Four Instructions to Help Explain Superscalar Out-of-Order

James C. Hoe
Computer Architecture Lab (CALCM)
Carnegie Mellon University

Motivation: CMU 18-744

- Advanced graduate-level design course
  - 2 pre-req courses in computer architecture
  - 15~20 students (self-selected group)
  - Has been taken by a few seniors
  - Multiple design projects

- Emphasis
  - deep/true understanding
  - hands-on implementation
  - teamwork and time management
**R10K OOO Core** [Yeager, Micro, Apr 1996]

**Mini-ISA**

- **SUB** rd, rs, rt
  - rf[rd] ← rf[rs] - rf[rt]
- **ADD** rd, rs, rt
  - always causes an exception when executed
- **BEQ** rs, rt, offset
  - always confirms its prediction when executed
- **BNE** rs, rt, offset
  - always reverses its prediction when executed
Verilog Simulation Environment

RTL Core

isource

oracle checker

Instruction Source

- Generates a new cache line (0 ~ 4 valid inst) per cycle
- Valid inst indicated by one-hot encoding on valid[3:0]
- Each inst has a corresponding sequence number (sn)
Stalling Instruction Fetch

- A new instruction cache line is fetched for every clock edge that accept is asserted on

- When stalled, entire cache line is held in place

Fetch Restart

- After each restart, all instructions between the first exceptional instruction (ADD, BNE) and the next restart (inclusive) are on the wrong path

- restart: a 1-cyc pulse, fetch is valid in the next cyc
Basic Rules of the Game

- Match the results of the oracle “checker”
- Instruction must issue as soon as possible
  - obeying (true) data and structural hazards
  - back-to-back dependent instructions must be capable of issuing on consecutive cycles
- Branch rewind must be fast (O(1) time
  - independent of the number of instructions in the ROB)
- Exception rewind can be slow (O(n) time
  - where n is the number of instructions in the ROB)
- Can’t cheat on the semantics of ADD and BNE
- Has to be synthesizable!!!

Project Timing (Teams of 2~3)

- Step 1: 1-inst-per-cycle, renaming and microdataflow
  - 2.5-weeks
- Step 2: 1-inst-per-cycle, branch rewind
- Step 3: 1-inst-per-cycle, branch rewind & exception
  - 2 weeks together
- Step 4: 4-inst-per-cycle, every thing goes
  - 1.5 week

Specialized isource module in each step
Concluding Remarks

- Most unexpected outcome
  - everyone finished on time and liked it
- Most challenging design problem
  - fast branch rewind (register freelist)
- Most rewarding feedback
  - company recruiters loved it!!
- Project materials
  - http://www.ece.cmu.edu/~jhoe/superscalar