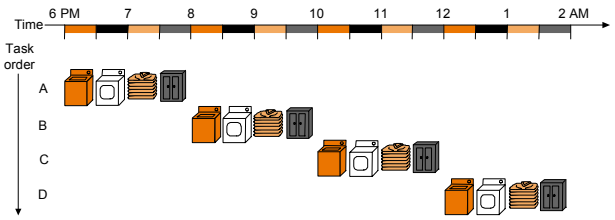
18-447 Lecture 10: Pipelined Implementations

James C. Hoe
Dept of ECE, CMU
February 23, 2009

Announcements: Project 1 is due this week
Midterm graded, results posted

Handouts: H09 Homework 3 (on Blackboard)
Graded Midterms
Midterm solutions with statstics

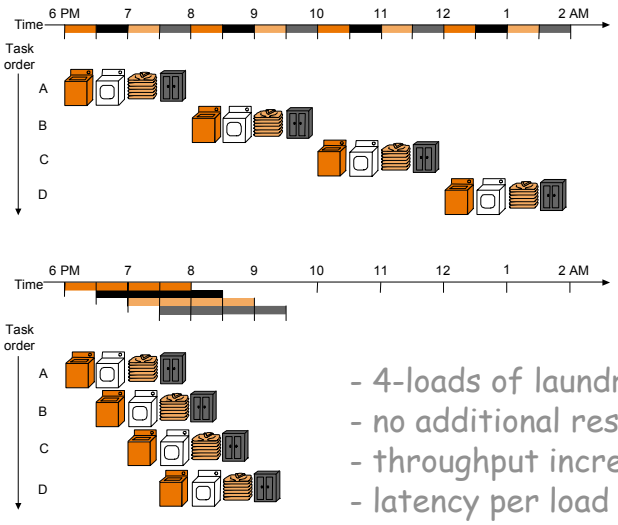
Doing laundry more quickly: in theory



- "place one dirty load of clothes in the washer"
- "when the washer is finished, place the wet load in the dryer"
- "when the dryer is finished, place the dry load on a table and fold"
- "when folding is finished, ask your roommate (??) to put the clothes away"

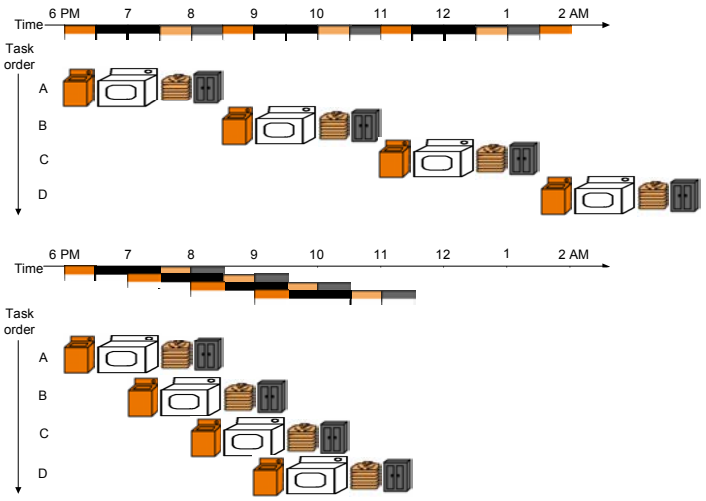
- steps to do a load are sequentially dependent
- no dependence between different loads

Doing laundry more quickly: in theory



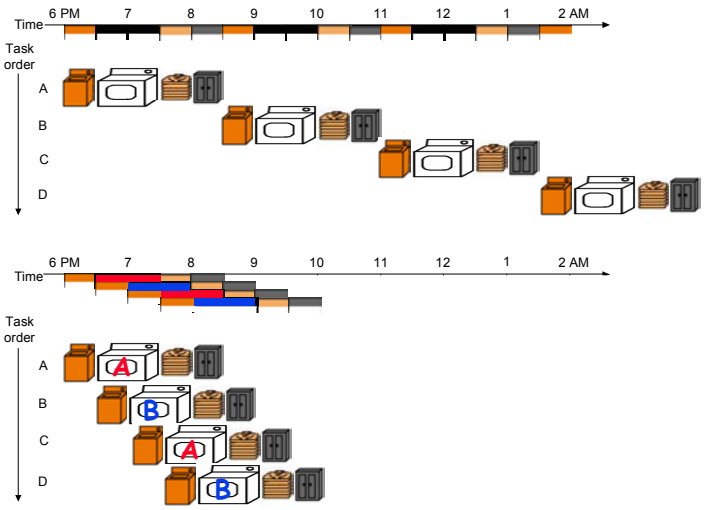
- 4-loads of laundry in parallel
- no additional resources
- throughput increased by 4
- latency per load is the same

Doing laundry more quickly: in practice



the slowest step decides throughput

Doing laundry more quickly: in practice



Throughput restored (2 loads per hour) using 2 dryers

Based on figures from [P&H CO&D, COPYRIGHT 2004 Elsevier. ALL RIGHTS RESERVED.]

