CS184b: Computer Architecture [Single Threaded Architecture: abstractions, quantification, and optimizations]

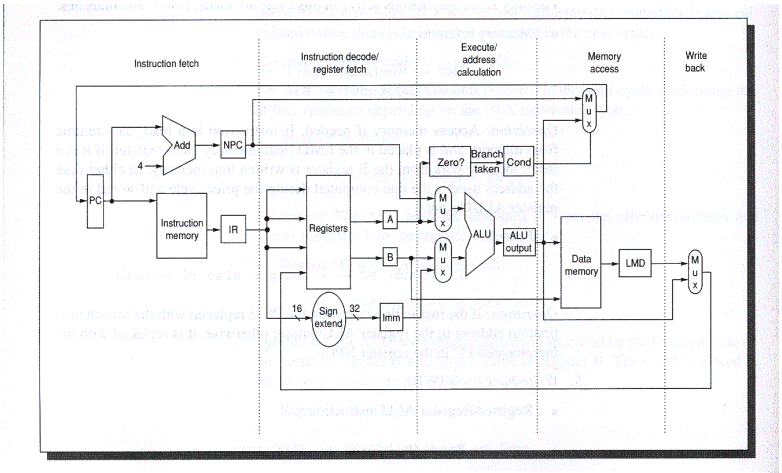
Day5: January 17, 2000

Pipelining ISA Processor Execution

Today

- Pipelined Processor Issue
  - Hazards
    - structural
    - data
    - control (next time)
  - accommodating
  - impact
- RISC/CISC wrapup (maybe)
- Admin question: this time good next term?

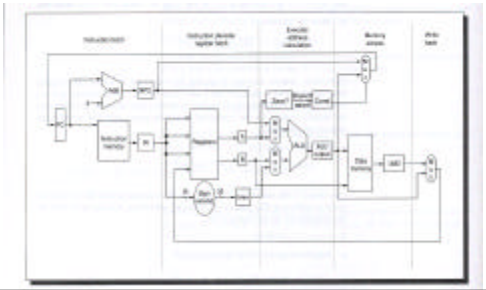
DLX Datapath



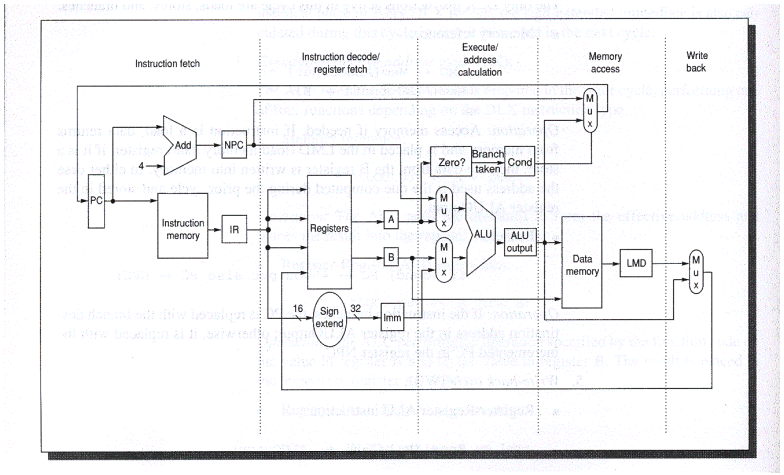
DLX unpipelined datapath from H&P (Fig. 3.1)

