Digital System Design

by

Dr. Lesley Shannon

Email: lshannon@ensc.sfu.ca

Course Website: http://www.ensc.sfu.ca/~lshannon/courses/ensc350

Simon Fraser University

Slide Set: 14

Date: March 23, 2009

Introduction to Slide Set 16

In this slide set, we will talk about Pipelining and give another datapath example.
To make it a bit more concrete, I will first present a commercial embedded processor called MIPS, and then show how MIPS can be pipelined.

Note: pretty much any datapath can be pipelined in this way, and it is one of the key ways to improve throughput of a system.

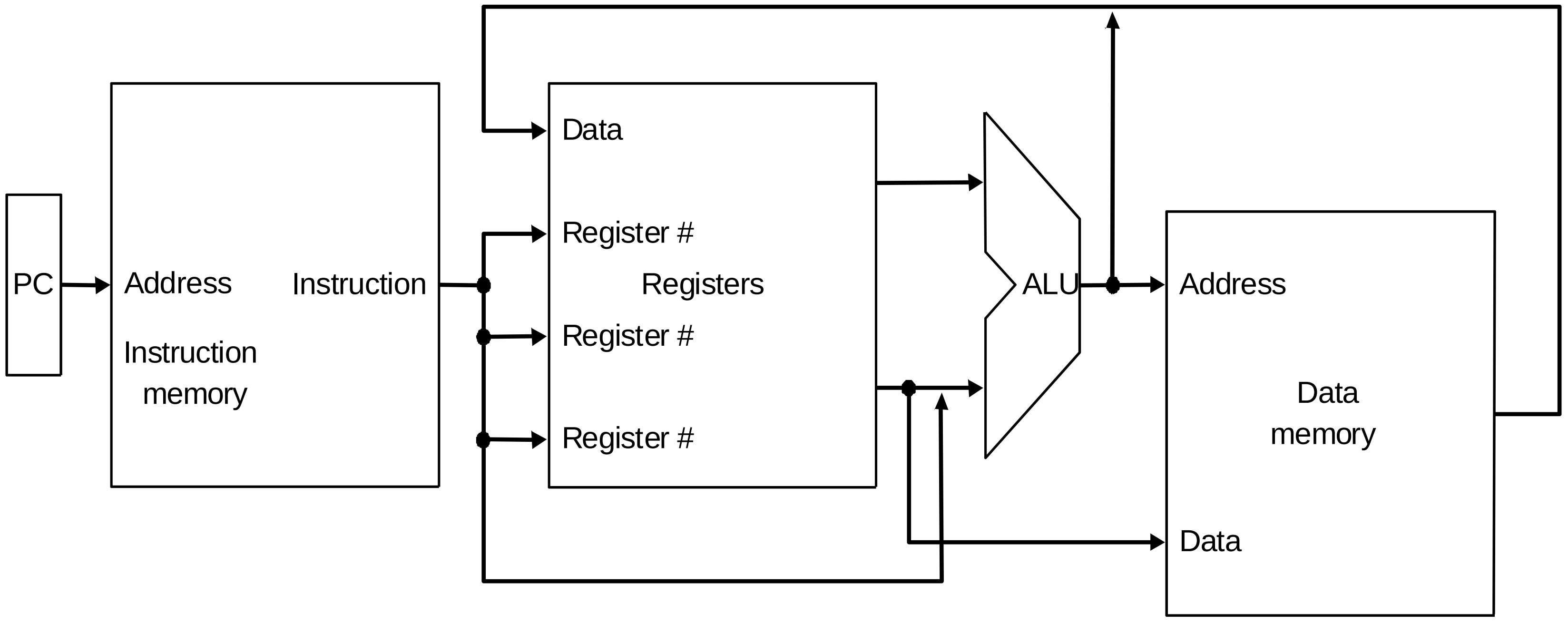
The MIPS Processor:

- On the next few slides, I'll show you the datapath of MIPS
  - It should be a review of concepts from 250 applying what you've learned in 350

- Simplified to contain only:
  - memory-reference instructions: `lw, sw`
  - arithmetic-logical instructions: `add, sub, and, or, slt`
  - control flow instructions: `beq, j`
- Generic Implementation:
  - use the program counter (PC) to supply instruction address
  - get the instruction from memory
  - read registers
  - use the instruction to decide exactly what to do

More Implementation Details

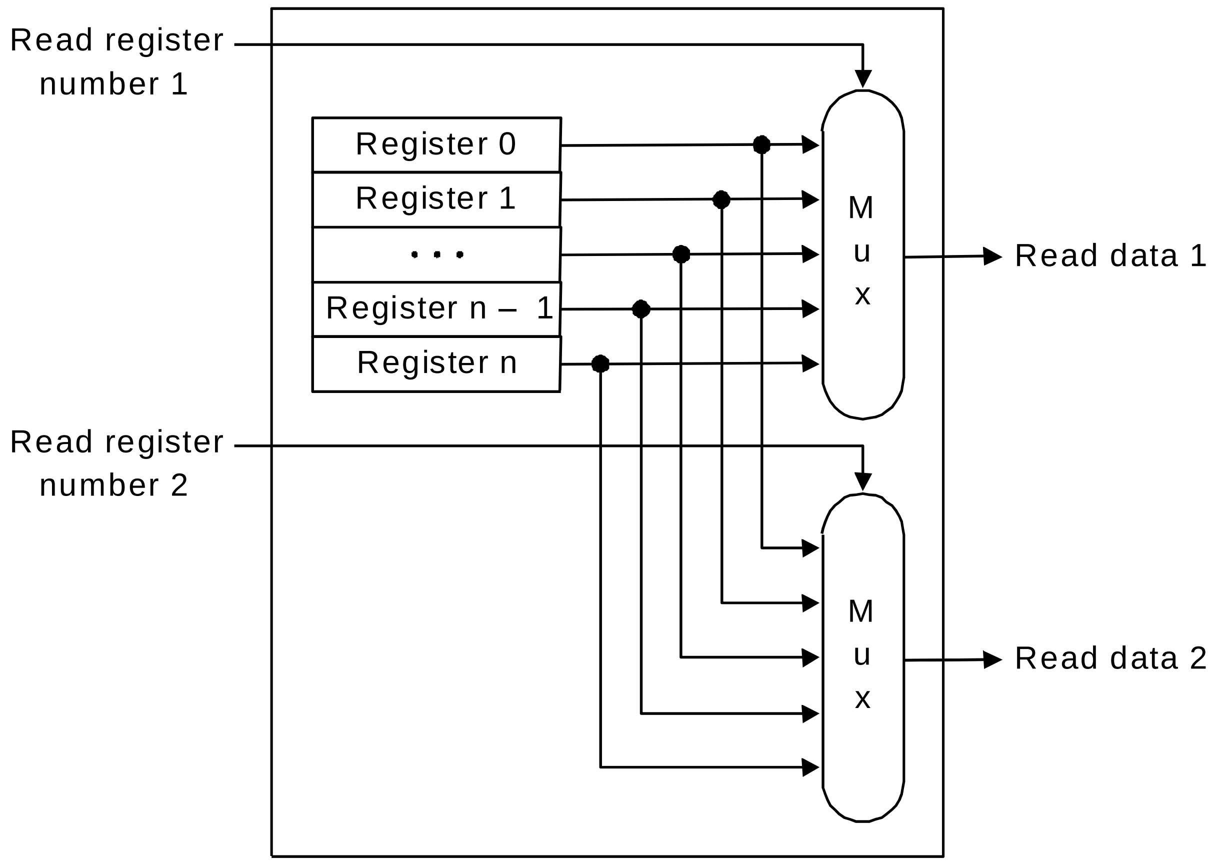
Simplified View:



Two types of functional units:

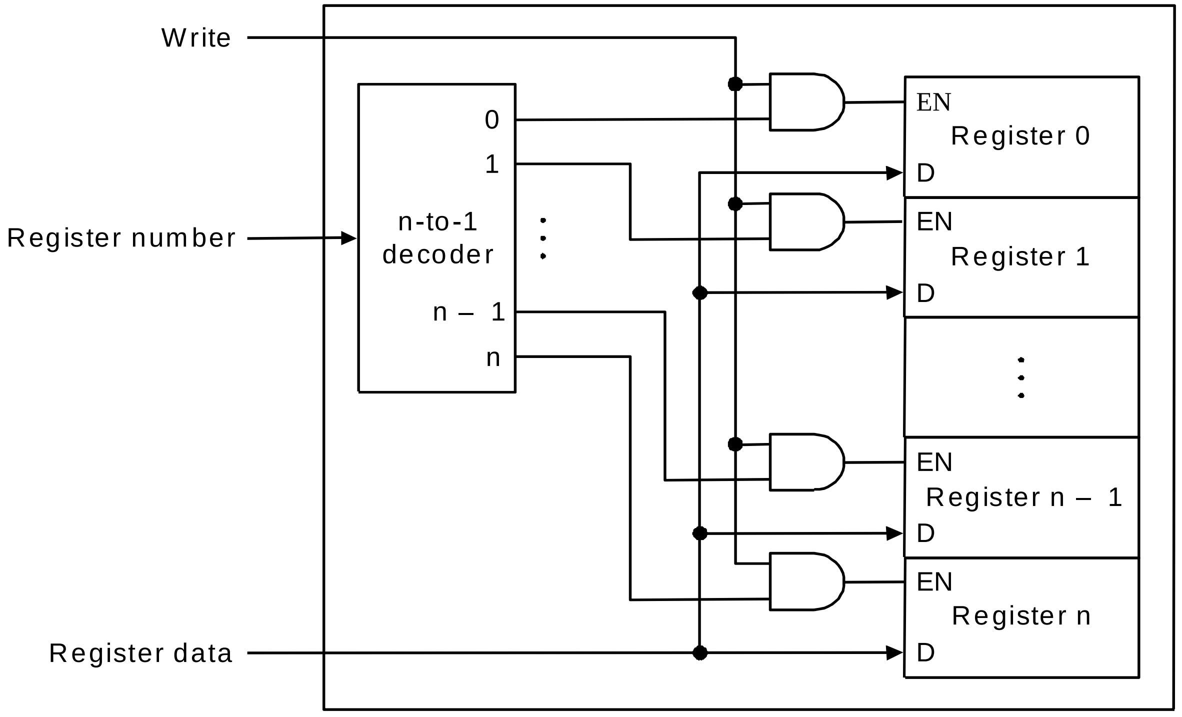
- **elements that operate on data values (combinational)**
- **elements that contain state (sequential)**

