

# Pipelining

Notes based on Patterson & Hennessey  
Chapter 6

Adapted from the lecture notes of Dr. John Kubiatowicz (UC Berkeley)





# Reading Assignment

- Heuring, Chapter 5  
– 5.1 – 5.3

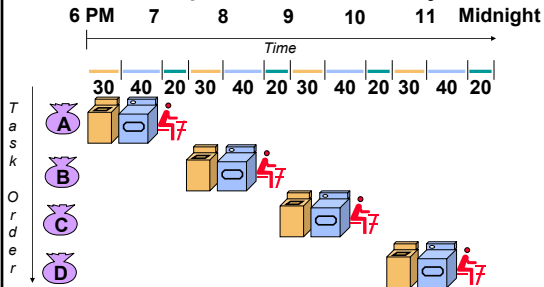
# Topics

- Pipelining
  - Advantages
  - Terminology
    - stages
    - pipeline hazards
      - structural
      - data
      - control
  - Performance comparison to non-pipelined

# Pipelining Analogy

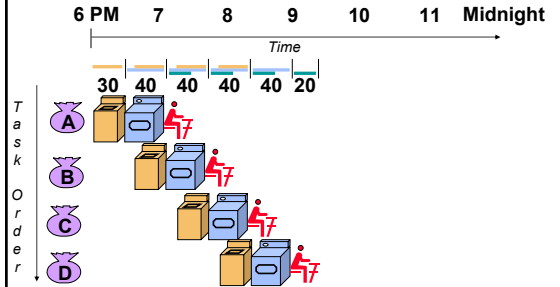
- Laundry
- Ann, Brian, Cathy, Dave  each have one load of clothes to wash, dry, and fold
- Washer takes 30 minutes 
- Dryer takes 40 minutes 
- Folding takes 20 minutes 

# Sequential Laundry



- Sequential laundry takes 6 hours for 4 loads

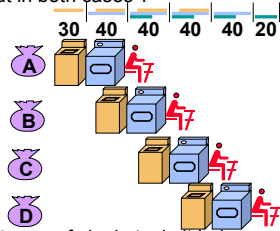
# Pipelined Laundry: Start work ASAP



- Pipelined laundry takes 3.5 hours for 4 loads

## Pipelining

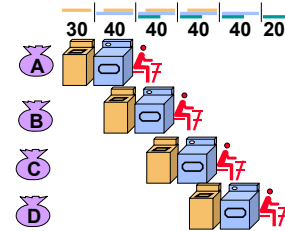
- Latency vs. Throughput
- Question
  - What is the latency in both cases ?
  - What is the throughput in both cases ?



Pipelining doesn't help **latency** of single task, it helps **throughput** of entire workload

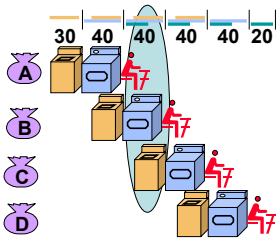
## Pipelining [contd...]

- Question
  - What is the fastest operation in the example ?
  - What is the slowest operation in the example



Pipeline rate limited by **slowest** pipeline stage

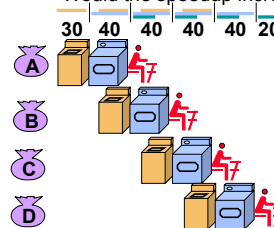
## Pipelining [contd...]



**Multiple** tasks operating **simultaneously** using different resources




## Pipelining Lessons [contd...]

- Question
  - Would the speedup increase if we had more steps ?



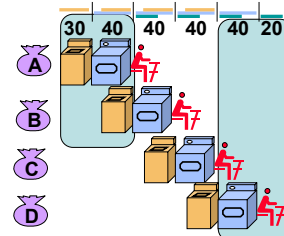
Potential Speedup = Number of pipe stages

## Pipelining [contd...]

- Washer takes 30 minutes 
- Dryer takes 40 minutes 
- Folding takes 20 minutes 
- Question
  - What affect is there if folding also took 40 minutes?