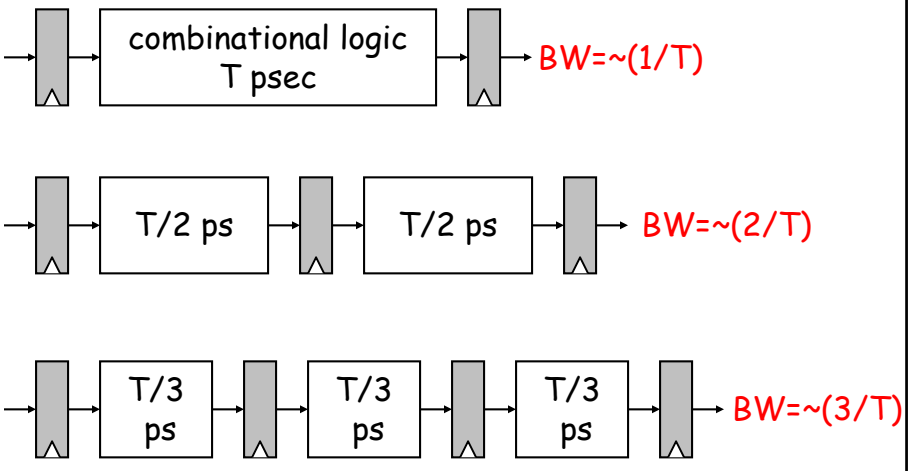
Pipeline Idealism

Motivation: Increase throughput with little increase in hardware

- Repetition of identical operations
  The same operation is repeated on a large number of different inputs
- Repetition of independent operations
  No ordering dependencies between repeated operations
- Uniformly partitionable suboperations
  Can be evenly divided into uniform-latency suboperations (that do not share resources)

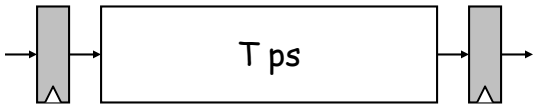
Good examples: automobile assembly line, doing laundry, but instruction pipeline???

Ideal Pipelining

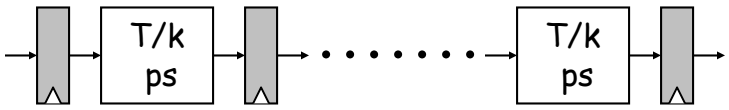


Performance Model

- Nonpipelined version with delay T
  BW = 1/(T+S) where S = latch delay



- k-stage pipelined version
  $BW_{k\text{-stage}}$ = 1 / (T/k +S)
  BW_{max} = 1 / (1 gate delay + S)



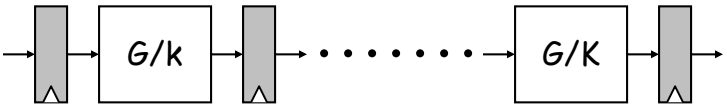
Cost Model

- Nonpipelined version with combinational cost G

  Cost = G+L where L = latch cost

- k-stage pipelined version

  $Cost_{k\text{-}stage} = G + Lk$



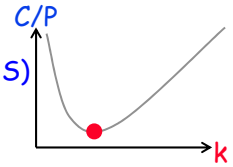
Cost/Performance Trade-off

[Peter M. Kogge, 1981]

Cost/Performance:

C/P = [Lk + G] / [1/(T/k + S)] = (Lk + G) (T/k + S)

