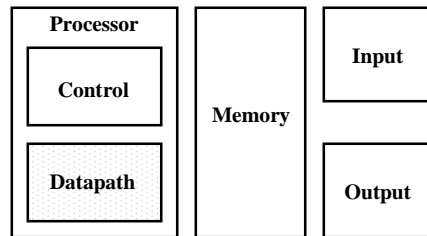


The Big Picture: Where are We Now?

- The Five Classic Components of a Computer



- Today's Topics:

- Pipelining by Analogy
- Introduction to MIPS pipelining

CS385, Spring-99

Copyright 1997 UCB

Pipelining is Natural!

Laundry Example

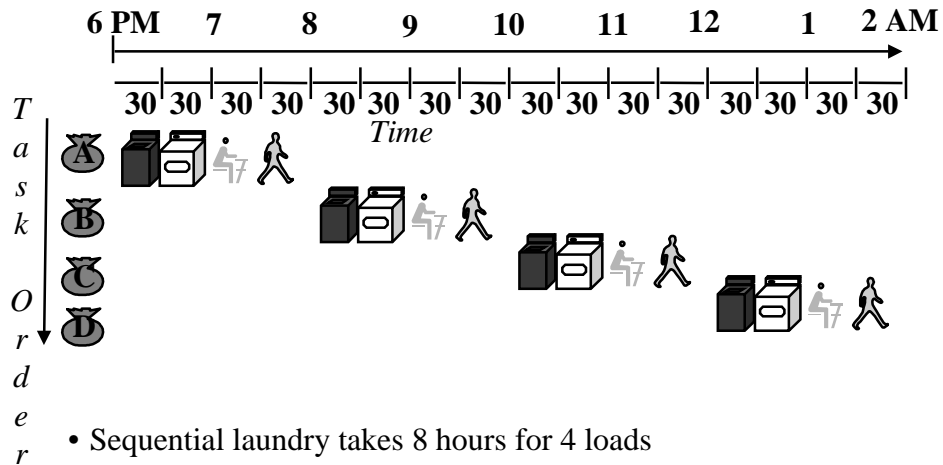
- Ann, Brian, Cathy, Dave each have one load of clothes to wash, dry, and fold
- Washer takes 30 minutes
- Dryer takes 30 minutes
- “Folder” takes 30 minutes
- “Stasher” takes 30 minutes to put clothes into drawers



CS385, Spring-99

Copyright 1997 UCB

Sequential Laundry

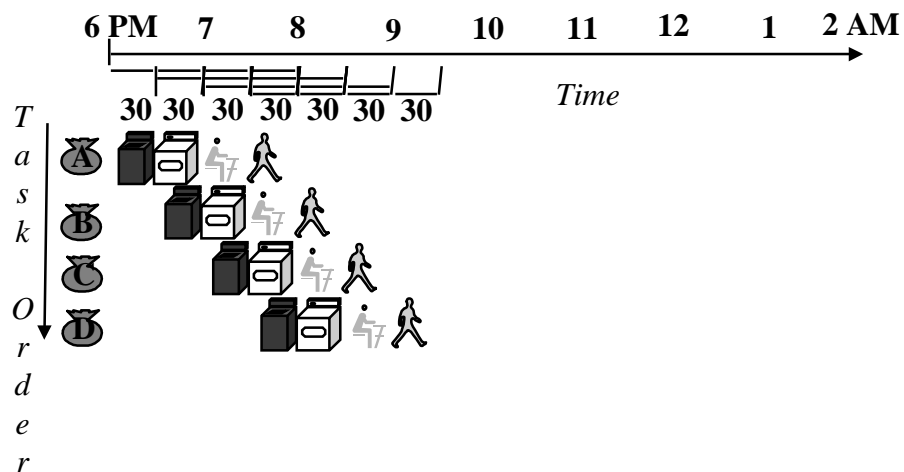


- Sequential laundry takes 8 hours for 4 loads
- If they learned pipelining, how long would laundry take?

CS385, Spring-99

Copyright 1997 UCB

Pipelined Laundry: Start work ASAP

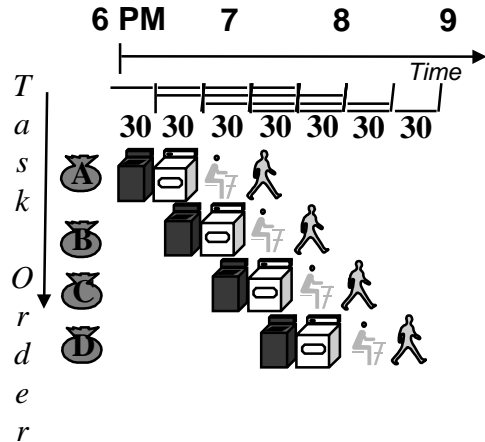


- Pipelined laundry takes 3.5 hours for 4 loads!

