

Introduction to Pipelining

Lecturer: Dr. Hui Annie Guo
huig@cse.unsw.edu.au
K17-501F (ext. 57136)

Overview

- What is pipelining?
- Why to use pipelining?
- Issues with pipelining design

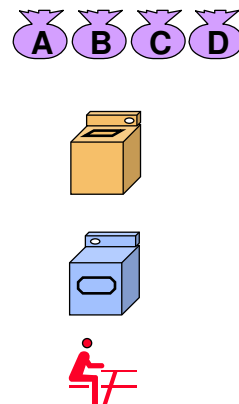
COMP3211/9211

2011S1 wk4_2 P2

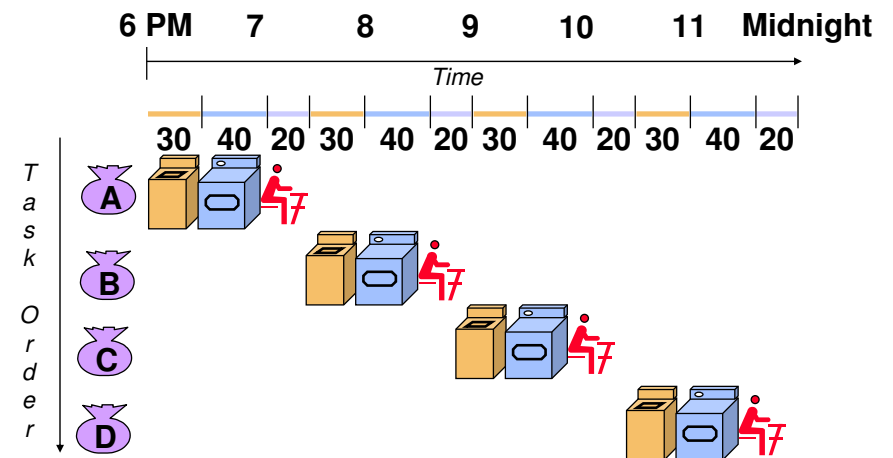
Pipelining is Natural!

Laundry Example

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



Sequential Laundry



- Sequential laundry takes 6 hours for 4 loads
- How long would laundry take if they learnt pipelining?