Register File



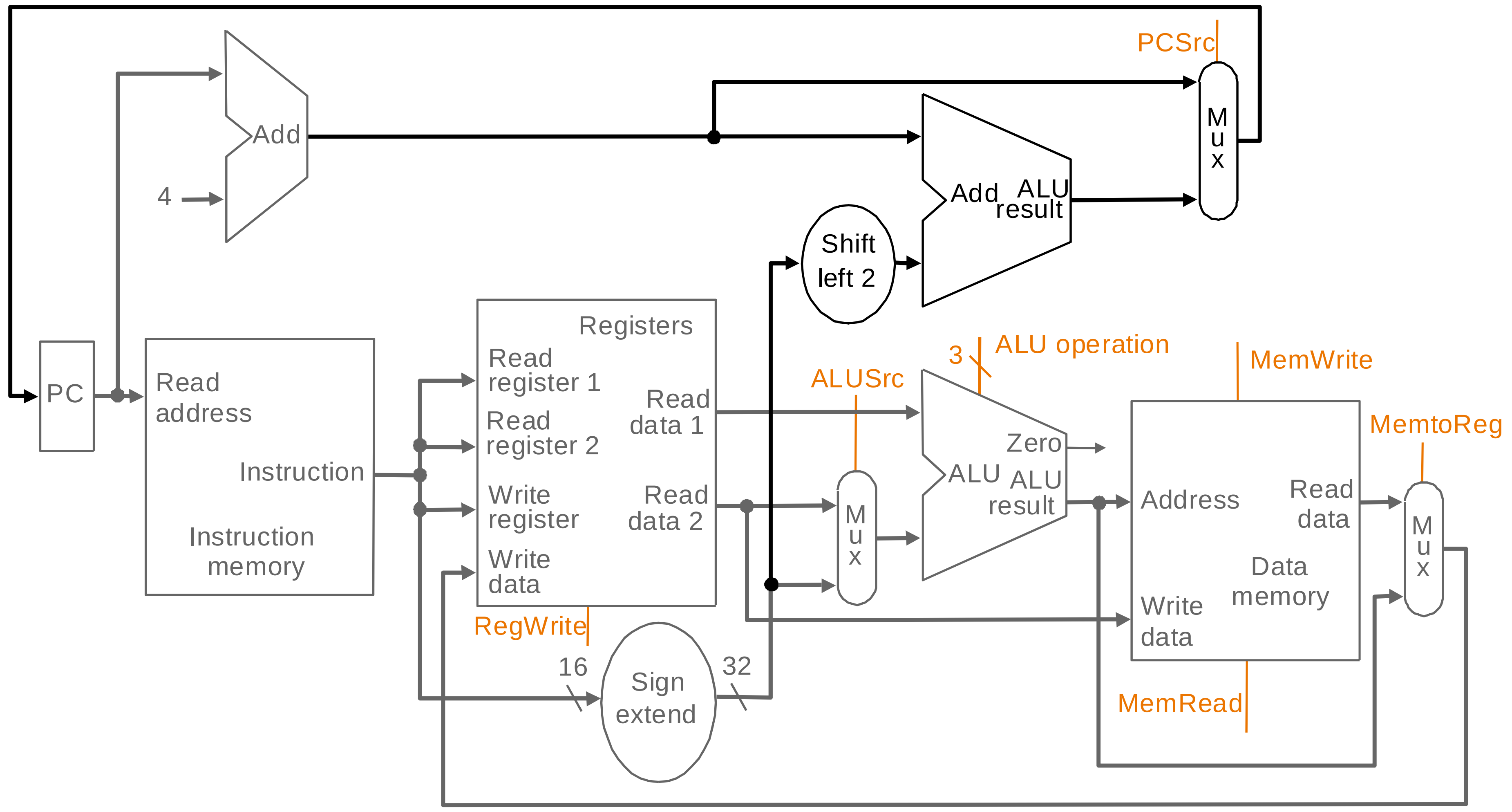
Register File

Clock not shown. Each register is clocked by the same clock, as in lab 3

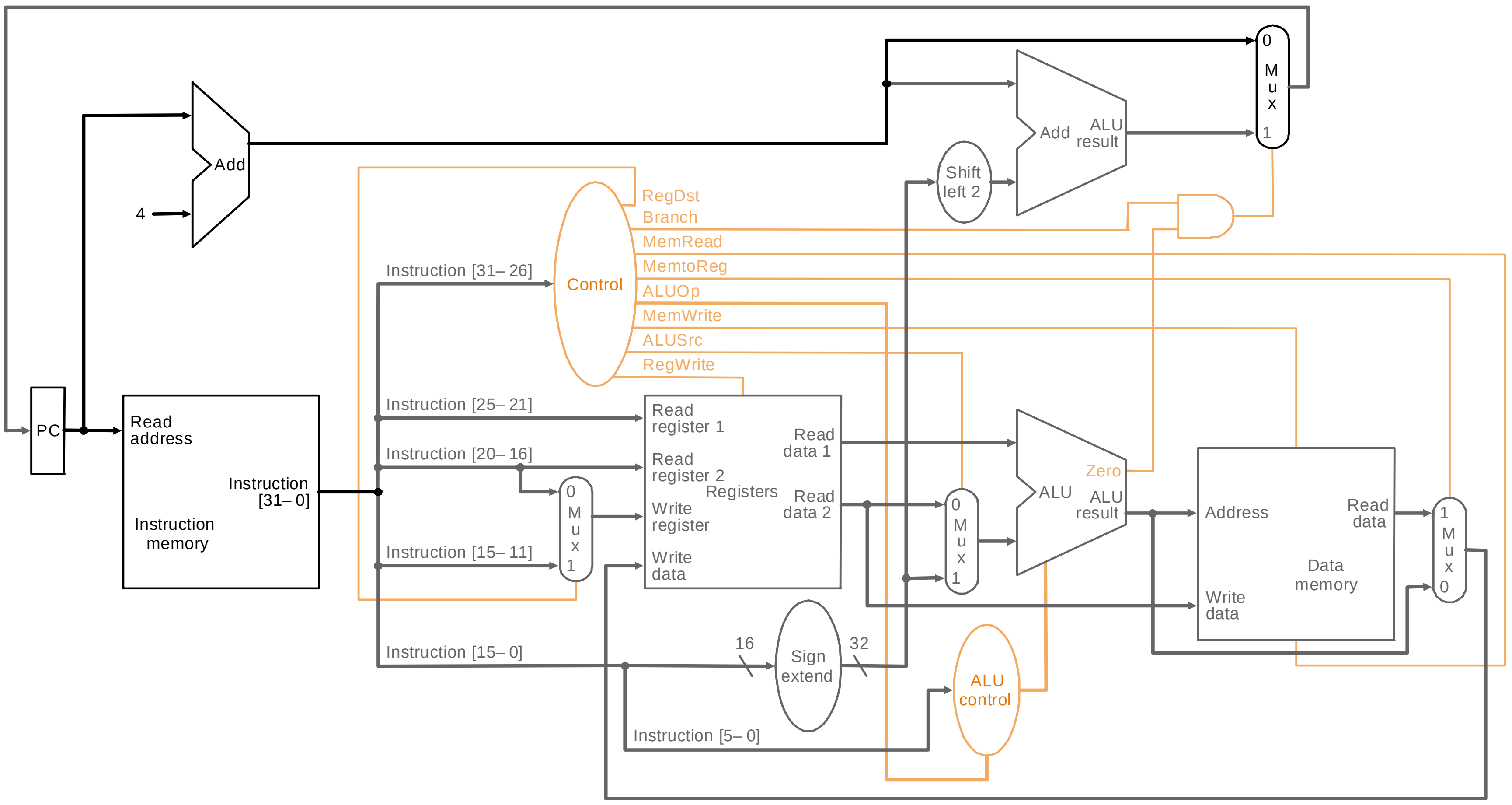


Building the Datapath

Use multiplexers to stitch them together



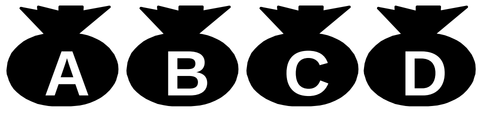
Control



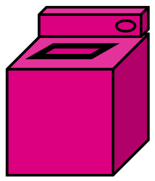
Don’t worry about the details, I just wanted to show you that there is some control logic along with the datapath (as in the datapath lectures examples)

What is Pipelining?