CS385, Spring-99

Copyright 1997 UCB

Pipelining Lessons

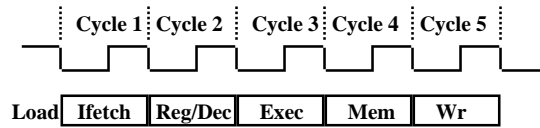


- Pipelining doesn't help latency of single task, it helps throughput of entire workload
- Multiple tasks operating simultaneously using different resources
- Potential speedup = Number of pipe stages
- Pipeline rate limited by slowest pipeline stage
- Unbalanced lengths of pipe stages reduces speedup
- Time to "fill" pipeline and time to "drain" it reduces speedup
- Stall for Dependences

CS385, Spring-99

Copyright 1997 UCB

The Five Stages of Load



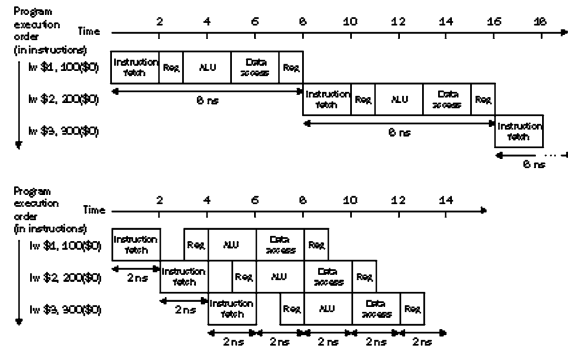
- Ifetch: Instruction Fetch
 - Fetch the instruction from the Instruction Memory
- Reg/Dec: Registers Fetch and Instruction Decode
- Exec: Calculate the memory address
- Mem: Read the data from the Data Memory
- Wr: Write the data back to the register file

CS385, Spring-99

Copyright 1997 UCB

Pipelining

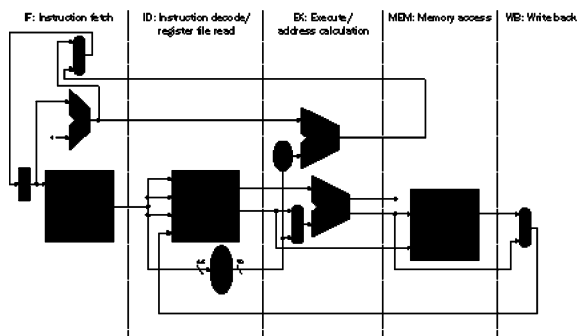
- Improve performance by increasing instruction throughput



CS385, Spring-99

Copyright 1997 UCB

Basic Idea

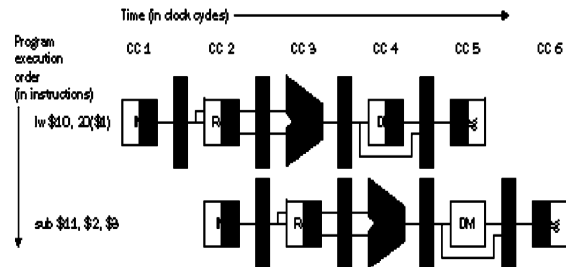


- *What do we need to add to actually split the datapath into stages?*

CS385, Spring-99

Copyright 1997 UCB

Graphically Representing Pipelines

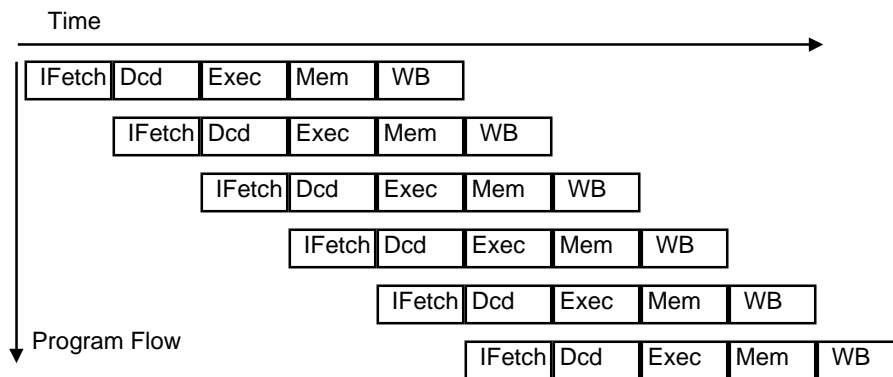


- Can help with answering questions like:
 - how many cycles does it take to execute this code?
 - what is the ALU doing during cycle 4?
 - use this representation to help understand datapaths