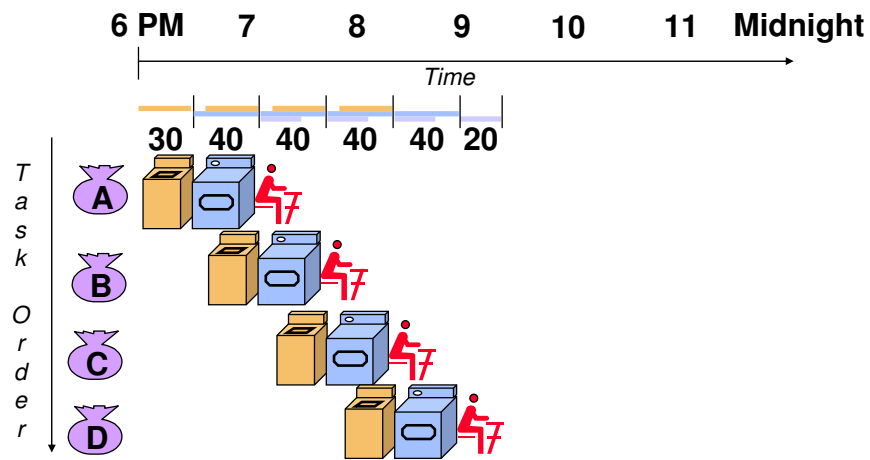
COMP3211/9211

2011S1 wk4_2 P3

COMP3211/9211

2011S1 wk4_2 P4

Pipelined Laundry: Start work ASAP

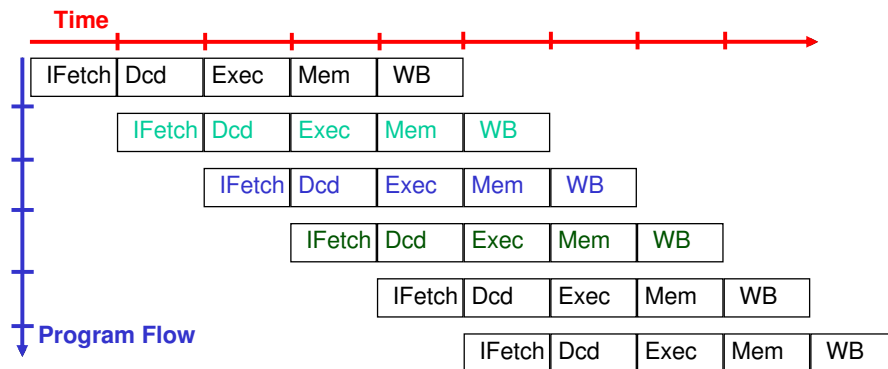


- Pipelined laundry takes 3.5 hours for 4 loads

Remarks

- 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

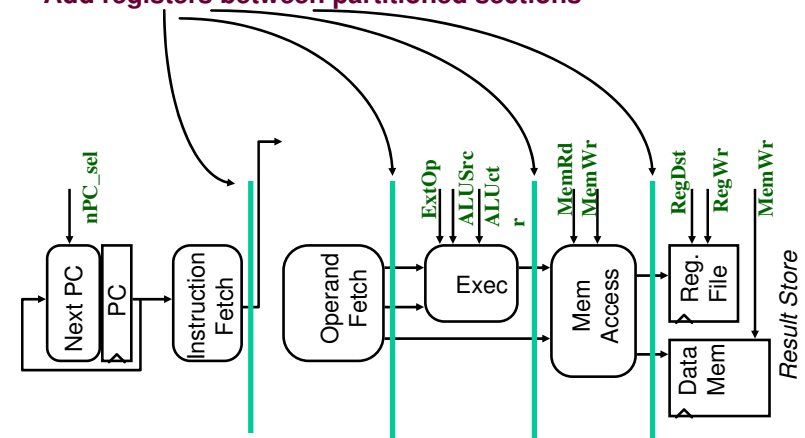
Pipelined Processor Execution



- Utilization?
- Now we just have to make it work

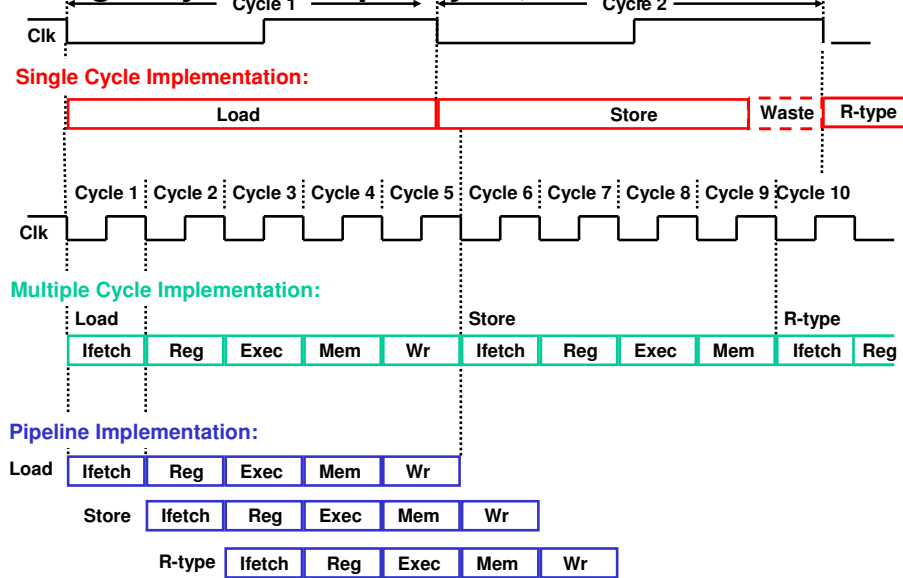
Recall: Partitioning the CPI=1 Datapath

- Add registers between partitioned sections



- Want to balance time spent in each section
 - Remember each cycle takes the same amount of time
 - The longest stage delay defines the cycle time

Single Cycle, Multiple Cycle, vs. Pipeline



COMP3211/9211

2011S1 wk4_2 P9

Why Pipeline?