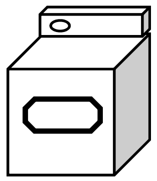
- Ann, Brian, Cathy, Dave
  each have one load of clothes
  to wash, dry, and fold



- Washer takes 30 minutes

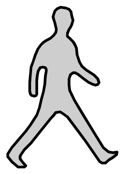


- Dryer takes 30 minutes

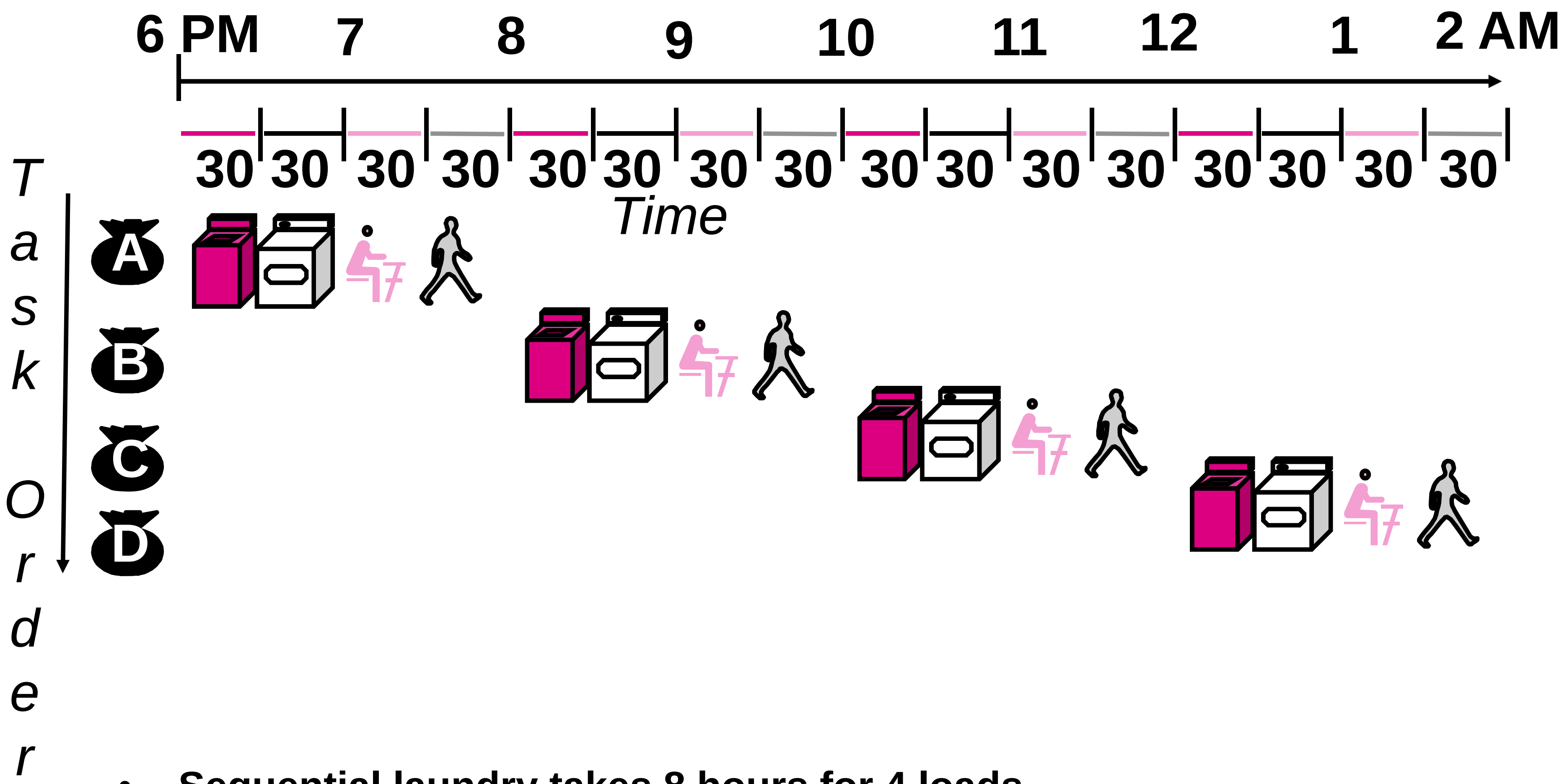


- It takes 30 minutes to fold clothes

- It takes 30 minutes
  to put clothes into drawers

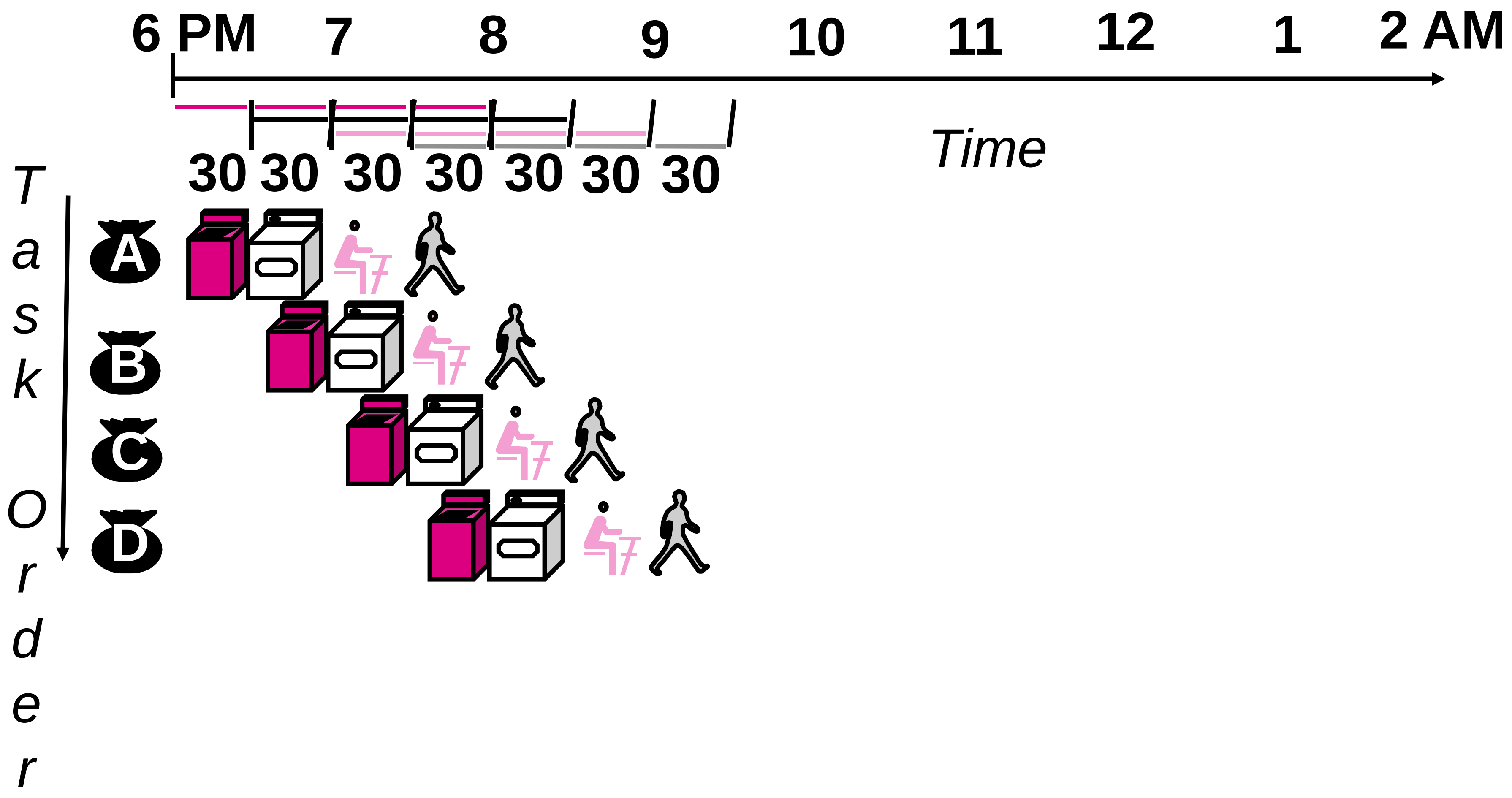


Sequential Laundry



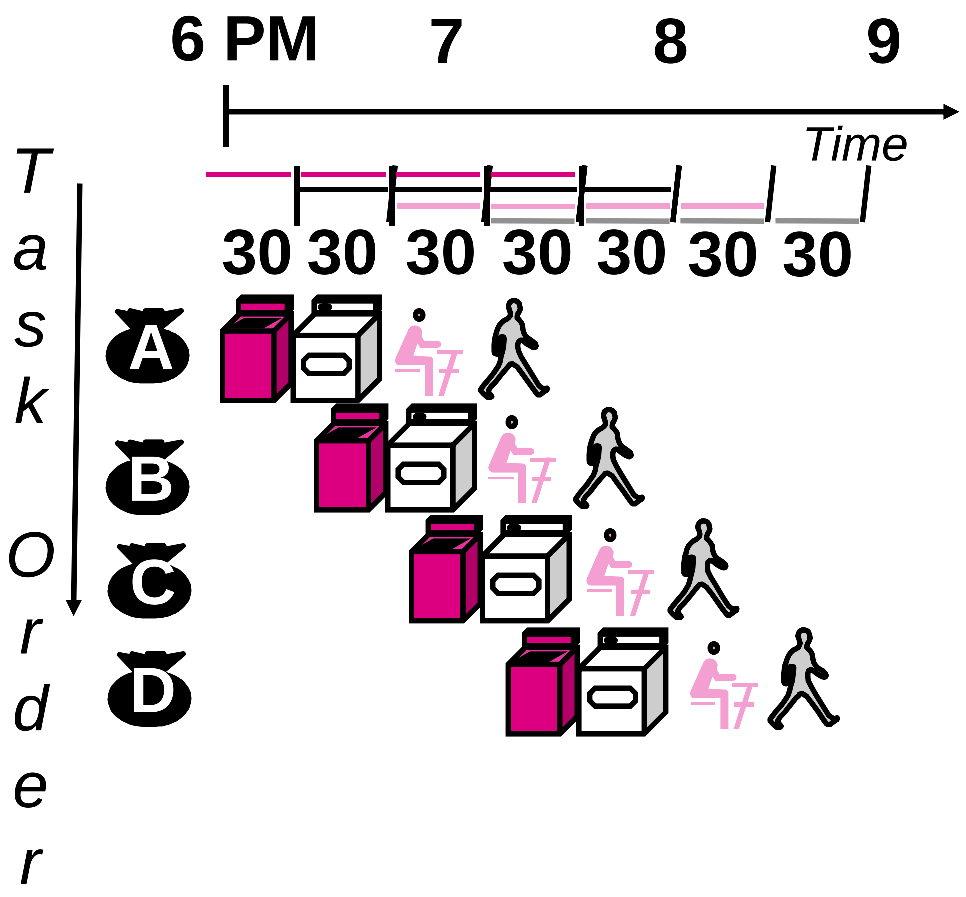
- Sequential laundry takes 8 hours for 4 loads
- If they learned pipelining, how long would laundry take?