DLX Model Behavior

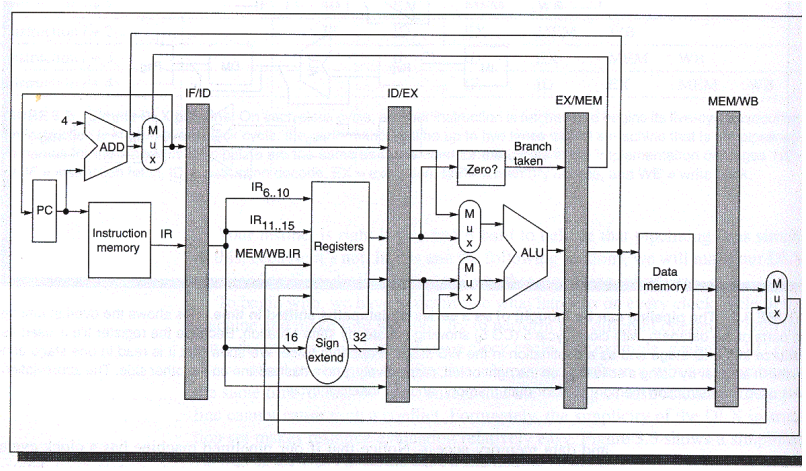
- Read Registers
- Perform primitive ALU Op
- Read/Write memory
- Write register result



Pipeline?



Pipeline DLX



DLX unpipelined datapath from H&P (Fig. 3.4)

Hazards

- Structural (resource bound)
- Data (value timing)
- Control (knowing where to go next)

Structural Hazards

- Arise when
  - Instructions have varying resource requirements
  - Usage of resources not occur at same time
  - Not want to provide resources for worst-case
    - typically because it's not the "common" case
    - performance impact small compared to cost of handling worst case

Structural Hazards

- Have to consider:
  - all possible overlaps of instructions
  - simplified by considering instruction classes
    - (e.g. add R1,R2,R3, sub R3,R4,R5, … all use same resource set…)

Structural Hazard: Identify

- Identify by:
  - looking at instruction (class) resource usage in pipeline
- Register-Register Op:
  - IF - I-mem port
  - ID - 2 register read ports
  - EXU - ALU
  - MEM - ---
  - WB - 1 register write port

Structural Hazard: Identify

- R-R: 1I 2RR 1A -- 1RW
- L/S: 1I 1RR 1A 1AB,1DB 1RW
- BR: 1I 1RR 1A -- --

- Conflicts:
  - standard DLX
  - RF has 1R, 1RW port

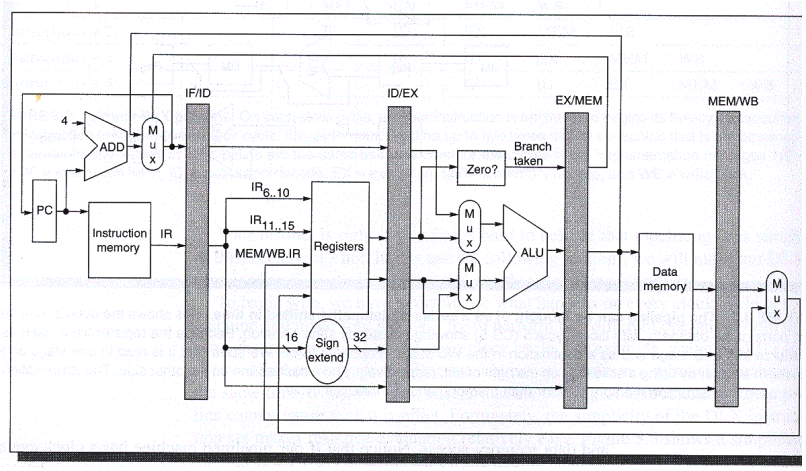
Structural Hazard: Identify

- Pipelined Memory access
- R-R: 1I 2RR 1A -- -- 1RW
- L: 1I 1RR 1A 1AB 1DB 1RW
- S: 1I 1RR 1A 1AB,1DB -- 1RW
- BR: 1I 1RR 1A -- --

Structural Hazards: Deal

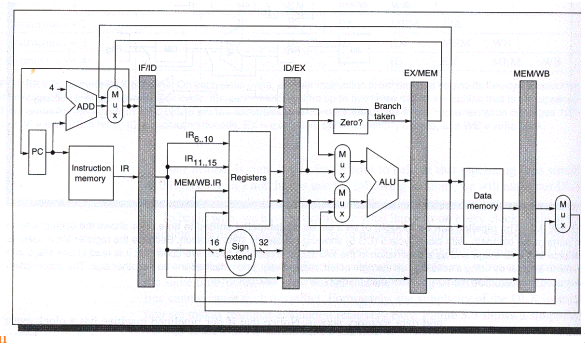
- Datapath cannot handle
- Always have the option of idling on a cycle
  - "Bubble" into pipeline
  - allow downstream continue, stall upstream
- Options:
  - detect when occurs and stall one instruction
  - detect will occur at issue and stall

