CS 201 Lecture 18
Pipelines and Hazards

Spring 2014
Typical pipeline

Credit: Wikimedia Commons
Components

Pipeline buffers

Writeback
Benefits and Challenges

• Increased instruction throughput*
  – One instruction, by itself, might take longer, but in terms of instructions per clock cycle, this is faster

• Increased clock rates
  – The less you try to do per clock tick, the faster you can push the clock

• Requires that instructions do not have dependencies between them

• Dependencies are sometimes called “hazards”, and they come in several kinds
Hazards

• Structural Hazard: different stages might need simultaneous access to the same hardware units
  – Note that the example has separate memory for instructions and data

• Control Hazard: current instruction determines address of the next instruction

• Data Hazard: current instruction needs results from a previous instruction before that previous instruction is finished

© John A. Stratton, 2014
Resolving Hazards

• Stall or insert “bubbles”
  – Wait for the offending instruction to get through the pipeline
  – Defeats the purpose of pipelining, but often necessary

• Add extra hardware to work around the hazards
  – If the data exists somewhere down the pipeline but just hasn’t made it back yet

• Guess and Hope (speculation)
  – Execute a branch? Make a guess about which way it goes and start fetching instructions
  – Throw them away if you were wrong