- Handin Code



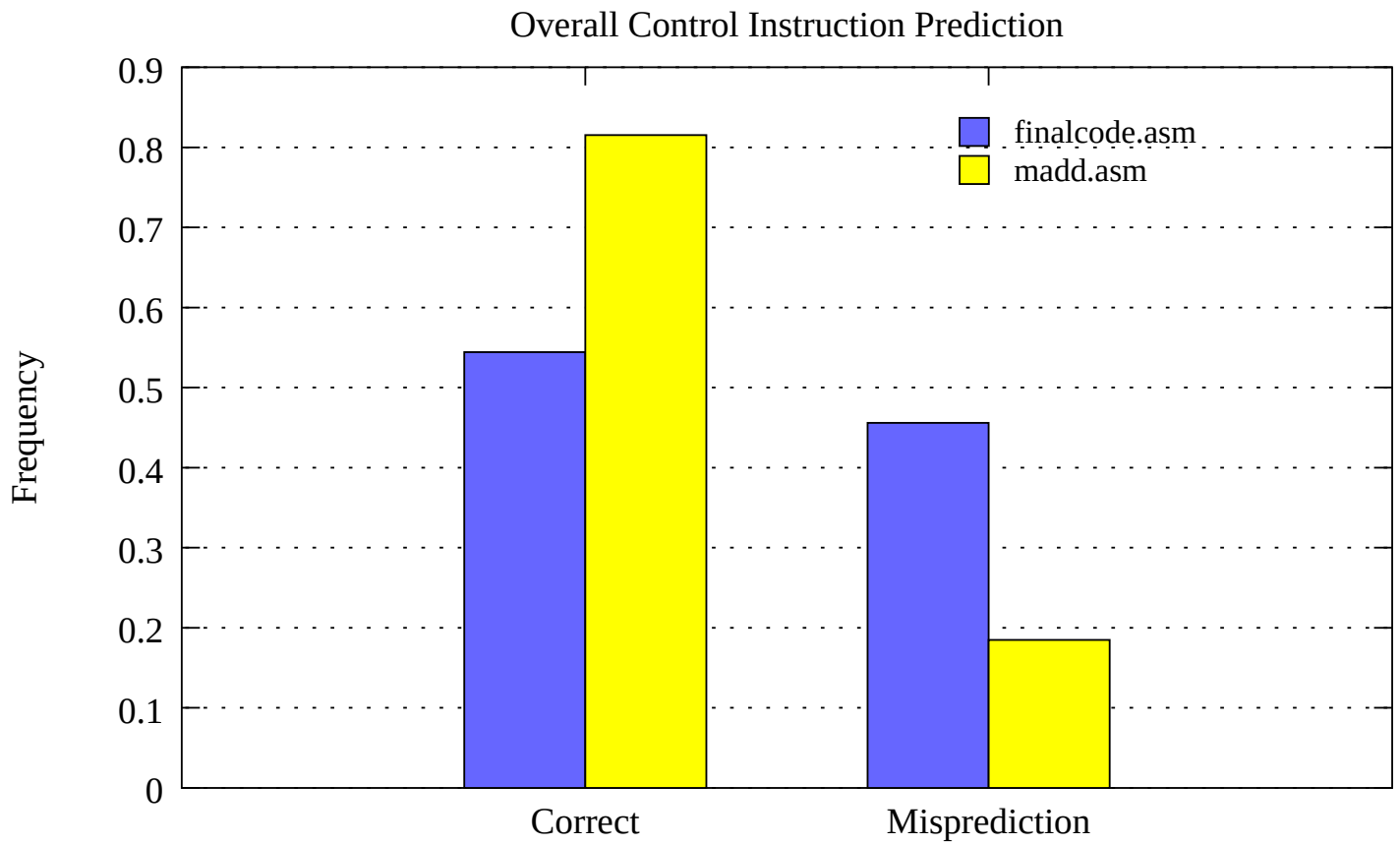
Performance Analysis

- Branch Prediction

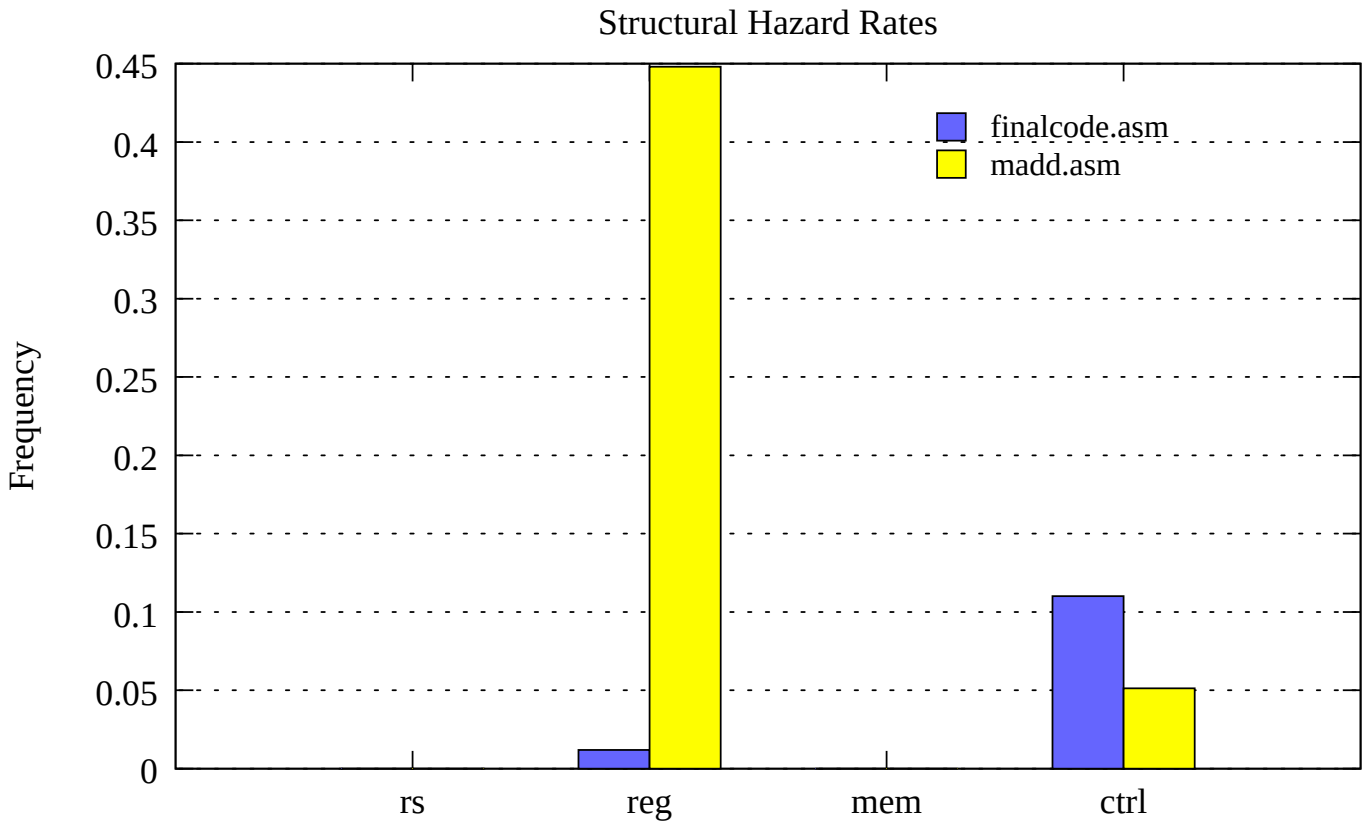


Performance Analysis