Pipelined Laundry: Start work ASAP



- **Pipelined laundry takes 3.5 hours for 4 loads!**

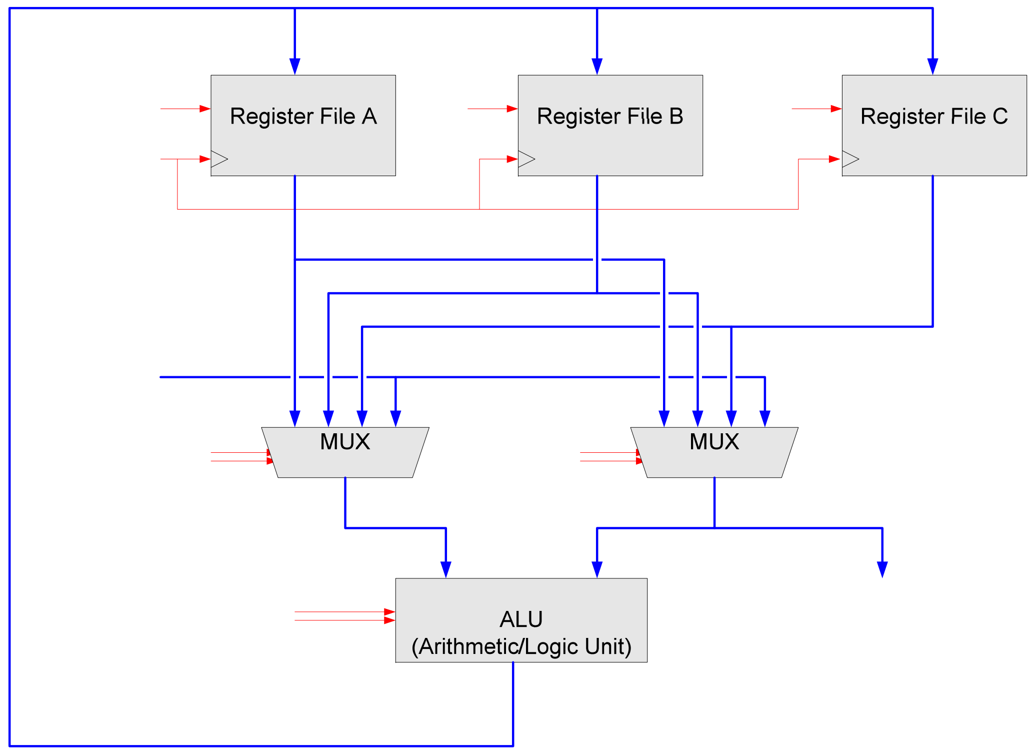
Pipelining Lessons



- Pipelining doesn't help latency of single task, it helps throughput of entire workload
- Multiple tasks operating simultaneously using different resources
- Potential speedup = Number 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

Estimate fastest clock we can use:

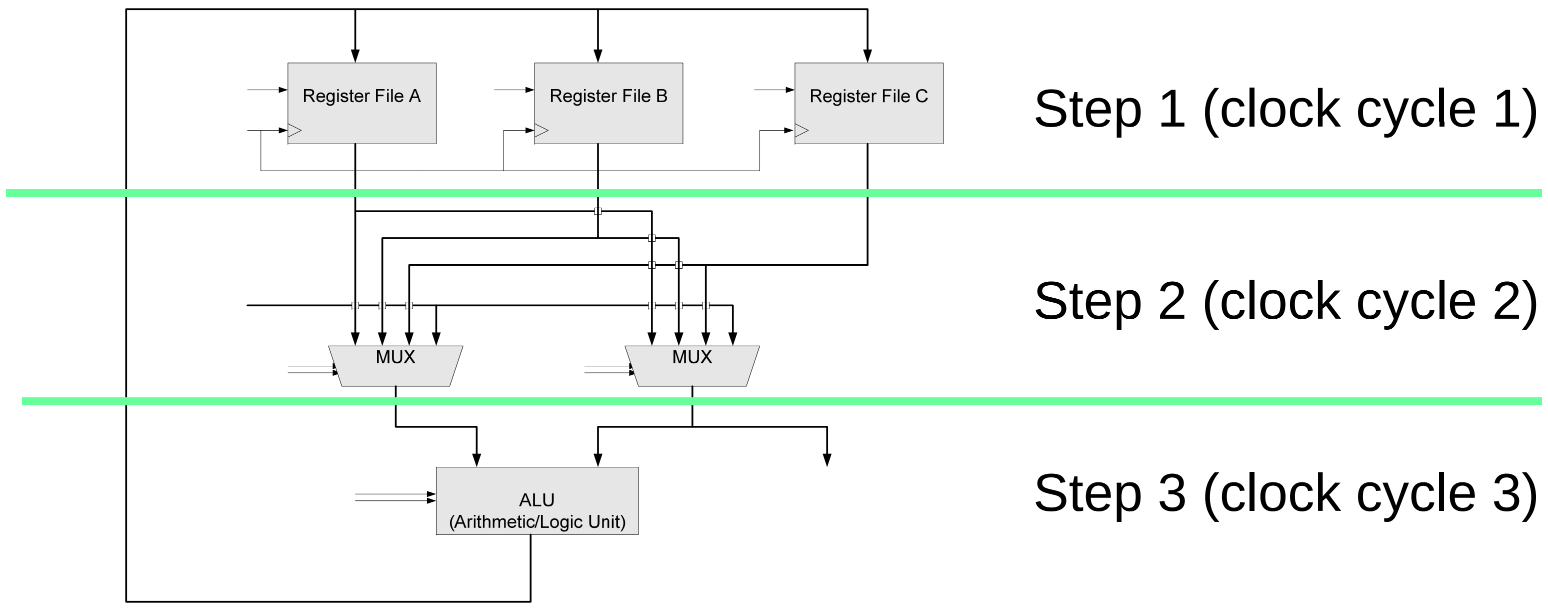
- assume delay of mux = 2ns, delay of ALU=2ns
- assume it takes 2ns after rising clock edge to read a register from a register file (Tclk-Q) and that the setup time is 0.



Shortest clock cycle we can get away with is _______ ns

Multi-Cycle Datapath:

We can use a shorter cycle time by dividing the job into three steps

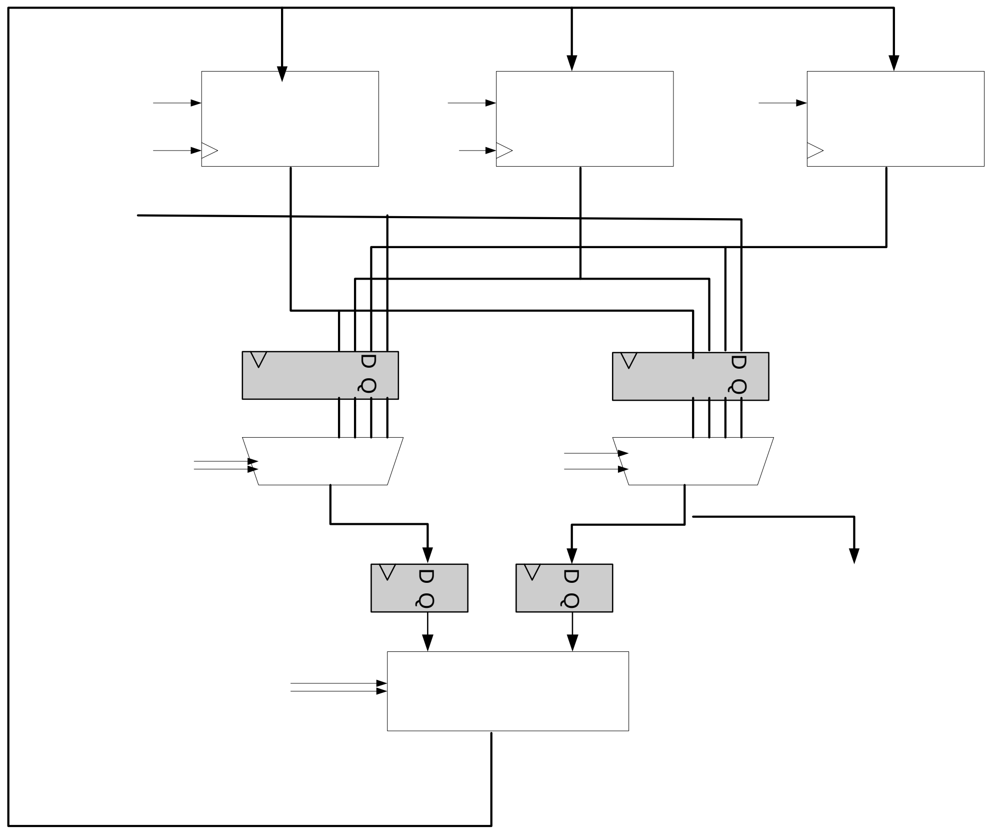


So each cycle can be done in 2 ns ! But we need three of them

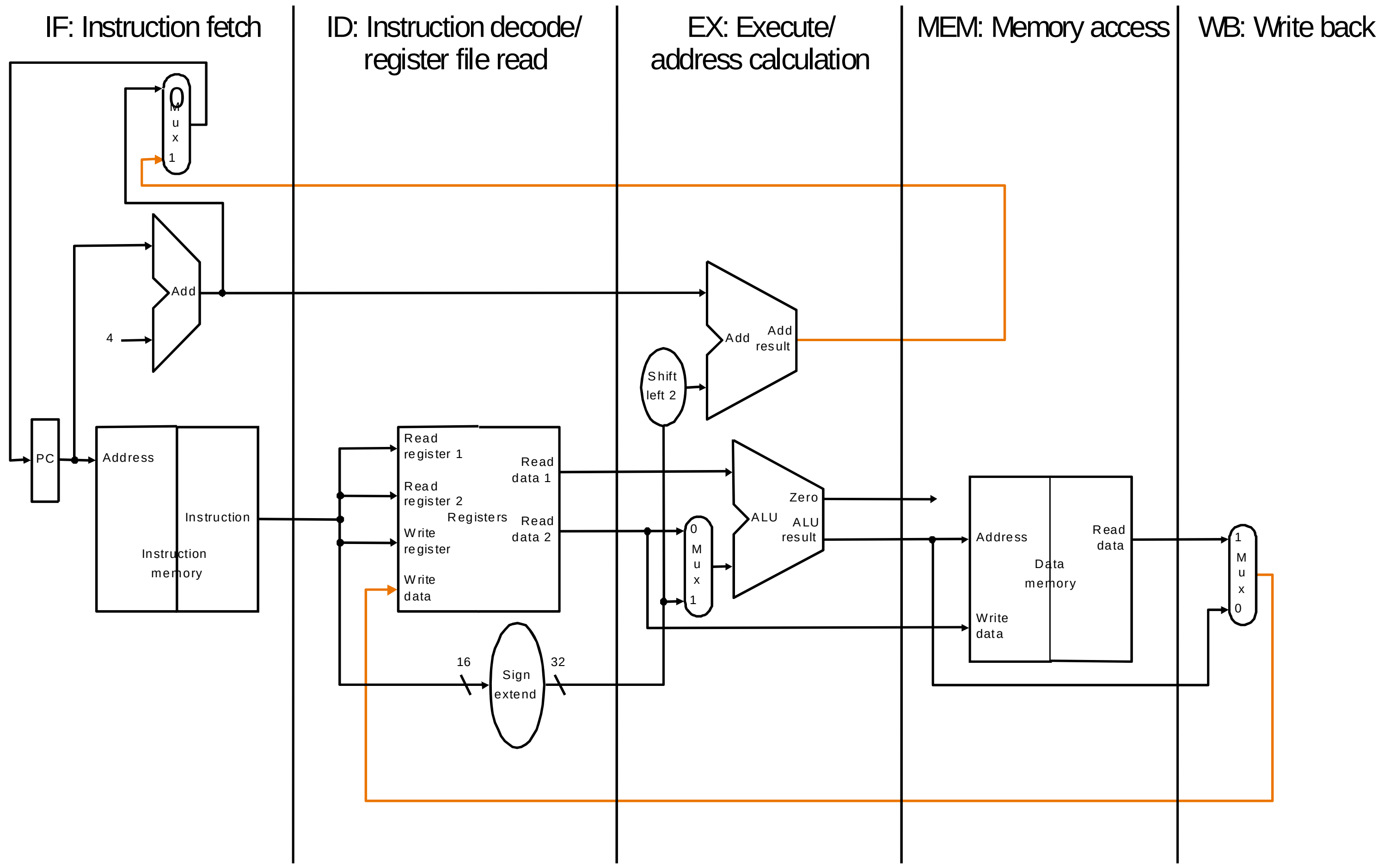
So what’s the advantage??? And what are we assuming???

Multi-Cycle Datapath:

Use Registers to hold values at stage boundaries:



Splitting the MIPS datapath into 5 Stages:



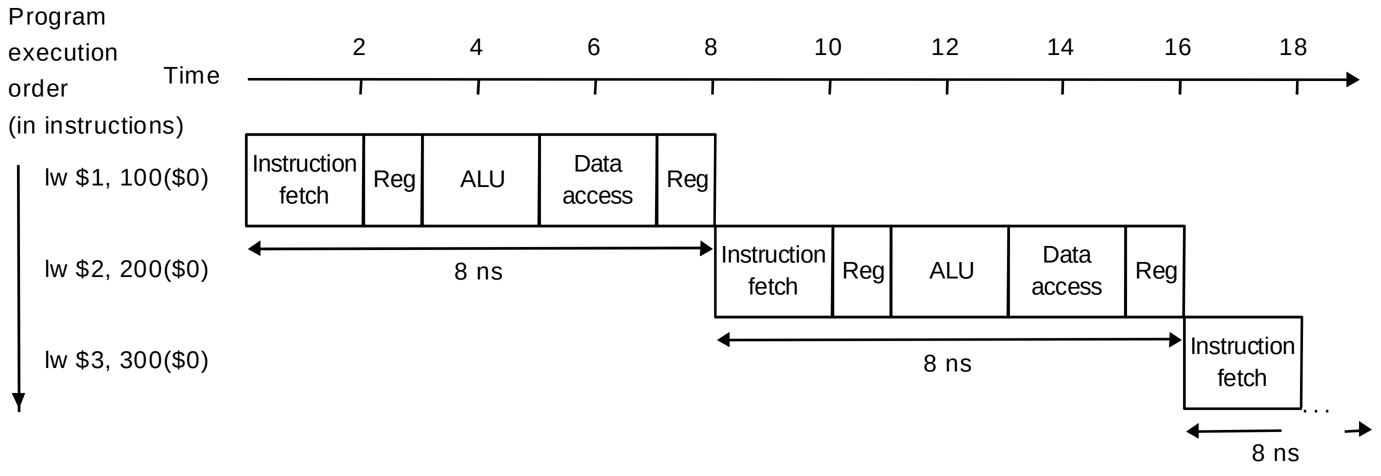
modified from 1998 Morgan Kaufmann Publishers

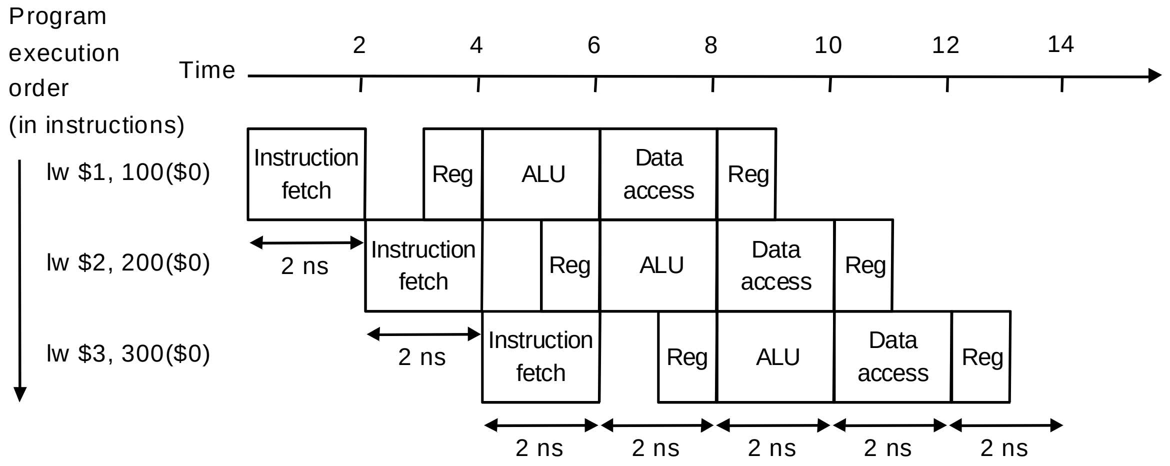
Five Execution Steps

- **Instruction Fetch**
- **Instruction Decode and Register Fetch**
- **Execution, Memory Address Computation, or Branch Completion**
- **Memory Access or R-type instruction completion**
- **Write-back step**

INSTRUCTIONS TAKE FROM 3 - 5 CYCLES!

Pipelining





Ideal speedup is number of stages in the pipeline. Do we achieve this?

Can pipelining get us into trouble?