= LT + GS + LSk + GT/k



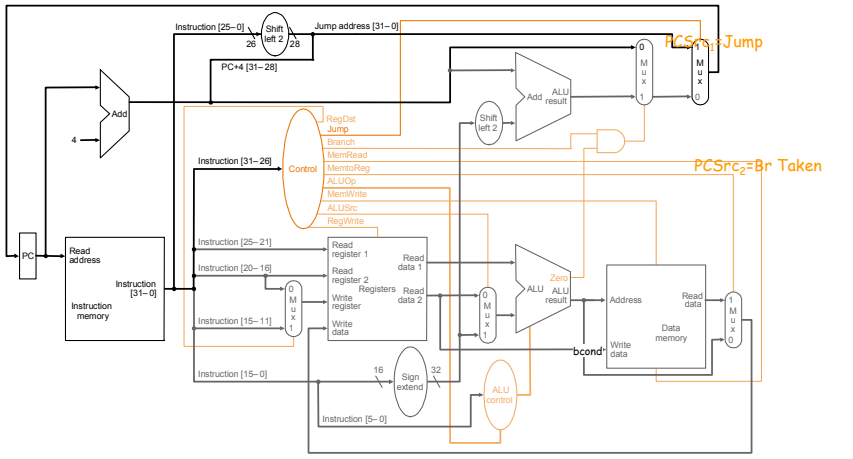
Optimal Cost/Performance: find min. C/P w.r.t. choice of k

$$\frac{d}{dk}\left(\frac{Lk+G}{\frac{1}{\frac{T}{k}+S}}\right) = 0 + 0 + LS - \frac{GT}{k^2}$$

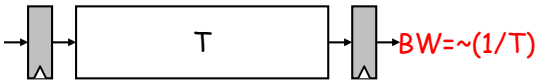
$$LS - \frac{GT}{k^2} = 0$$

$$k_{opt} = \sqrt{\frac{GT}{LS}}$$

The Reality of Pipelining Instruction Execution . . . .

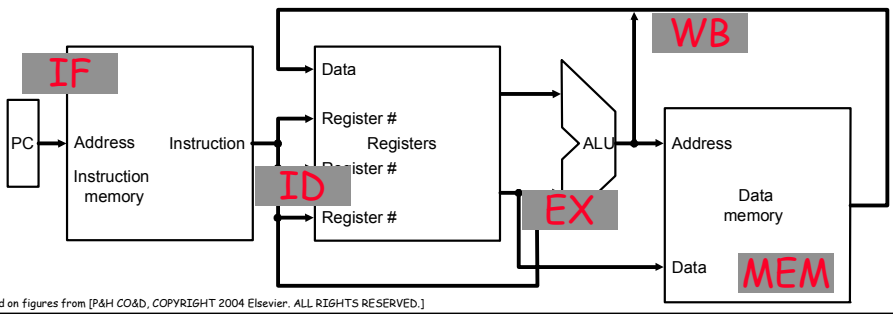


ALU operation

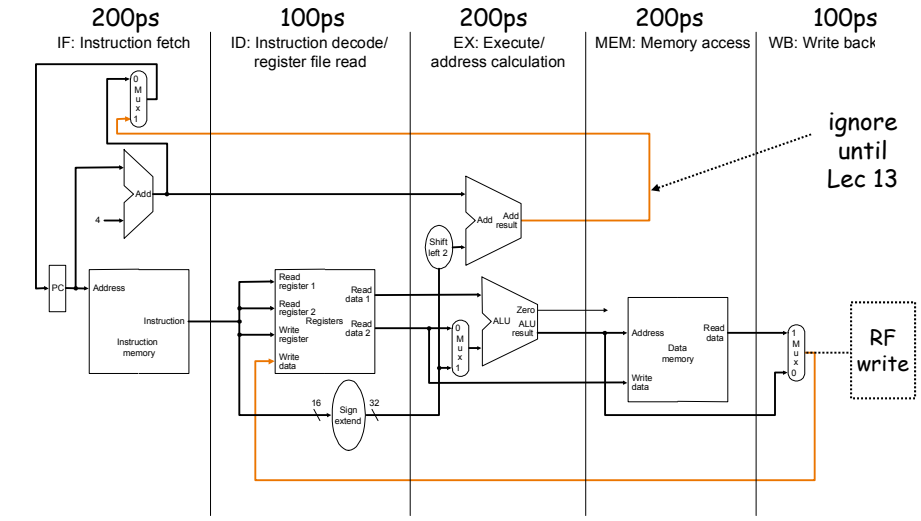


RISC Instruction Processing

- 5 generic steps
  - instruction fetch
  - instruction decode and operand fetch
  - ALU/execute
  - memory access (not required by non-mem instructions)
  - write-back



