

**Slide Set #21:
Exploiting ILP**

Chapter 6 and beyond

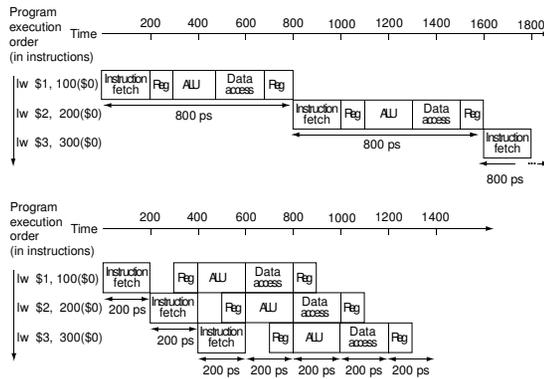
1

**Basic Pipelining Wrap-up
(from Slide Set 20)**

2

Pipelining

- Improve performance by increasing instruction throughput



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

3

Big Picture

- Remember the single-cycle implementation
 - Inefficient because low utilization of hardware resources
 - Each instruction takes one long cycle
- Two possible ways to improve on this:

	Multicycle	Pipelined
Clock cycle time (vs. single cycle)		
Amount of hardware used (vs. single cycle)		
Split instruction into multiple stages (1 per cycle)?		
Each stage has its own set of hardware?		
How many instructions executing at once?		

4

Pipelining and Beyond

5

Example – Multiple Issue

How many cycles does it take for this code to execute on a 2-issue CPU?

```
add $t0, $t1, $t2
lw  $s1, 0($s2)
add $t0, $t0, $t4
sw  $s1, 0($s3)
```

Answer?

7

Exploiting More ILP

- ILP = _____
(parallelism within a single program)
- How can we exploit more ILP?
 1. _____
(Split execution into many stages)
 2. _____
(Start executing more than one instruction each cycle)

6

Multiple Issue Processors

- Key metric: CPI → IPC
- Key questions:
 1. What set of instructions can be issued together?
 2. Who decides which instructions to issue together?
 - Static multiple issue
 - Dynamic multiple issue

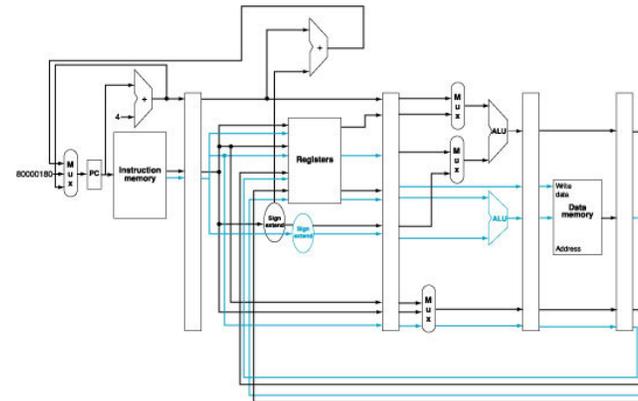
8

Multiple Issue Processors

- What extra hardware do we need to do Static Multiple Issue?
- What else for Dynamic Multiple Issue?

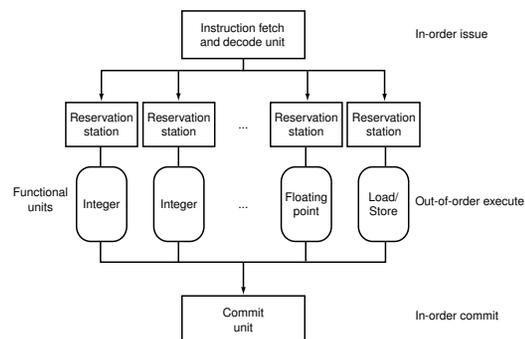
9

Example – MIPS Static Multiple Issue



10

Example – Dynamic Multiple Issue Scheduling



11

Exercise #1

Assume you must execute the following instructions in order. In any one cycle you can issue at most one integer op and one load or store. Show the resultant pipeline diagram. What's the total number of cycles? If you can't issue an instruction on a certain cycle, wait for the next cycle.

```
lw $t0, 0($s2)
sub $s1, $t0, $s3
lw $t2, 0($s2)
add $a0, $a1, $a2
add $a0, $a0, $a3
```

12

Exercise #2

Use same assumptions as with Exercise #1, but first schedule the code to try and eliminate stalls. Show the new pipeline diagram and total number of cycles.

```
lw  $t0, 0($s2)
sub  $s1, $t0, $s3
lw  $t2, 0($s2)
add  $a0, $a1, $a2
add  $a0, $a0, $a3
```

13

Exercise #4

- Look ahead at the slide for Idea #4 – loop unrolling. What is the possible bug?

15

Exercise #3: Static vs. Dynamic Multiple Issue

- Which do you think has been commercially successful – static or dynamic issue? Why?

14

Ideas for improving Multiple Issue

1. Non-blocking caches
2. Speculation
3. Register renaming
4. Loop unrolling

16

Idea #3: Register renaming

```
lw $t0, 0($s0)
sw $t0, 4($s0)
lw $t0, 0($s2)
sw $t0, 4($s2)
```

Problem?

Solution?

Idea #4: Loop unrolling

```
Loop: lw $t0, 0($s1)      Loop: lw $t0, 0($s1)
      sw $t0, 0($s2)      lw $t1, 4($s1)
      addi $s1, $s1, -4    lw $t2, 8($s1)
      addi $s2, $s2, -4    lw $t3, 12($s1)
      bne $s1, $zero, Loop sw $t0, 0($s2)
                                sw $t1, 4($s2)
                                sw $t2, 8($s2)
                                sw $t3, 12($s2)
                                addi $s1, $s1, -16
                                addi $s2, $s2, -16
                                bne $s1, $zero, Loop
```

Why is this a good idea?

Chapter 6 Summary