CS385, Spring-99

Copyright 1997 UCB

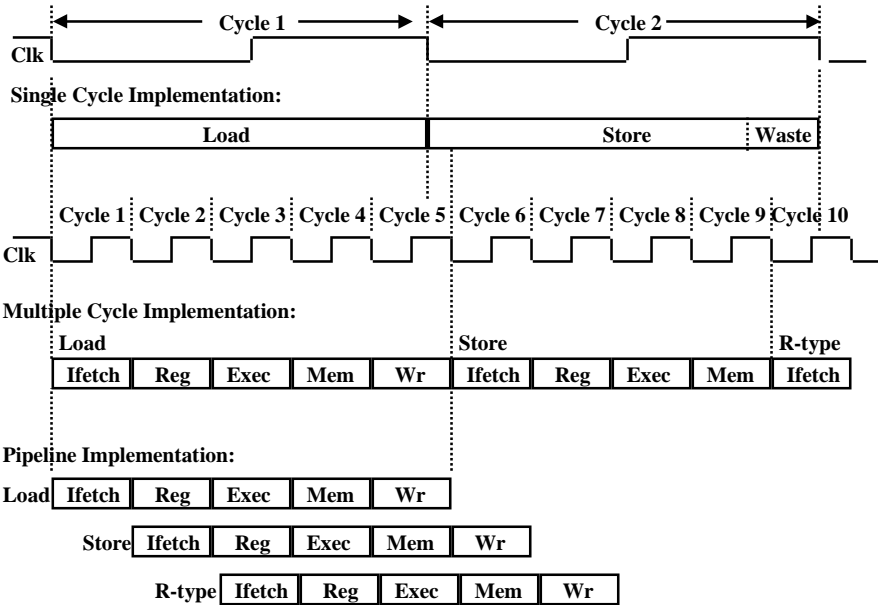
Conventional Pipelined Execution Representation



CS385, Spring-99

Copyright 1997 UCB

Single Cycle, Multiple Cycle, vs. Pipeline



CS385, Spring-99

Copyright 1997 UCB

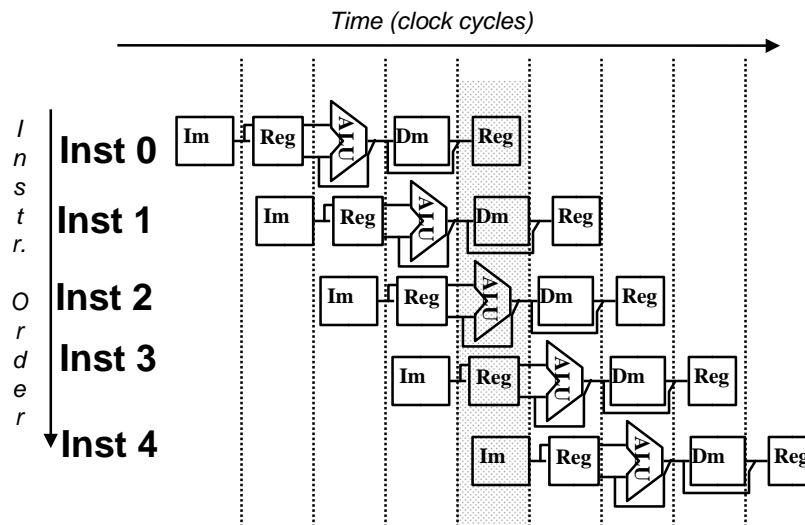
Why Pipeline?

- Suppose we execute 100 instructions
- Single Cycle Machine
 - 45 ns/cycle x 1 CPI x 100 inst = 4500 ns
- Multicycle Machine
 - 10 ns/cycle x 4.6 CPI (due to inst mix) x 100 inst = 4600 ns
- Ideal pipelined machine
 - 10 ns/cycle x (1 CPI x 100 inst + 4 cycle drain) = 1040 ns

CS385, Spring-99

Copyright 1997 UCB

Why Pipeline? Because the resources are there!



CS385, Spring-99

Copyright 1997 UCB

Can pipelining get us into trouble?

- Yes: Pipeline Hazards
 - structural hazards: attempt to use the same resource two different ways at the same time
 - E.g., combined washer/dryer would be a structural hazard or folder busy doing something else (watching TV)
 - data hazards: attempt to use item before it is ready
 - E.g., one sock of pair in dryer and one in washer; can't fold until get sock from washer through dryer
 - instruction depends on result of prior instruction still in the pipeline
 - control hazards: attempt to make a decision before condition is evaluated
 - E.g., washing football uniforms and need to get proper detergent level; need to see after dryer before next load in
 - branch instructions
- Can always resolve hazards by waiting
 - pipeline control must detect the hazard
 - take action (or delay action) to resolve hazards

CS385, Spring-99

Copyright 1997 UCB