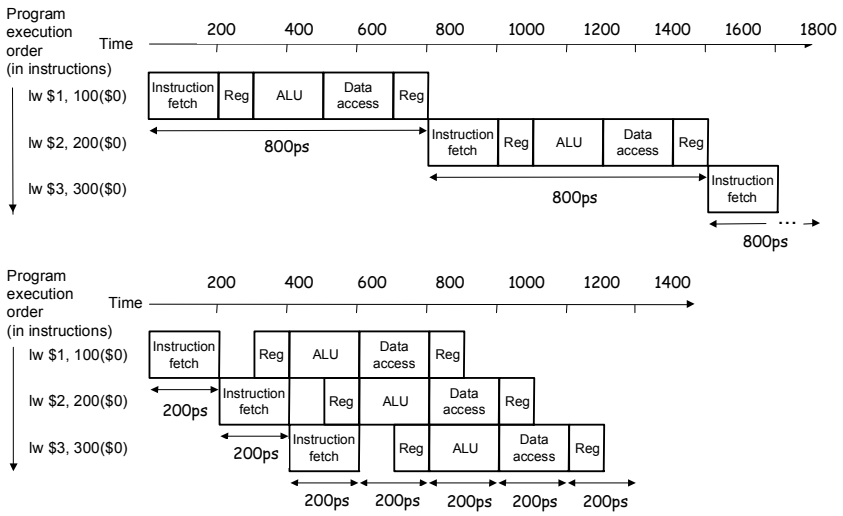
Dividing into Stages



Is this the correct partitioning?

Why not 4 or 6 stages? Why not different boundaries

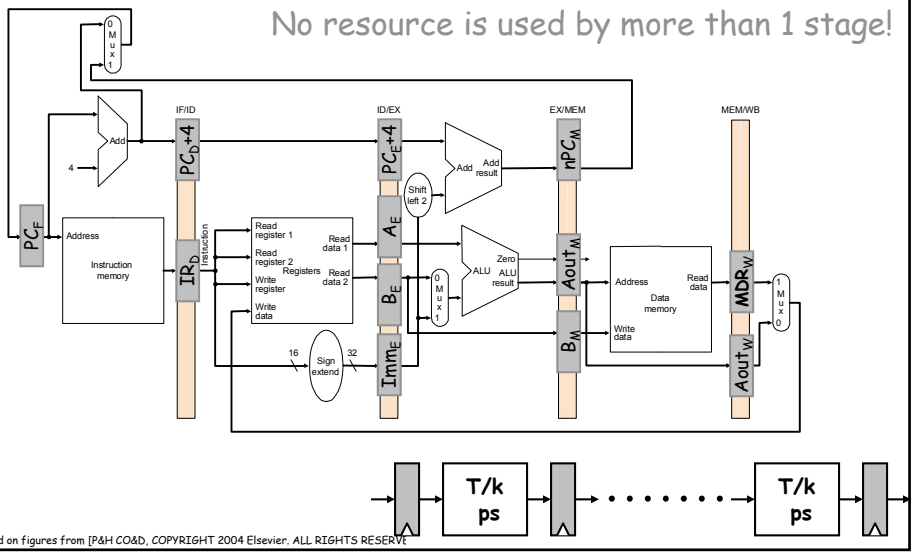
Pipelining



5-stage speedup is 4, not 5 as predicated by the ideal model

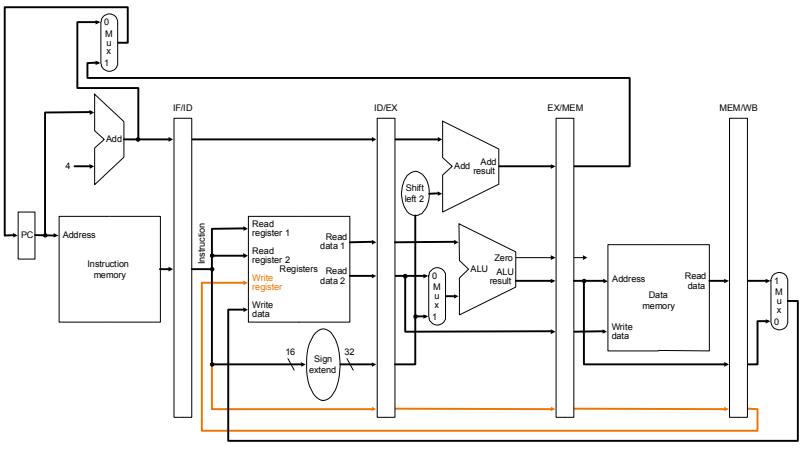
Pipeline Registers

No resource is used by more than 1 stage!