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

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: try to make 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

Data Hazards:

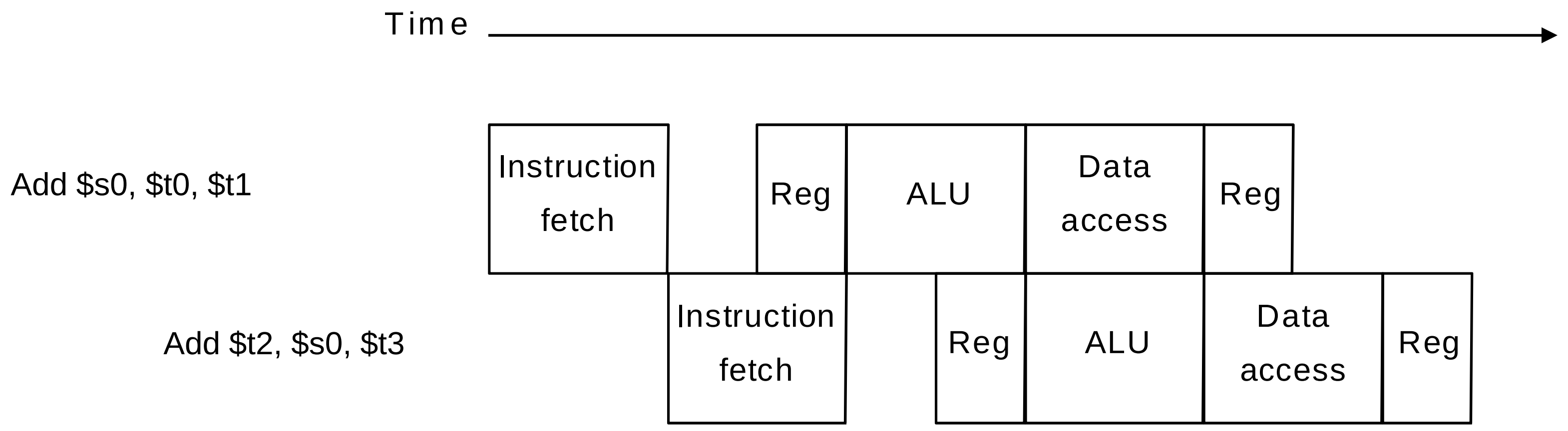
Consider the following instructions:

add $s0, $t0, $t1 -- This means add registers t0 and t1 and store the result in s0

add $t2, $s0, $t3 -- This means add registers s0 and t3 and store the result in t2

...

Do you see a problem?



Data Hazards:

One Solution: Compiler could add no-op instructions

```
add $s0, $t0, $t1
nop
nop
nop                                                    ...
add $t2, $s0, $t3
```

But this really slows us down.

Can re-order instructions (perhaps we can get some useful work done instead of executing no-ops).

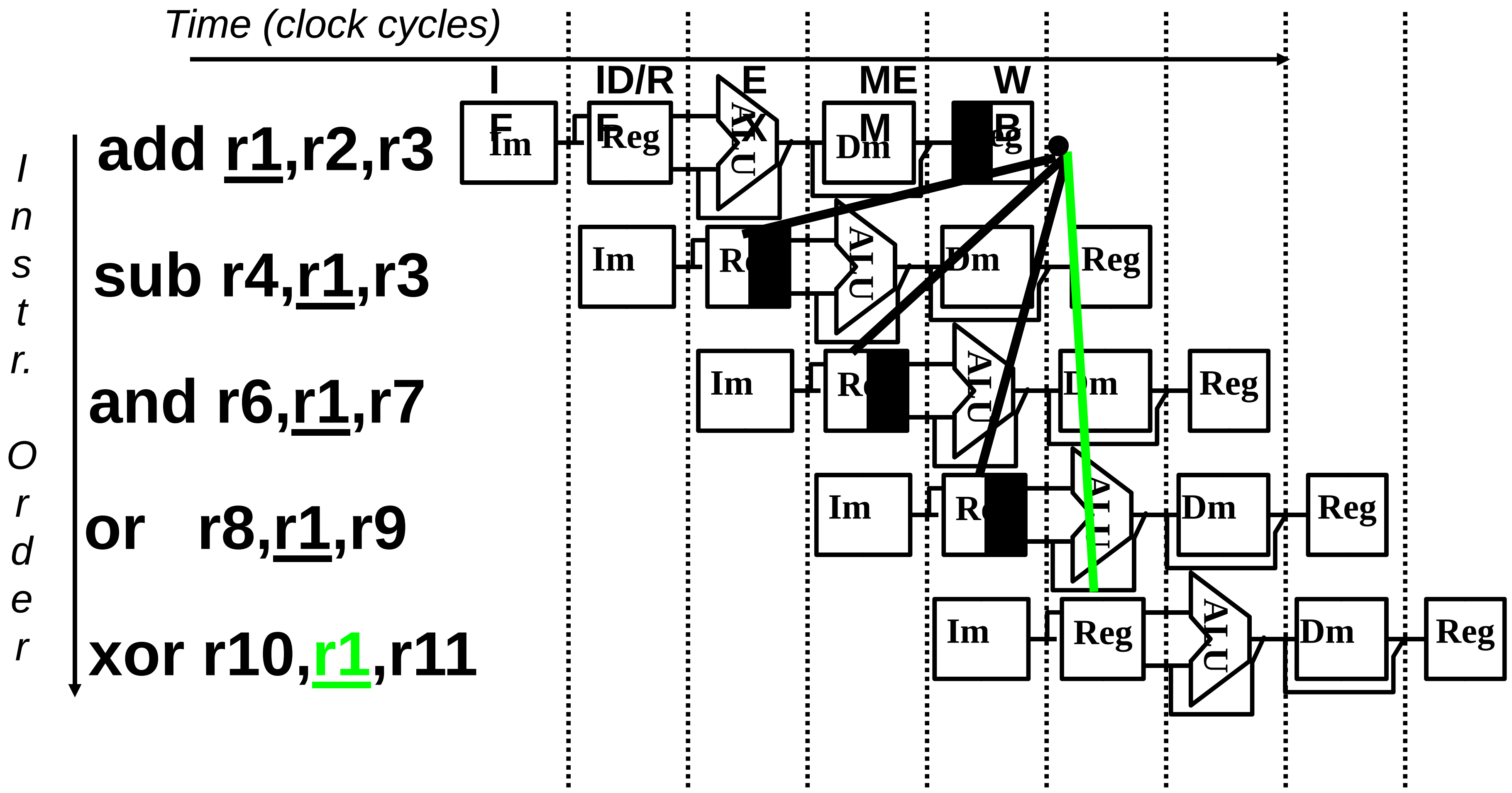
- but this is tricky to deal with in a compiler

Data Hazard on r1

```
add r1 ,r2,r3
sub r4, r1 ,r3
and r6, r1 ,r7
or   r8, r1 ,r9
xor r10, r1 ,r11
```

Data Hazard on r1:

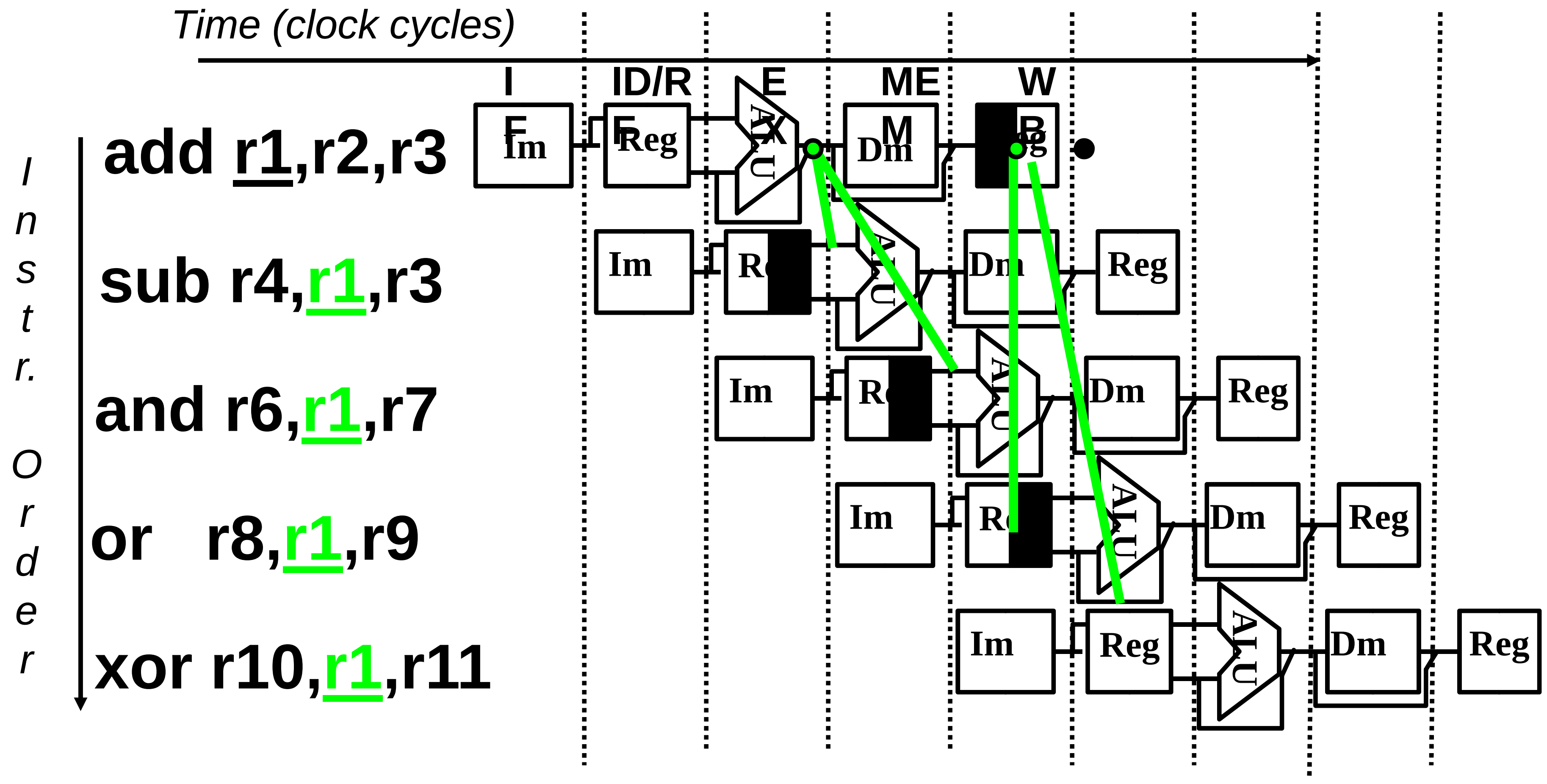
- **Dependencies backwards in time are hazards**



