2-Way, 5-Stage super-pipeline MIPS Processor

Microarchitecture Performance Impact

- IPC > 1

- CPI >= 1

Microarchitecture Design

Duplicates functional units to create n-ways (maintains memories but increases number of ports on them) that allows ideal IPC = n

Microarchitecture

- Has logic to ensure the two simultaneous operations don’t have dependencies(Modern GPs use multiple techniques to achieve maximum optimization thus reducing number of empty execution slots)

- Dependencies that result into hazards are tackled using OOO, aggressive Branch predictors and register renaming techniques while employing one of the ff.
  - VLIW
  - Statically Scheduling Instructions(OOO left to compiler, no Hardware OOO capability)
  - Dynamically Scheduling Instructions(OOO is independent of compiler and done at execution Time)

Implementation of an **In-Order**, 2-way/issue, 5 stage, super-pipeline, 32-bit MIPS RISC based processor

Design Proposal

- Minimal optimization
- Avoid OOO issue.
- PC’s next value is determined by scalar way 1
- While execution is in-order, employ some form of Scheduling;
  - If two instructions issued simultaneously occur in the order A, B such that A is a Branch Type or Jump Type instruction then make B a nop. Since PC address is determined by scalar way 1, PC = BranchAddress

- If two instructions issued simultaneously occur in the order A, B such that A is an integer instruction and B is a Branch or Jump Type. Make Instruction A read as B and B as A. Allowing next

  PC = Branch/Jump Address;

- The above is under condition that, A and B are independent.

- If A and B are dependent on each other, and a issued in the same cycle, like condition 1, set B to a nop, and update PC to PC+4. This loads B and B's subsequent/succeeding instruction

- Any hazards b/n subsequent issues employ pipeline techniques to avoiding hazards

- Can serves as hardware for a statically scheduled OOO architecture

Performance Analysis

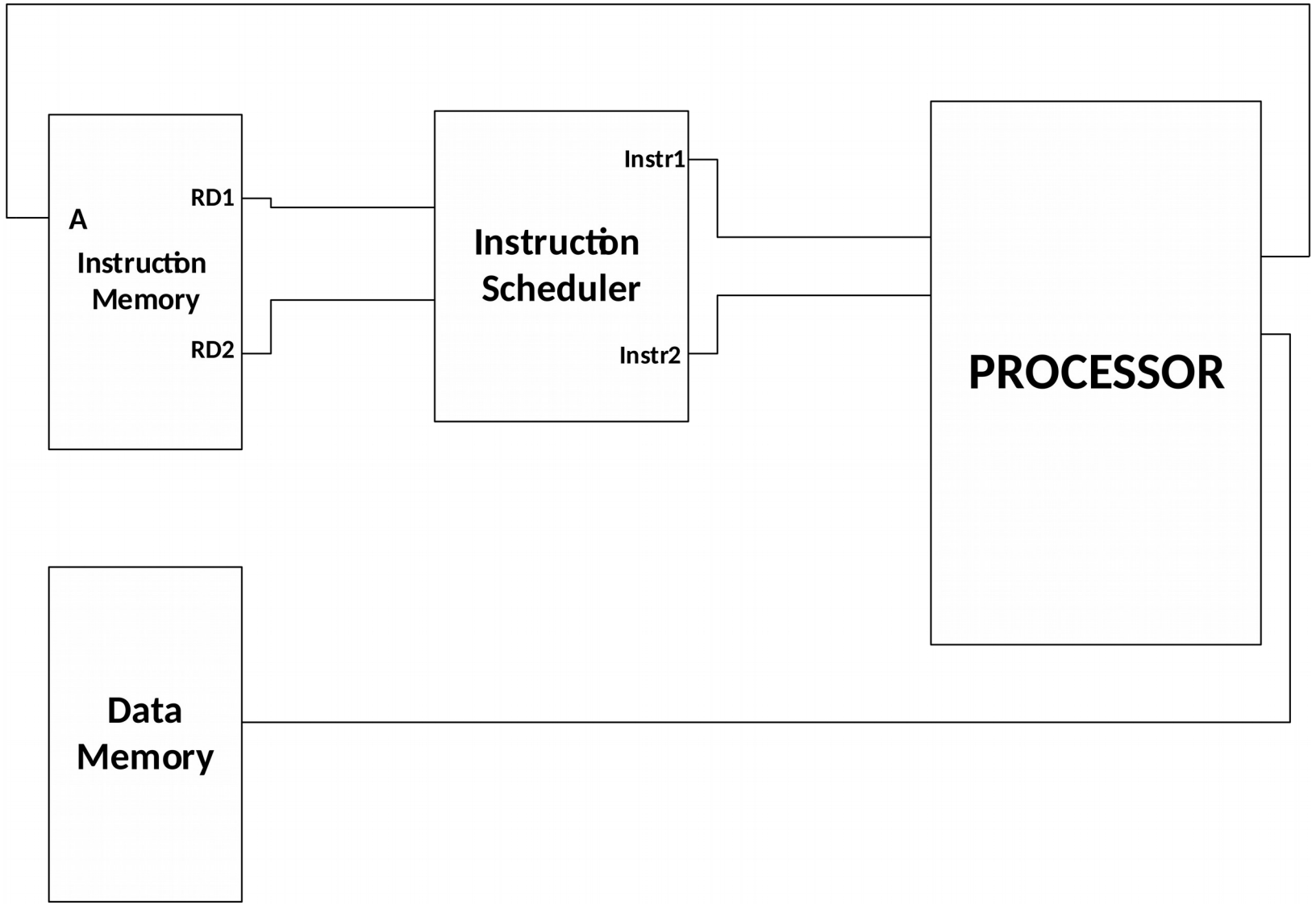
- Determine execution time and throughput performance metrics, when all issued instructions at a time are independent.

- Determine execution and throughput performance metrics, when all issued instructions are dependent on each other.

- Determine execution and throughput performance metrics, when some issued instructions at a time are dependent on each other.

- Determine execution and throughput performance metrics, when 3 is reordered. This can be assumed as the case of a scalar processor with static scheduling

- Compare and analyze results.



Top Level Module Of Proposal

NEXT STEPS

- Implementation of a 2-way scalar, 5 stage pipeline, **Out-of-Order**, Dynamically scheduled, 32-bit MIPS RISC based processor

  (Tomasulo's Algorithm)

- Analyze results with built In-Order processor