Based on figures from [P&H CO&D, COPYRIGHT 2004 Elsevier. ALL RIGHTS RESERVE

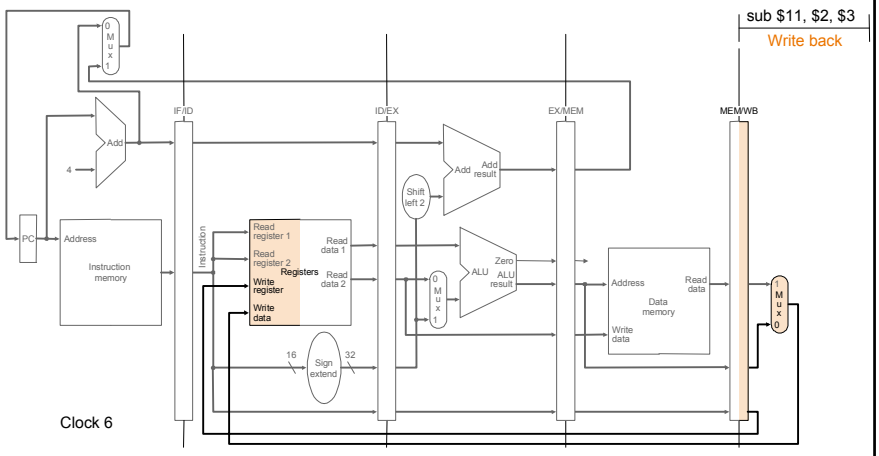
Pipelined Operation



All instruction classes must follow the same path and timing through the pipeline stages. Any performance impact?

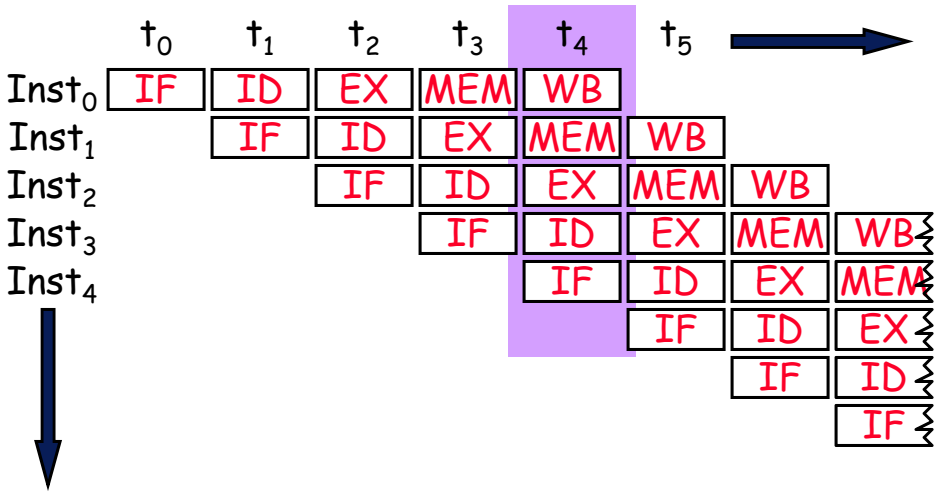
Based on figures from [P&H CO&D, COPYRIGHT 2004 Elsevier. ALL RIGHTS RESERVED.]

Pipelined Operation



Based on figures from [P&H CO&D, COPYRIGHT 2004 Elsevier. ALL RIGHTS RESERVED.]

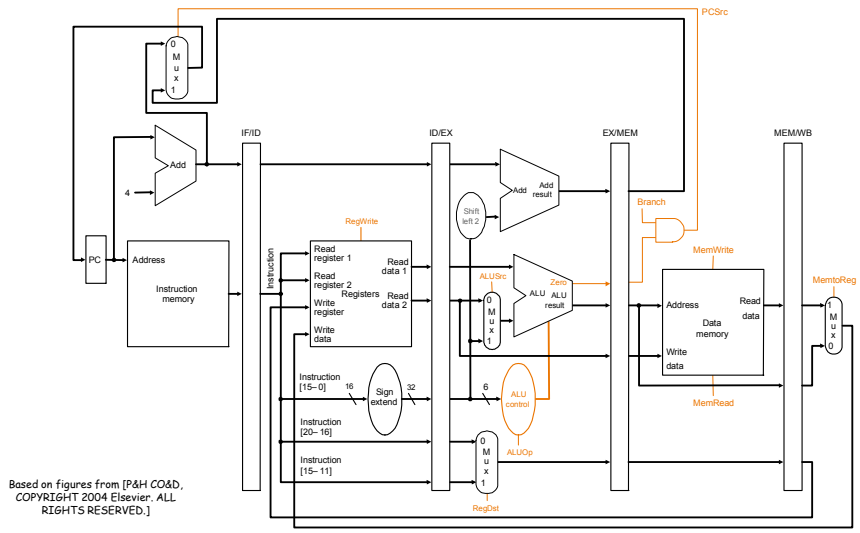
Illustrating Pipeline Operation: Operation View