Modified From: © D. Patterson, UCB, 1997

Data Hazard Solution:

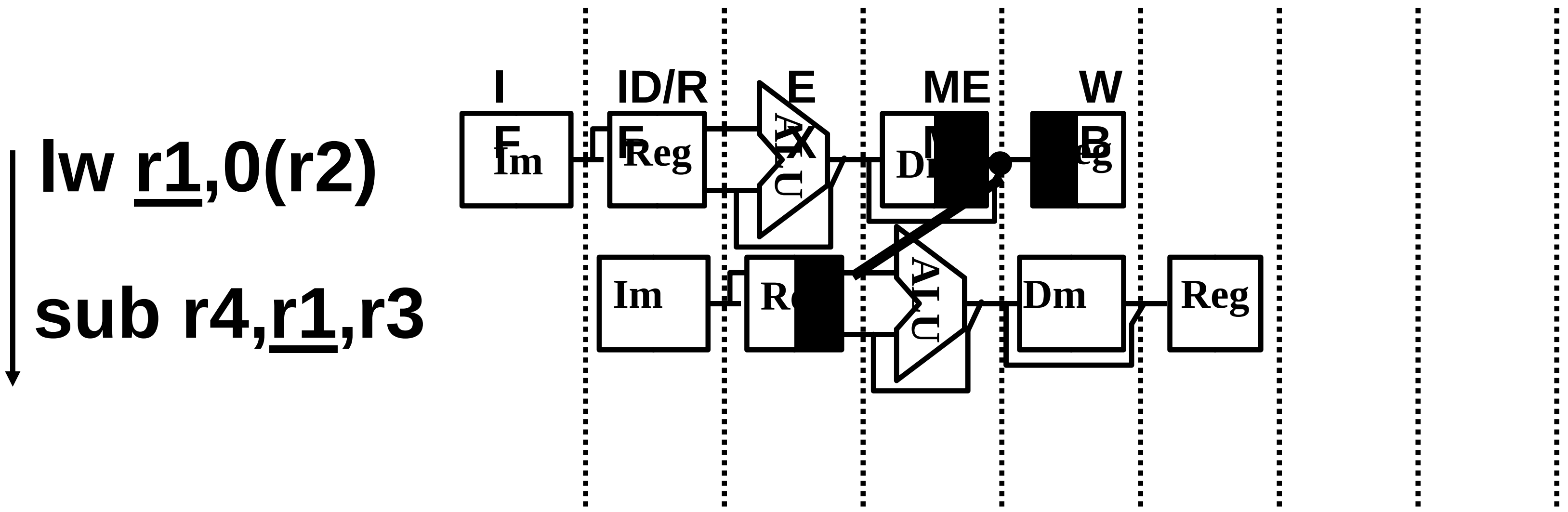
- "Forward" result from one stage to another



Modified From: © D. Patterson, UCB, 1997

Forwarding (or Bypassing): What about Loads

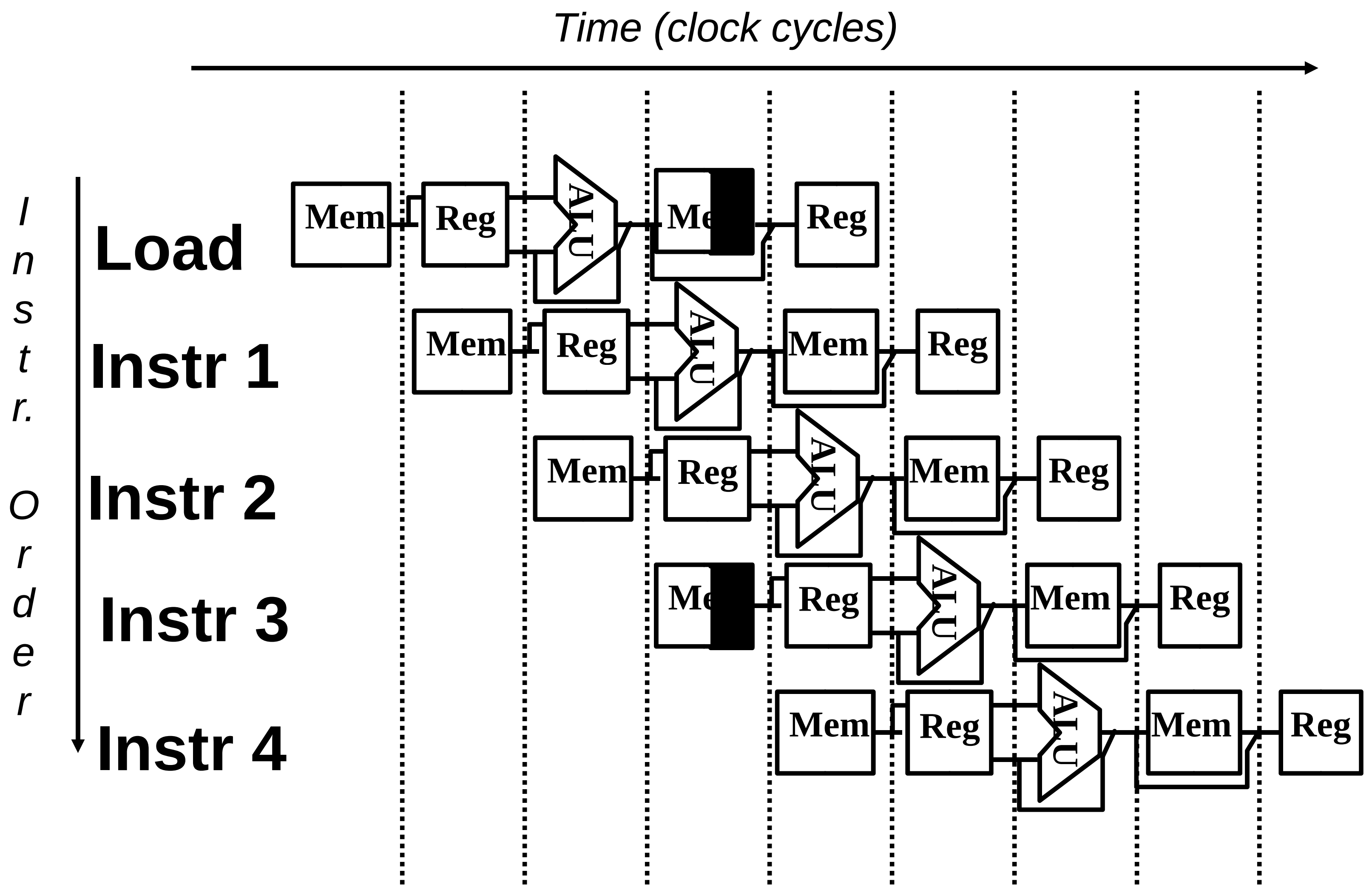
- Dependencies backwards in time are hazards



- Can’t solve with forwarding:
- Must delay/stall instruction dependent on loads

Modified From: © D. Patterson, UCB, 1997

Single Memory is a Structural Hazard



Modified From: © D. Patterson, UCB, 1997

Control Hazards:

Another problem: What happens when we execute a branch?

- we don’t know if we will be taking the branch until the last step

- but by then, other instructions are in the pipeline!

...

- need to flush pipeline whenever we take a branch

Some processors have “delay slots”

- the next instruction after a branch is always execute

- rely on compiler to “fill” the slot with something useful

Most processors have h/w to predict if a branch will be taken

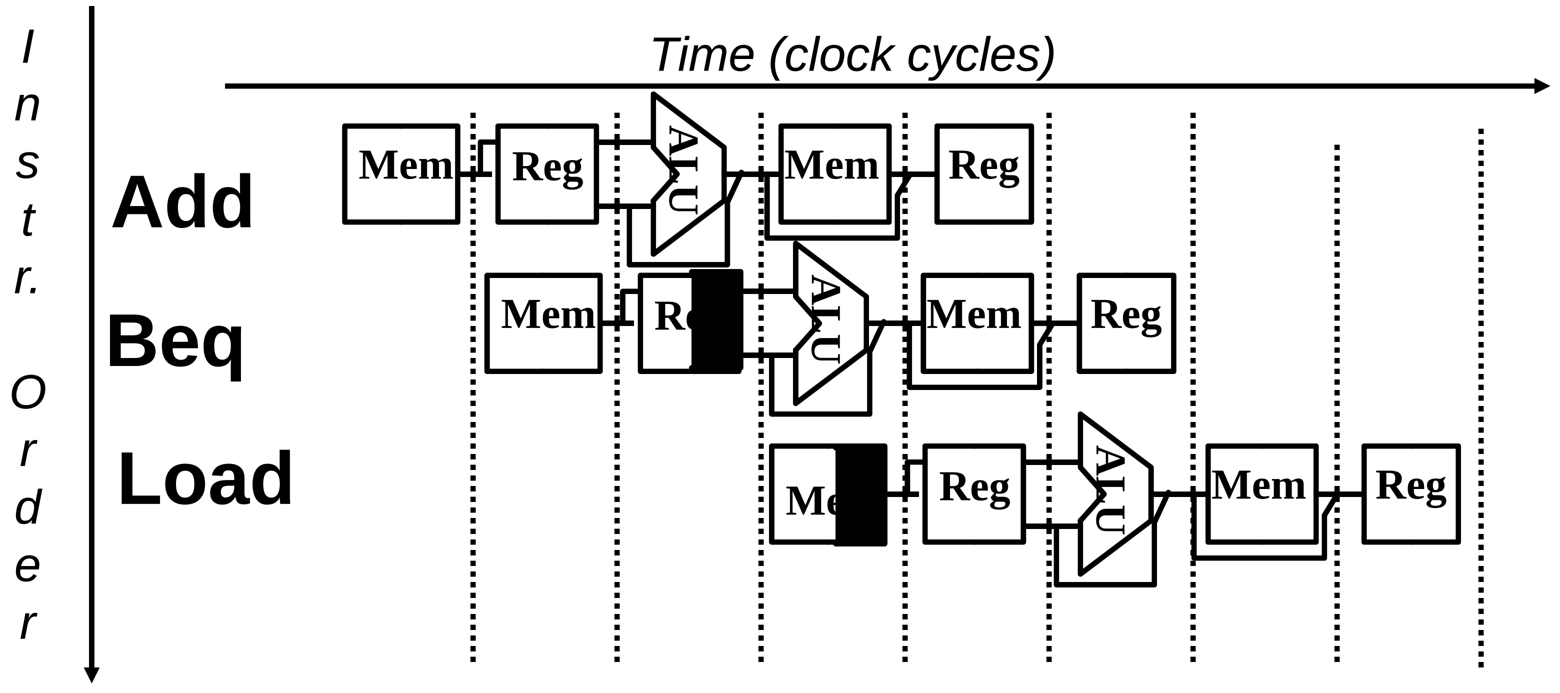
- after a branch, load pipeline with most-likely next instruction