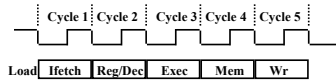
Unbalanced lengths of pipe stages reduces speedup

## Pipelining [contd...]



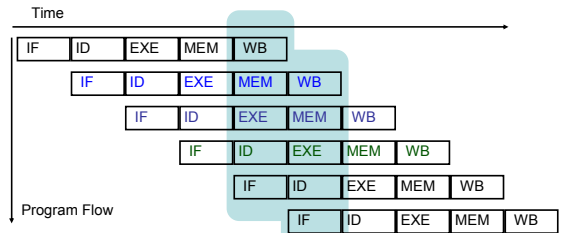
Time to "fill" pipeline and time to "drain" it reduces speedup

## Five Stages of an Instruction



- Instruction Fetch
  - Fetch the instruction from the Instruction Memory
- Operand Fetch and Instruction Decode
- Execute:
  - Calculate the memory address
- Memory:
  - Read the data from the Data Memory
- Writeback:
  - Write the data back to the register file

## Conventional Pipelined Execution Representation



## Example

IFetch: 200 ps  
 Register Read: 100 ps  
 ALU: 200 ps  
 Memory: 200 ps  
 Register Write: 100 ps



## Example [contd...]

- $Time_{\text{pipeline}} = Time_{\text{non-pipeline}} / \text{Pipe stages}$ 
  - Assumptions
    - Stages are perfectly balanced
    - Ideal conditions

## Definitions

- Performance is in units of things per sec
  - bigger is better
- If we are primarily concerned with response time

–  $performance(x) = \frac{1}{\text{execution\_time}(x)}$

"X is n times faster than Y" means

$$n = \frac{\text{Performance}(X)}{\text{Performance}(Y)} = \frac{\text{Execution\_time}(Y)}{\text{Execution\_time}(X)}$$

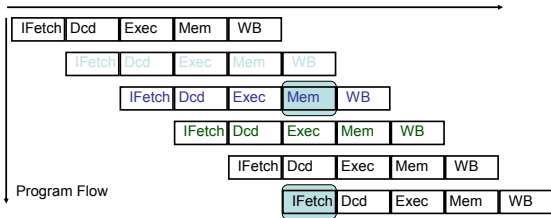
## Example [contd...]

- Speedup in this case =  $24/14 = 1.7$
- Lets add 1000 more instructions
  - Time (non-pipelined) =  $1000 \times 8 + 24 \text{ ns} = 8024 \text{ ns}$
  - Time (pipelined) =  $1000 \times 2 + 14 \text{ ns} = 2014 \text{ ns}$
  - Speedup =  $8024 / 2014 = 3.98 \approx 4$  (approx) =  $8/2$

Instruction throughput is important metric (as opposed to individual instruction) as real programs execute billions of instructions in practical case !!!

## Pipeline Hazards

- Structural Hazard



## Pipeline Hazard [contd...]

- Control Hazard

- Example

- add \$4, \$5, \$6
- beq \$1, \$2, 40
- lw \$3, 300(\$0)

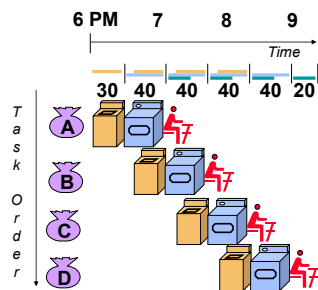
## Pipeline Hazards [contd...]

- Data Hazards

- Example

- add \$s0, \$t0, \$t1
- sub \$t2, \$s0, \$t3

## Summary Pipelining Lessons



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

## Summary of Pipeline Hazards

- Structural Hazards

- Hardware design

- Control Hazard

- Decision based on results

- Data Hazard

- Data Dependency