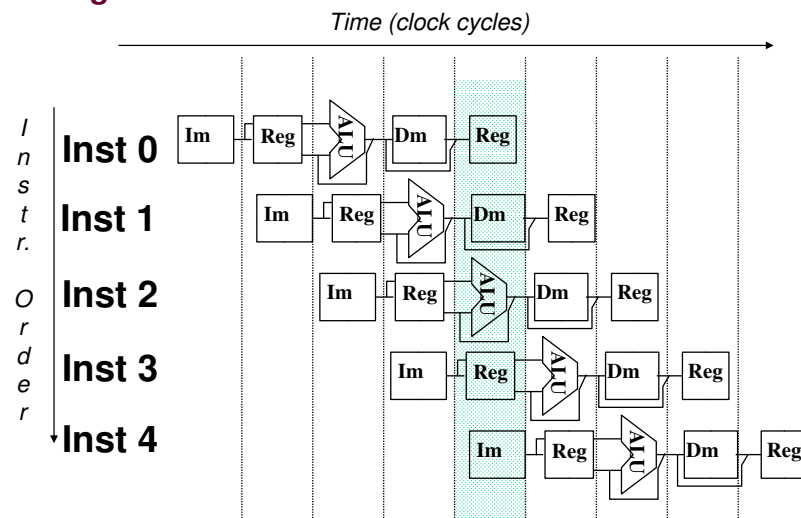
- **High performance**
- **Example**
 - Suppose we execute 100 instructions
 - **Single Cycle Machine**
 - $45 \text{ ns/cycle} \times 1 \text{ CPI} \times 100 \text{ inst} = 4500 \text{ ns}$
 - **Multicycle Machine**
 - $10 \text{ ns/cycle} \times 4.1 \text{ CPI (due to inst mix)} \times 100 \text{ inst} = 4100 \text{ ns}$
 - **Ideal pipelined machine**
 - $10 \text{ ns/cycle} \times (1 \text{ CPI} \times 100 \text{ inst} + 4 \text{ cycle drain}) = 1040 \text{ ns}$

COMP3211/9211

2011S1 wk4_2 P10

Why Pipeline?

- **High hardware utilization**



COMP3211/9211

2011S1 wk4_2 P11

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
 - **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

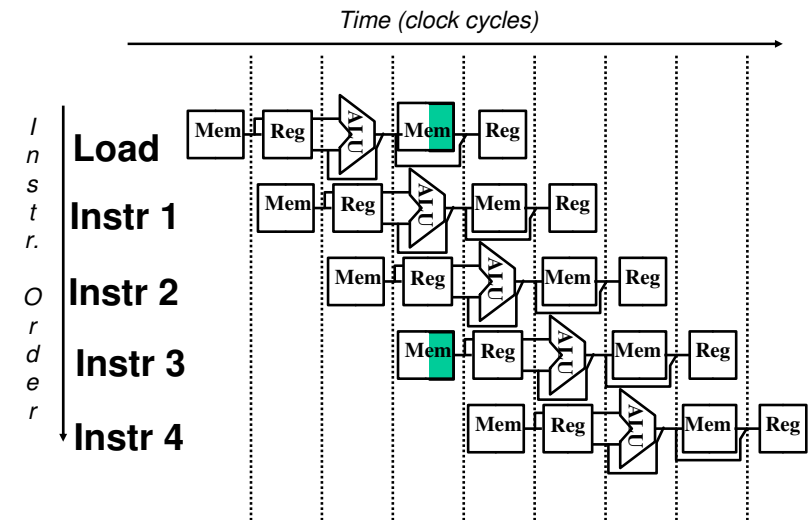
2011S1 wk4_2 P12

Solutions?