- if we are wrong, we still have to flush

Control Hazard Solutions

Stall: wait until decision is clear

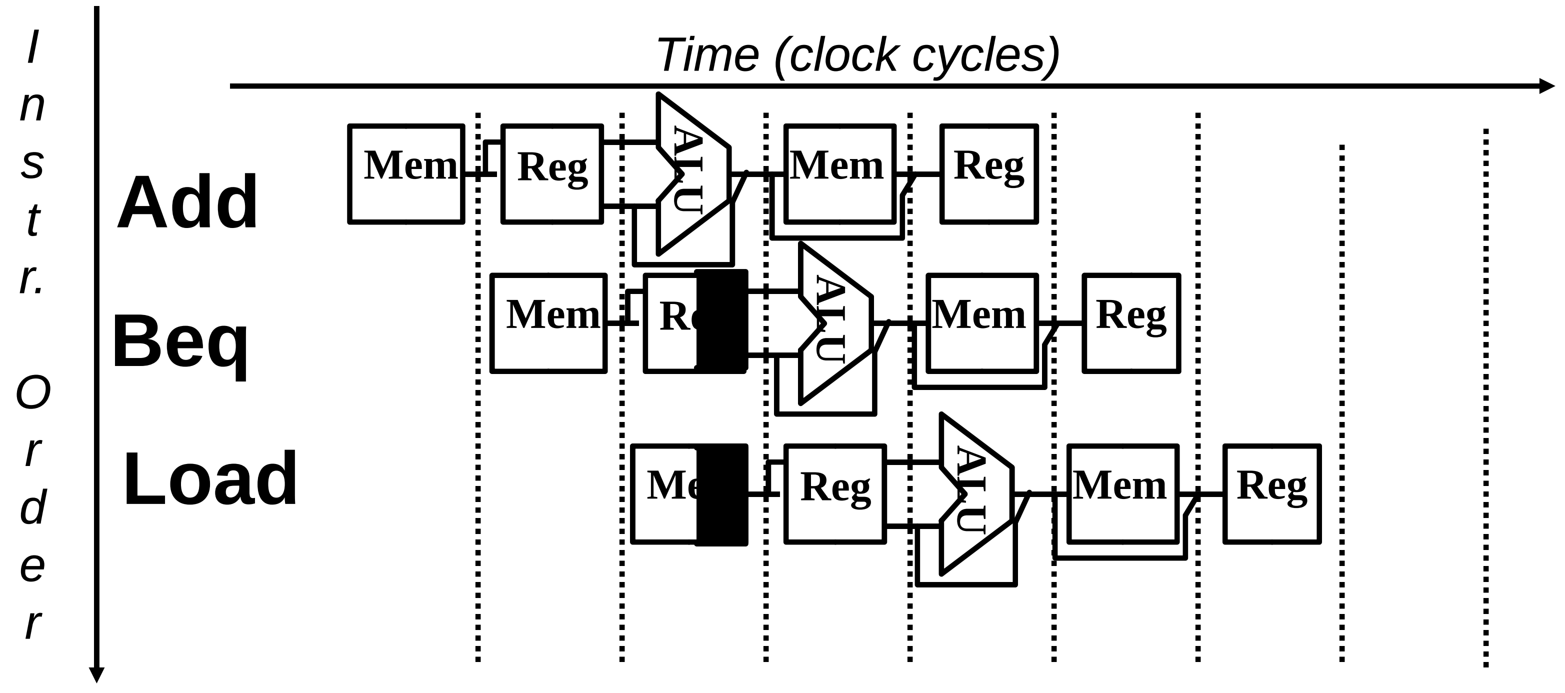
- **Its possible to move up decision to 2nd stage by adding hardware to check registers as being read**



Impact: 1 or 2 clock cycles per branch instruction
=> slow

Control Hazard Solutions

- **Predict: guess one direction then back up if wrong**
  - **Predict not taken**

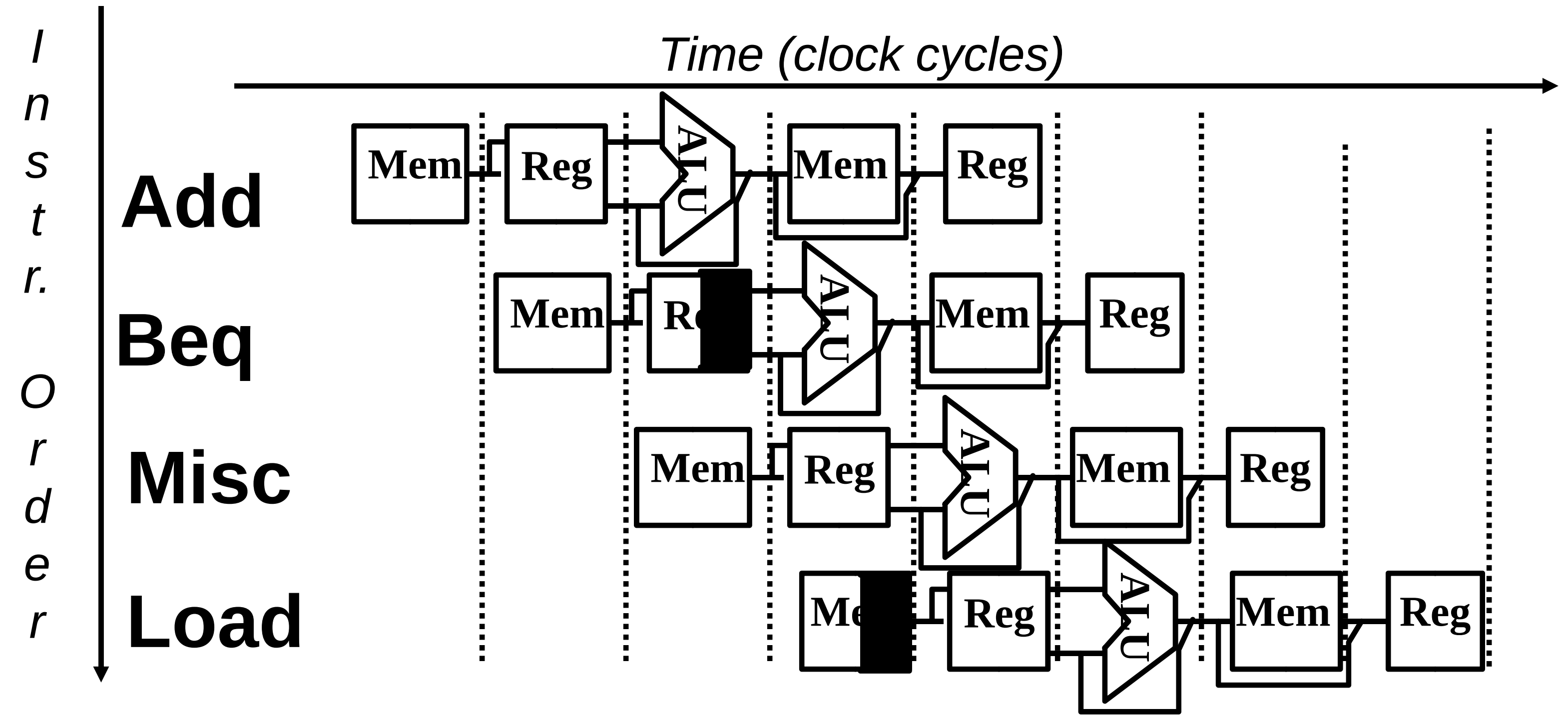


- **Impact: 1 clock cycles per branch instruction if right, 2 if wrong (right - 50% of time)**
- **More dynamic scheme: history of 1 branch (- 90%)**

Modified From: © D. Patterson, UCB, 1997

Control Hazard Solutions

- Redefine branch behavior (takes place after next instruction) "delayed branch"



Summary:

Datapaths can be pipelined, executing several instructions at once

Need to worry about control and data hazards:
- Can be taken care of by hardware or compiler or both!

Superscalar machines have an additional twist: there are several functional units (eg. ALUs)
- can have several instructions in each stage at once
- hardware or compiler decides which instruction will be assigned to which functional unit
- in this case, we have to worry about “structural hazards”
    - don’t assign two instructions to the same ALU at a time