Illustrating Pipeline Operation: Resource View

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
IF	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}
ID		I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9
EX			I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8
MEM				I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
WB					I_0	I_1	I_2	I_3	I_4	I_5	I_6

Control Points



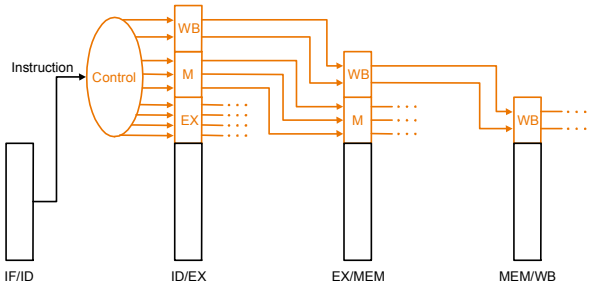
Based on figures from [P&H CO&D, COPYRIGHT 2004 Elsevier. ALL RIGHTS RESERVED.]

Identical set of control points as the single-cycle datapath!!

Sequential Control: Special Case

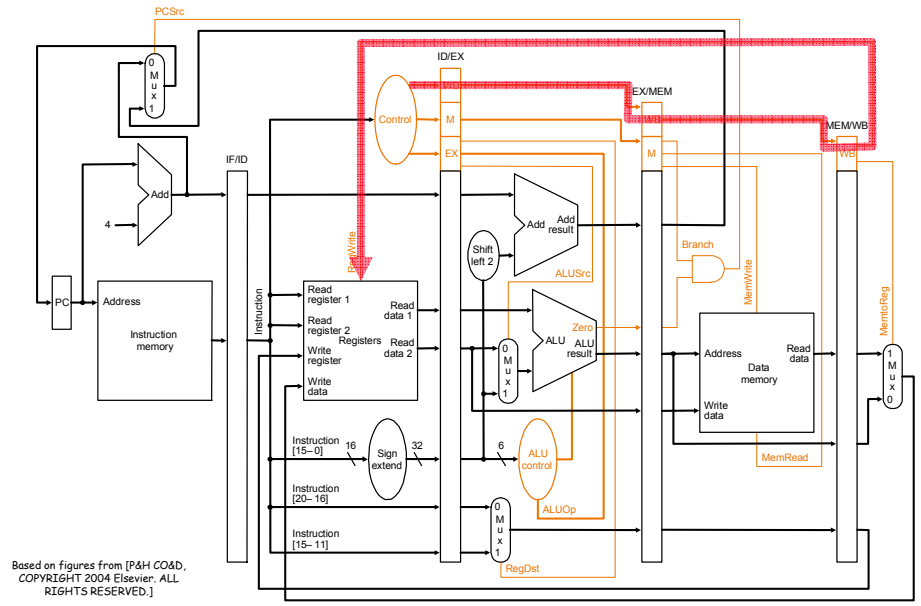
- For a given instruction
  - same control settings as single-cycle, but
  - control signals required at different cycles, depending on stage

  ⇒ decode once using the same logic as single-cycle and buffer control signals until consumed



⇒ or carry relevant "instruction word/field" down the pipeline and decode locally within each stage (still same logic)

Pipelined Control



Based on figures from [P&H CO&D, COPYRIGHT 2004 Elsevier. ALL RIGHTS RESERVED.]

Instruction Pipeline Reality

- Identical operations ... NOT!
  - ⇒ unifying instruction types
    - coalescing instruction types into one "multi-function" pipe
    - external fragmentation (some idle stages)
- Uniform Suboperations ... NOT!
  - ⇒ balance pipeline stages
    - stage quantization to yield balanced stages
    - internal fragmentation (some too-fast stages)
- Independent operations ... NOT!
  - ⇒ resolve data and resource hazards
    - duplicate contended resources
    - inter-instruction dependency detection and resolution

MIPS ISA features are engineered for improved pipelineability

How to do better on Midterm 2

- Read the text before lecture
- Pay attention in lecture
- Do homework
- Do lab
- Ask questions in lecture, lab, office hour
- Eat breakfast every morning
- Don't run with scissors

. . . . . Do you really need me to tell you any of these?