- **Variety of solutions**
 - Some will be discussed in detail later
- **A straightforward solution:**
 - Resolving hazards by waiting
 - Pipeline control must detect the hazard

2011S1 wk4_2 P13

Single Memory is a Structural Hazard



Detection is easy in this case! (right half highlight means read, left half write)

COMP3211/9211

2011S1 wk4_2 P14

Structural Hazards Limit Performance

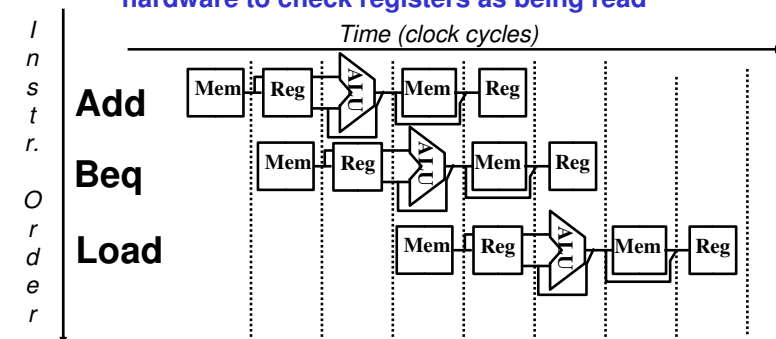
- **Example: if 1.3 memory accesses per instruction and only one memory access per cycle is allowed then**
 - average CPI ≥ 1.3
 - otherwise resource is more than 100% utilized
- **The solution is to provide two memories**

COMP3211/9211

2011S1 wk4_2 P15

Conditional Branches Cause Control Hazard

- **Should wait until decision is clear**
 - It's possible to move up decision to 2nd stage by adding hardware to check registers as being read



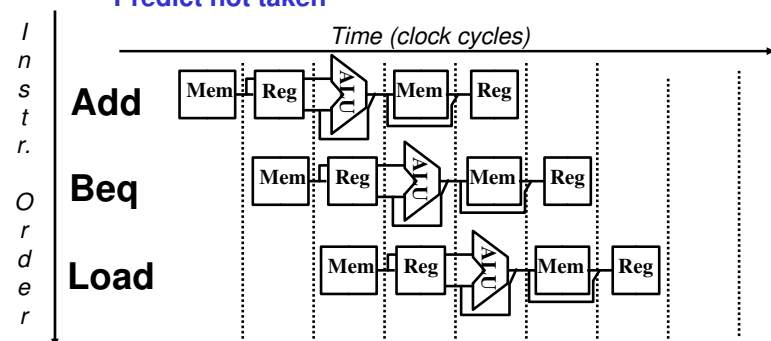
- **Impact: 2 clock cycles per branch instruction**
=> increases CPI & slows down overall performance

COMP3211/9211

2011S1 wk4_2 P16

Control Hazard Solutions

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



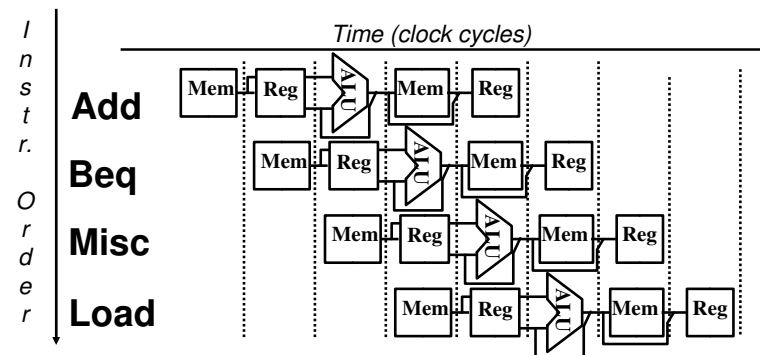
- Impact: 1 clock cycle per branch instruction if right, 2 if wrong (right - 50% of time)
- More dynamic scheme: history of 1 branch (~90%✓)

