Complex Pipelines

Ryan Lee

(slides adapted from prior 6.823 offerings)
Dependence vs. hazard

• Dependence is a property of programs

• Whether a dependence results in a hazard is a property of pipeline organizations
Data hazard types

- RAW
- WAR
- WAW

I1: ADDI  
f0, f0, 0

I2: ADDI  
f3, f0, 3

I3: ADDI  
f4, f0, 4

I4: ADDI  
f0, f5, 1

I5: XOR   
f6, f6, f6

I6: ADDI  
f0, f7, 1

Reads/Writes to f0

- R (I1)
- W (I1)
- W (I4)
- W (I6)
Scoreboard

• A data structure that detects hazards dynamically

• Applicable to both in-order and out-of-order issue

• Why do we need this?
  • Many execution units
  • Variable execution latency
  • Dynamic instruction scheduling
Scoreboard

• Can have many implementations!
• Example: In-order issue
  • WAR cannot happen (if value is latched to functional unit at issue)

  I1: ADDI      f1, f0, 1
  I2: ADDI      f0, ❸··· f2, 1

  Due to in-order issue
  Register read happens before write for an instruction

• Can be simplified as Busy[FU#] and WP[reg#] (if WAW resolved conservatively)
Scoreboard

• What strategy does it use to resolve RAW?
  • Stall

• How about bypass?
  • Less beneficial since the register write can happen right after execution finishes
  • Can still be incorporated to allow register read and write to happen in the same cycle
Out-of-order execution

• Want: we want to somehow avoid stalling due to WAR and WAW hazards...
  • Strategy?
    Do something else
  
• Technique?
  Register Renaming

• Problem: Imprecise exceptions
Questions?