inst.eecs.berkeley.edu/~cs61c

CS61C : Machine Structures

Lecture 30 – Introduction to Pipelined Execution

2004-11-08

Lecturer PSOE Dan Garcia

www.cs.berkeley.edu/~ddgarcia

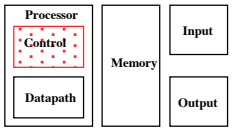
Pixar does it again! Our neighbors have a hit with The Incredibles!

Review: Single cycle datapath

° 5 steps to design a processor
- 1. Analyze instruction set => datapath requirements
- 2. Select set of datapath components & establish clock methodology
- 3. Assemble datapath meeting the requirements
- 4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer.
- 5. Assemble the control logic

° Control is the hard part

° MIPS makes that easier
- Instructions same size
- Source registers always in same place
- Immediates same size, location
- Operations always on registers/immediates



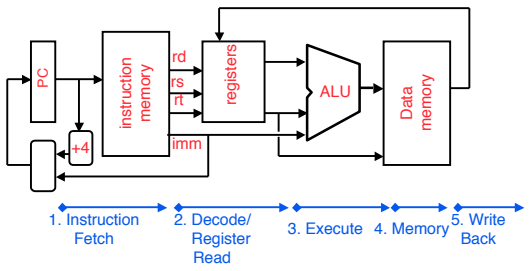
Review Datapath (1/3)

- Datapath is the hardware that performs operations necessary to execute programs.
- Control instructs datapath on what to do next.
- Datapath needs:
  - access to storage (general purpose registers and memory)
  - computational ability (ALU)
  - helper hardware (local registers and PC)

Review Datapath (2/3)

- Five stages of datapath (executing an instruction):
  1. Instruction Fetch (Increment PC)
  2. Instruction Decode (Read Registers)
  3. ALU (Computation)
  4. Memory Access
  5. Write to Registers
- ALL instructions must go through ALL five stages.

Review Datapath (3/3)



Gotta Do Laundry

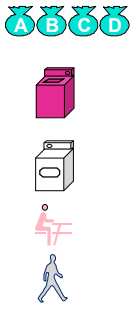
° Ann, Brian, Cathy, Dave each have one load of clothes to wash, dry, fold, and put away

° Washer takes 30 minutes

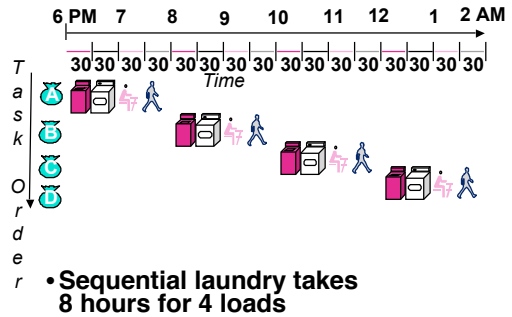
° Dryer takes 30 minutes

° "Folder" takes 30 minutes

° "Stasher" takes 30 minutes to put clothes into drawers

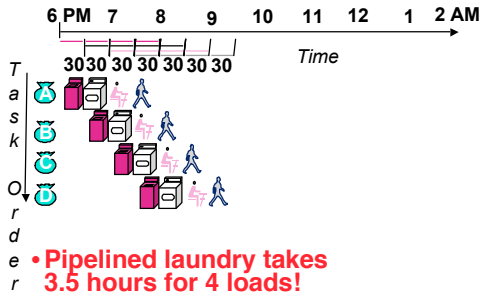


Sequential Laundry



- Sequential laundry takes 8 hours for 4 loads

Pipelined Laundry

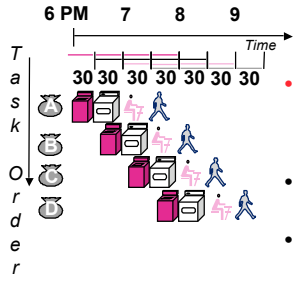


- Pipelined laundry takes 3.5 hours for 4 loads!

General Definitions

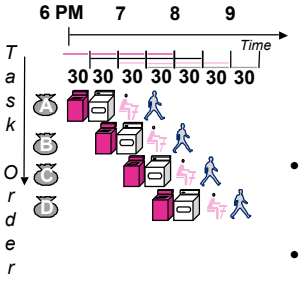
- Latency: time to completely execute a certain task
  - for example, time to read a sector from disk is disk access time or disk latency
- Throughput: amount of work that can be done over a period of time

Pipelining Lessons (1/2)



- Pipelining doesn't help latency of single task, it helps throughput of entire workload
- Multiple tasks operating simultaneously using different resources
- Potential speedup = Number pipe stages
- Time to "fill" pipeline and time to "drain" it reduces speedup: 2.3X v. 4X in this example

Pipelining Lessons (2/2)

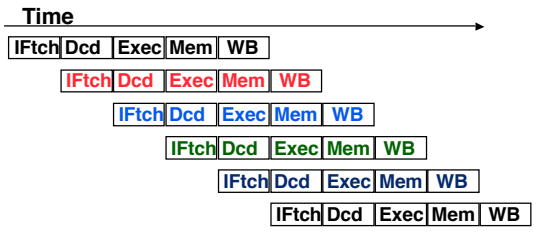


- Suppose new Washer takes 20 minutes, new Stasher takes 20 minutes. How much faster is pipeline?
- Pipeline rate limited by slowest pipeline stage
- Unbalanced lengths of pipe stages also reduces speedup