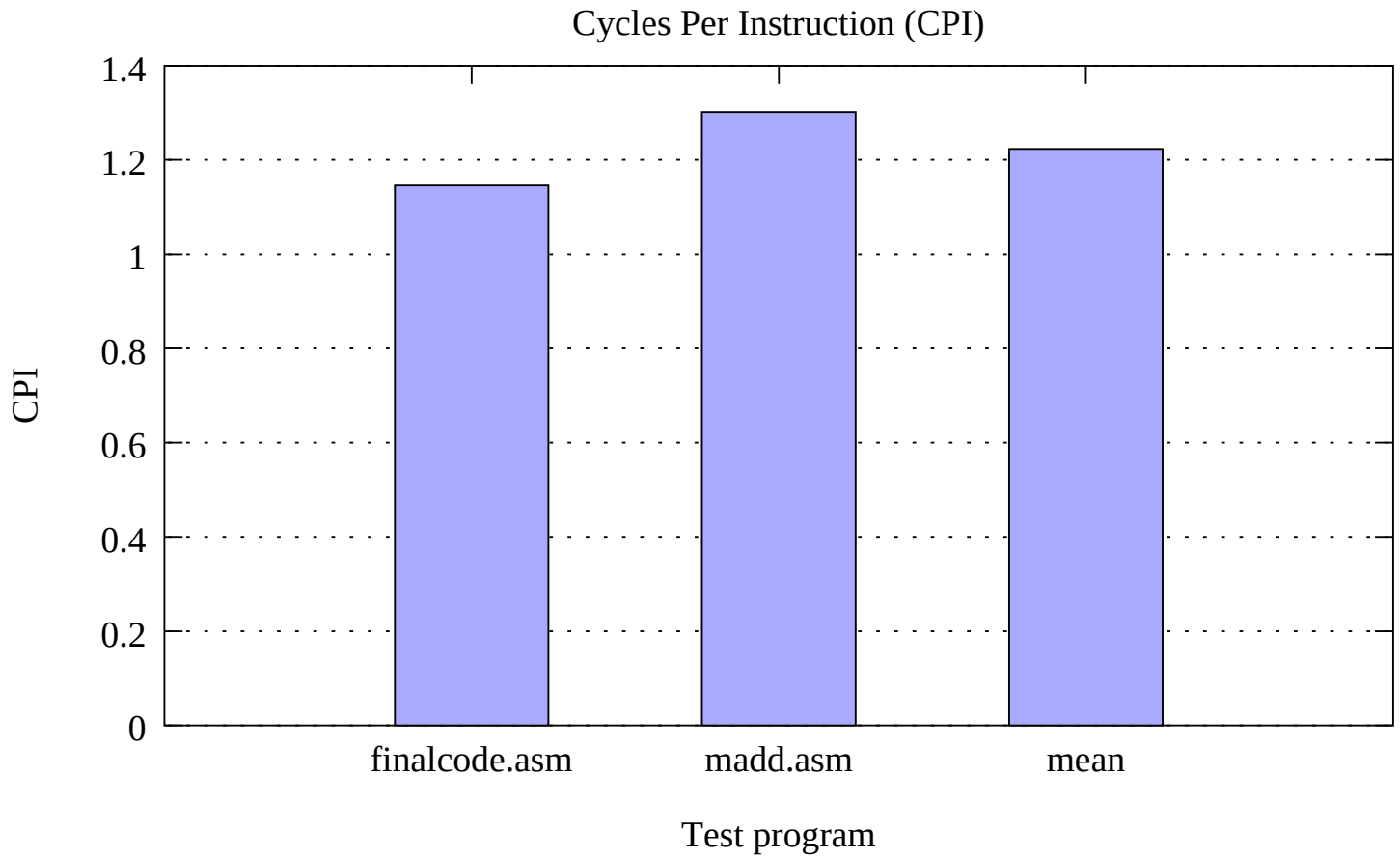
- Overall Control Instruction Prediction



Performance Analysis



Performance Analysis



Conclusion

- Completing a superscalar design in a semester is intense.
  - ~1000 total hours
  - Good group dynamics
- More work is needed to predict indirect branches.