COMP3211/9211

2011S1 wk4_2_P17

Control Hazard Solutions

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



- Impact: 1 clock cycles per branch instruction if can find instruction to put in “slot” (- 50% of time)
- As launch more instruction per clock cycle, less useful

COMP3211/9211

2011S1 wk4_2_P18

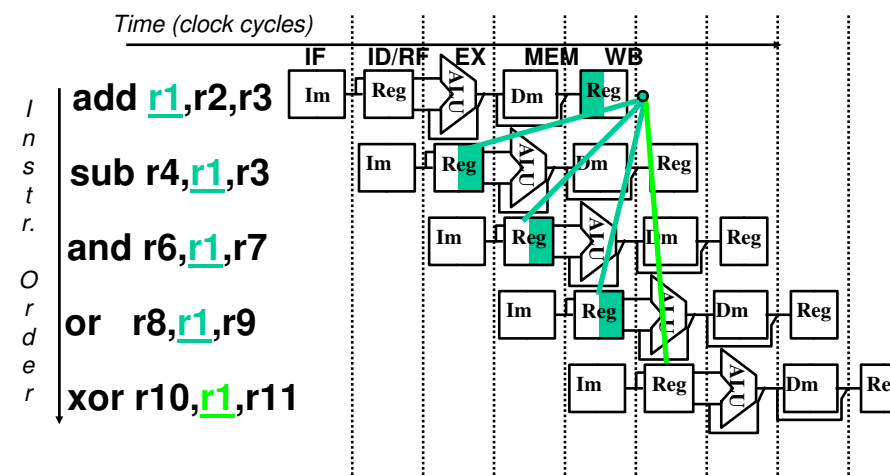
Data Dependencies Cause Data Hazard

add r1, r2, r3 ; store result in reg r1
 sub r4, r1, r3 ; read operand in reg r1
 and r6, r1, r7
 or r8, r1, r9
 xor r10, r1, r11

COMP3211/9211

2011S1 wk4_2_P19

Dependencies Backwards In Time are Hazards

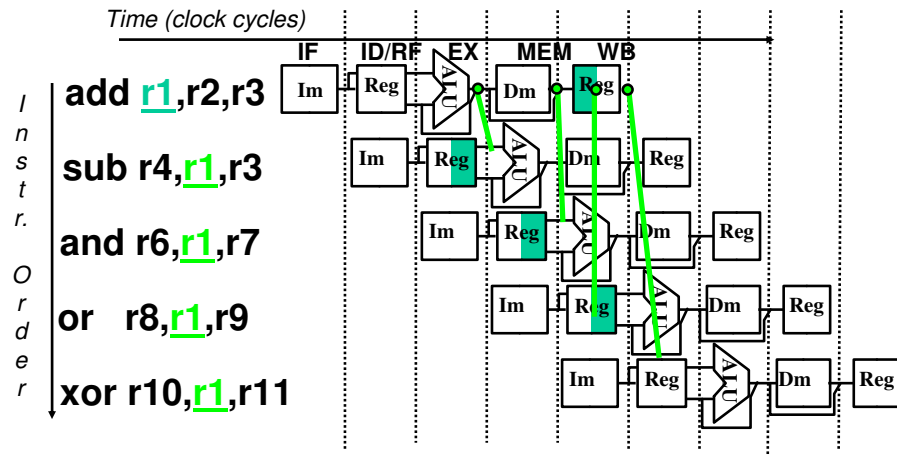


COMP3211/9211

2011S1 wk4_2_P20

Data Hazard Solution:

- “Forward” result from one stage to another



- “OR” OK if define read/write properly