Pipeline DLX



Data Hazards

- Pipeline Latency
- Instruction effects not completed before next operation begins

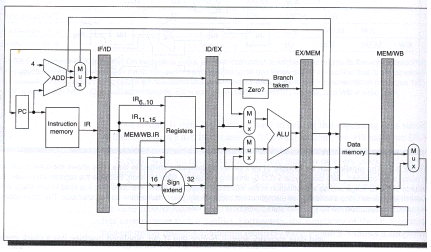


Data Hazard: Example

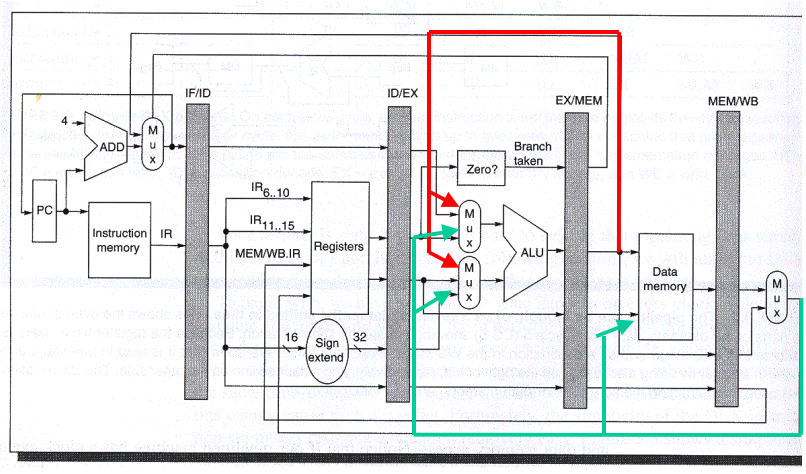
- ADD R1,R2,R3
- XOR R4,R1,R5

Data Hazard: Solving

- Primary problem (DLX)
  - data not back in RF
  - read stale data
- Partially solve with bypassing
  - when good data exist before use



Data Hazard: Solving

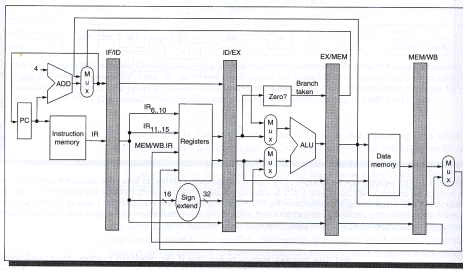


Data Hazard

- Note: since ops may stall, interrupt, resume
  - cannot decide how to set bypass muxes in ID stages
  - have to determine based on state of pipeline

Data Hazard

- Not all cases can bypass
  - if data not available anywhere, yet...
  - e.g.
    - LW R1,4(R2)
    - ADD R3,R1,R4



Model/Common Case

- Common/simple case
- Implementation transparency
- Could have slowed the initiation interval for all ops
- OR could have said can't use value for number of cycles
- But, only few sequences/cases problematic
  - let rest run faster

Types of Data Hazards

- RAW (example seen)
- WAW
  - order of writes transposed going to memory
  - leave wrong value in memory
- WAR
  - read gets value computed "after" it should have completed

Compiler

- Instruction Scheduling can try to avoid
  - (making some assumptions about implementation)
- Schedule instructions in delay slot
  - possible when have parallelism/independent tasks
- another example where optimize across larger block give tighter results

Big Ideas

- Preserve the (simple, stable) model
- While providing high-performance implementation

Big Ideas

- Ops with different requirements
- Some cases can run faster than others
- Fast in simple, common cases
- Correct in others