Steps in Executing MIPS

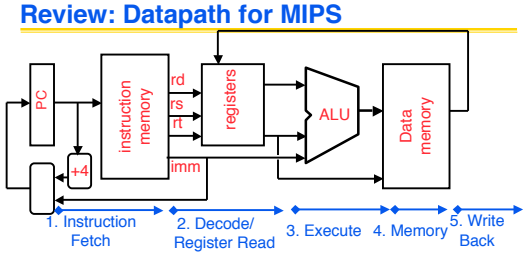
1) IFetch: Fetch Instruction, Increment PC
2) Decode Instruction, Read Registers
3) Execute:
   Mem-ref: Calculate Address
   Arith-log: Perform Operation
4) Memory:
   Load: Read Data from Memory
   Store: Write Data to Memory
5) Write Back: Write Data to Register

Pipelined Execution Representation

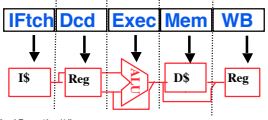


- Every instruction must take same number of steps, also called pipeline "stages", so some will go idle sometimes

Review: Datapath for MIPS

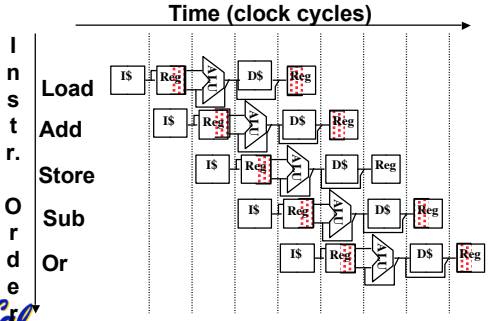


- Use datapath figure to represent pipeline



Graphical Pipeline Representation

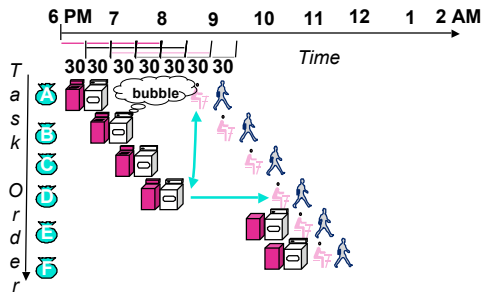
(In Reg, right half highlight read, left half write)



Example

- Suppose 2 ns for memory access, 2 ns for ALU operation, and 1 ns for register file read or write; compute instr rate
- Nonpipelined Execution:
  - lw : IF + Read Reg + ALU + Memory + Write Reg = 2 + 1 + 2 + 2 + 1 = 8 ns
  - add: IF + Read Reg + ALU + Write Reg = 2 + 1 + 2 + 1 = 6 ns
- Pipelined Execution:
  - Max(IF,Read Reg,ALU,Memory,Write Reg) = 2 ns

Pipeline Hazard: Matching socks in later load



A depends on D; stall since folder tied up

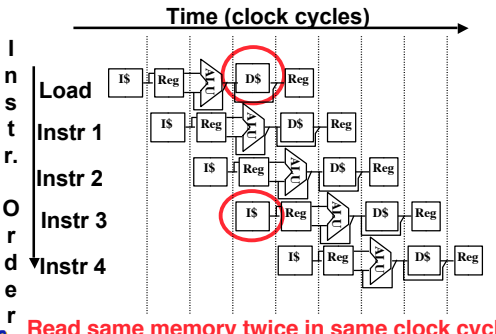
Administrivia

- Final Exam will be in 230 Hearst Gym
  - Tue, 2004-12-14, 12:30–3:30pm
- Thanks to Andrew for filling in on Fri!
- Cal still ranked in the top 5
  - Survived a scare on Sat
  - We're the top candidate for the Rose Bowl

Problems for Computers

- Limits to pipelining: Hazards prevent next instruction from executing during its designated clock cycle
  - Structural hazards: HW cannot support this combination of instructions (single person to fold and put clothes away)
  - Control hazards: Pipelining of branches & other instructions stall the pipeline until the hazard; "bubbles" in the pipeline
  - Data hazards: Instruction depends on result of prior instruction still in the pipeline (missing sock)

Structural Hazard #1: Single Memory (1/2)

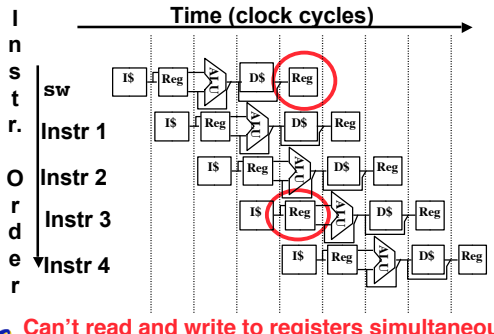


Read same memory twice in same clock cycle

Structural Hazard #1: Single Memory (2/2)

- Solution:
  - infeasible and inefficient to create second memory
  - (We'll learn about this more next week)
  - so simulate this by having two Level 1 Caches (a temporary smaller [of usually most recently used] copy of memory)
  - have both an L1 Instruction Cache and an L1 Data Cache
  - need more complex hardware to control when both caches miss

Structural Hazard #2: Registers (1/2)



Can't read and write to registers simultaneously

Structural Hazard #2: Registers (2/2)

- Fact: Register access is *VERY* fast: takes less than half the time of ALU stage
- Solution: introduce convention
  - always Write to Registers during first half of each clock cycle
  - always Read from Registers during second half of each clock cycle
  - Result: can perform Read and Write during same clock cycle

Things to Remember

- Optimal Pipeline
  - Each stage is executing part of an instruction each clock cycle.
  - One instruction finishes during each clock cycle.
  - On average, execute far more quickly.
- What makes this work?
  - Similarities between instructions allow us to use same stages for all instructions (generally).
  - Each stage takes about the same amount of time as all others: little wasted time.