

28 October

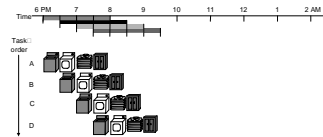
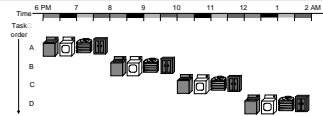
- 10 Classes to go!
- Read Sections 7.1 and 7.2
- You read 6.1 for this time. Right?
- Pipelining then on to Memory hierarchy

10/28/2004

Comp 120 Fall 2004

1

Doing Laundry



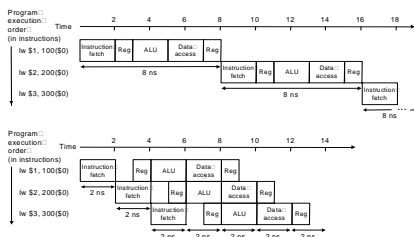
10/28/2004

Comp 120 Fall 2004

2

Pipelining

- Improve performance by increasing instruction throughput



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

10/28/2004

Comp 120 Fall 2004

3

Pipeline control

- We have 5 stages. What needs to be controlled in each stage?
 - Instruction Fetch and PC Increment
 - Instruction Decode / Register Fetch
 - Execution
 - Memory Stage
 - Register Write Back
- How would control be handled in an automobile plant?
 - a fancy control center telling everyone what to do?
 - should we use a finite state machine?

10/28/2004

Comp 120 Fall 2004

4

Pipelining

- What makes it easy
 - all instructions are the same length
 - just a few instruction formats
 - memory operands appear only in loads and stores
- What makes it hard?
 - structural hazards: suppose we had only one memory
 - control hazards: need to worry about branch instructions
 - data hazards: an instruction depends on a previous instruction
- Individual Instructions still take the same number of cycles
- But we've improved the through-put by increasing the number of simultaneously executing instructions

10/28/2004

Comp 120 Fall 2004

5

Structural Hazards

Inst Fetch	Reg Read	ALU	Data Access	Reg Write			
	Inst Fetch	Reg Read	ALU	Data Access	Reg Write		
		Inst Fetch	Reg Read	ALU	Data Access	Reg Write	
			Inst Fetch	Reg Read	ALU	Data Access	Reg Write

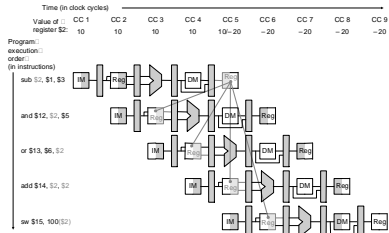
10/28/2004

Comp 120 Fall 2004

6

Data Hazards

- Problem with starting next instruction before first is finished
 - dependencies that "go backward in time" are data hazards



10/28/2004

Comp 120 Fall 2004

7

Software Solution

- Have compiler guarantee no hazards
- Where do we insert the "nops" ?

```
sub    $2, $1, $3
and    $12, $2, $5
or     $13, $6, $2
add    $14, $2, $2
sw     $15, 100($2)
```

- Problem: this really slows us down!

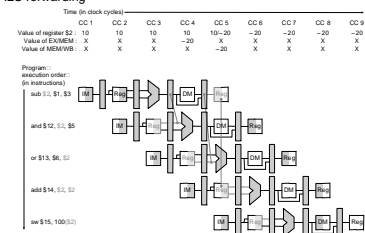
10/28/2004

Comp 120 Fall 2004

8

Forwarding

- Use temporary results, don't wait for them to be written
 - register file forwarding to handle read/write to same register
 - ALU forwarding



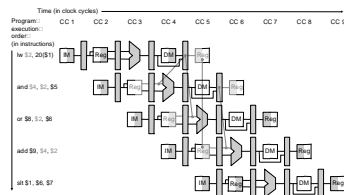
10/28/2004

Comp 120 Fall 2004

9

Can't always forward

- Load word can still cause a hazard:
 - an instruction tries to read a register following a load instruction that writes to the same register.



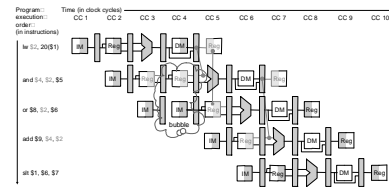
10/28/2004

Comp 120 Fall 2004

10

Stalling

- We can stall the pipeline by keeping an instruction in the same stage



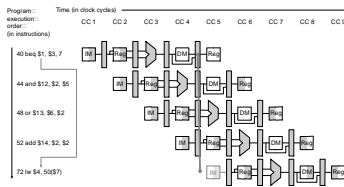
10/28/2004

Comp 120 Fall 2004

11

Branch Hazards

- When we decide to branch, other instructions are in the pipeline!



10/28/2004

Comp 120 Fall 2004

12

Improving Performance

- Try to avoid stalls! E.g., reorder these instructions:

```
lw $t0, 0($t1)
lw $t2, 4($t1)
sw $t2, 0($t1)
sw $t0, 4($t1)
```

- Add a "branch delay slot"
 - the next instruction after a branch is always executed
 - rely on compiler to "fill" the slot with something useful
- Superscalar: start more than one instruction in the same cycle

10/28/2004

Comp 120 Fall 2004

13

Dynamic Scheduling

- The hardware performs the "scheduling"
 - hardware tries to find instructions to execute
 - out of order execution is possible
 - speculative execution and dynamic branch prediction
- All modern processors are very complicated
 - Pentium 4: 20 stage pipeline, 6 simultaneous instructions
 - PowerPC and Pentium: branch history table
 - Compiler technology important

10/28/2004

Comp 120 Fall 2004

14

Chapter 7 Preview

Memory Hierarchy

10/28/2004

Comp 120 Fall 2004

15

Memory Hierarchy

- Memory devices come in several different flavors
 - SRAM – Static Ram
 - fast (1 to 10ns)
 - expensive (>10 times DRAM)
 - small capacity (< ¼ DRAM)
 - DRAM – Dynamic RAM
 - 16 times slower than SRAM (50ns – 100ns)
 - Access time varies with address
 - Affordable (\$160 / gigabyte)
 - 1 Gig considered big
 - DISK
 - Slow! (10ms access time)
 - Cheap! (< \$1 / gigabyte)
 - Big! (1Tbyte is no problem)

10/28/2004

Comp 120 Fall 2004

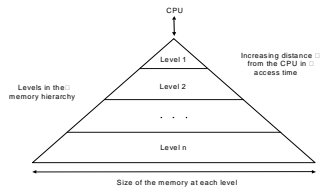
16

Memory Hierarchy

- Users want large and fast memories!

Try to give it to them

- build a memory hierarchy



10/28/2004

Comp 120 Fall 2004

17

Locality

- A principle that makes having a memory hierarchy a good idea
- If an item is referenced,

temporal locality: it will tend to be referenced again soon

spatial locality: nearby items will tend to be referenced soon.

Why does code have locality?

10/28/2004

Comp 120 Fall 2004

18

9

10/28/2004

Comp 120 Fall 2004

19