Complex Pipelines

Ryan Lee

(slides adapted from prior 6.823 offerings)
Agenda

• Lab1 due today – please don’t forget to commit and push your solutions!

• Recitations from today will reinforce lecture materials with:
  • Review slides
  • Going over problem sets
  • Going over prior quizzes

• Please ask questions!
Dependence vs. hazard

• Dependence is a property of programs

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

- RAW
- WAR
- WAW

I1: FADD.D  f0, f0, 0
I2: FMUL.D  f3, f0, 3
I3: FSUB.D  f4, f0, 4
I4: FADD.D  f0, f5, 1
I5: FADD.D  f6, f6, f6
I6: FDIV.D  f0, f7, 1

- Reads/Writes to f0
  R (I1)
  W (I1)
  R (I2)
  R (I3)
  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)

\[
\text{I1: FADD.D } f_1, f_0, 1 \\
\text{I2: FADD.D } f_0, f_2, 1
\]

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**?
  • Expensive when there are many functional units!
  • Less beneficial where writeback stage is a small portion of the entire pipeline
  • Can still be incorporated to allow register read and write to happen in the same cycle
Questions?