Pipelining
Performance Measurements

- **Cycle Time**: Time in between clock ticks
- **Latency**: Time to finish a complete job, start to finish
- **Throughput**: Average jobs completed per unit time
- **CyclesPerJob**: Number of cycles between finishing jobs.
Goals

- Faster clock rate
- Use machine more efficiently
- No longer execute only one instruction at a time
Laundry

- Laundry-o-matic washes, dries & folds
- **Wash**: 30 min
- **Dry**: 40 min
- **Fold**: 20 min
- It switches them internally with no delay
- How long to complete 1 load?
Laundry

- **Laundry-o-matic** washes, dries & folds
- **Wash**: 30 min
- **Dry**: 40 min
- **Fold**: 20 min
- It switches them internally with no delay
- How long to complete 1 load? **90 min**
Laundry-o-Matic – Single Cycle

<table>
<thead>
<tr>
<th>Load</th>
<th>0</th>
<th>30</th>
<th>60</th>
<th>90</th>
<th>120</th>
<th>150</th>
<th>180</th>
<th>210</th>
<th>240</th>
<th>270</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>W</td>
<td>D</td>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>W</td>
<td>D</td>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Minutes
Laundry-o-Matic – SingleCycle

Load

Minutes

W D F

W D F

W D F
Laundry-o-Matic

- Cycle Time: Clothing is switched every ____ minutes
- Latency: A single load takes a total of _____ minutes
- Throughput: A load completes each _____ minutes
- CyclesPerLoad: Every ____ cycles, a load completes
Laundry-o-Matic

- **Cycle Time**: Clothing is switched every 90 minutes
- **Latency**: A single load takes a total of _____ minutes
- **Throughput**: A load completes each _____ minutes
- **CyclesPerLoad**: Every ____ cycles, a load completes
Laundry-o-Matic

- **Cycle Time:** Clothing is switched every 90 minutes
- **Latency:** A single load takes a total of 90 minutes
- **Throughput:** A load completes each ______ minutes
- **CyclesPerLoad:** Every ____ cycles, a load completes
Laundry-o-Matic

- Cycle Time: Clothing is switched every 90 minutes
- Latency: A single load takes a total of 90 minutes
- Throughput: A load completes each 90 minutes
- CyclesPerLoad: Every ___ cycles, a load completes
Laundry-o-Matic

- **Cycle Time:** Clothing is switched every 90 minutes
- **Latency:** A single load takes a total of 90 minutes
- **Throughput:** A load completes each 90 minutes
- **CyclesPerLoad:** Every 1 cycles, a load completes
Pipelined Laundry

• Split the laundry-o-matic into a washer, dryer, and folder (what a concept)
• Moving the laundry from one to another takes 6 minutes
We have to include time to switch *stages*
Pipelined Laundry

Load

Minutes
Two loads can not be in Dryer at the same time.
Pipelined Laundry

Switch all loads at the same time

Load

Minutes

0 30 60 90 120 150 180 210 240 270
Pipelined Laundry

Minutes

Load

0 30 60 90 120 150 180 210 240 270

W
D
F
W
D
F
W
D
F
Pipelined Laundry

• Cycle Time: Clothing is switched every ____ minutes
• Latency: A single load takes a total of ______ minutes
• Throughput: A load completes each ______ minutes
• CyclesPerLoad: Every ____ cycles, a load completes
Pipelined Laundry

- **Cycle Time**: Clothing is switched every 46 minutes
- **Latency**: A single load takes a total of _____ minutes
- **Throughput**: A load completes each _____ minutes
- **CyclesPerLoad**: Every ____ cycles, a load completes
Pipelined Laundry

- **Cycle Time:** Clothing is switched every 46 minutes
- **Latency:** A single load takes a total of 138 minutes
- **Throughput:** A load completes each ______ minutes
- **CyclesPerLoad:** Every ____ cycles, a load completes
Pipelined Laundry

- Cycle Time: Clothing is switched every 46 minutes
- Latency: A single load takes a total of 138 minutes
- Throughput: A load completes each 46 minutes
- CyclesPerLoad: Every ____ cycles, a load completes
Pipelined Laundry

- **Cycle Time:** Clothing is switched every 46 minutes
- **Latency:** A single load takes a total of 138 minutes
- **Throughput:** A load completes each 46 minutes
- **CyclesPerLoad:** Every 1 cycles, a load completes
Single-Cycle vs Pipelined

- ______ has the higher cycle time
- ______ has the higher clock rate
- ______ has the higher single-load latency
- ______ has the higher throughput
- ______ has the higher CPL (Cycles per Load)
- More stages makes a ______ clock rate
Single-Cycle vs Pipelined

- Single has the higher cycle time
- _______ has the higher clock rate
- _______ has the higher single-load latency
- _______ has the higher throughput
- _______ has the higher CPL (Cycles per Load)
- More stages makes a _______ clock rate
Single-Cycle vs Pipelined

- Single has the higher cycle time
- Pipelined has the higher clock rate
- _______ has the higher single-load latency
- _______ has the higher throughput
- _______ has the higher CPL (Cycles per Load)
- More stages makes a _______ clock rate
Single-Cycle vs Pipelined

- Single has the higher cycle time
- Pipelined has the higher clock rate
- Pipelined has the higher single-load latency
- _________ has the higher throughput
- _________ has the higher CPL (Cycles per Load)
- More stages makes a ________ clock rate
Single-Cycle vs Pipelined

- Single has the higher cycle time
- Pipelined has the higher clock rate
- Pipelined has the higher single-load latency
- Pipelined has the higher throughput
- _______ has the higher CPL (Cycles per Load)
- More stages makes a _______ clock rate
Single-Cycle vs Pipelined

- Single has the higher cycle time
- Pipelined has the higher clock rate
- Pipelined has the higher single-load latency
- Pipelined has the higher throughput
- Neither has the higher CPL (Cycles per Load)
- More stages makes a _______ clock rate
Single-Cycle vs Pipelined

• Single has the higher cycle time
• Pipelined has the higher clock rate
• Pipelined has the higher single-load latency
• Pipelined has the higher throughput
• Neither has the higher CPL (Cycles per Load)
• More stages makes a Higher clock rate
Obstacles to speedup in Pipelining

1. 
2. 

Ideal cycle time w/out above limitations with n stage pipeline:
Obstacles to speedup in Pipelining

1. Uneven Stages
2. 

Ideal cycle time w/out above limitations with n stage pipeline:
Obstacles to speedup in Pipelining

1. Uneven Stages
2. Pipeline Register Delay

Ideal cycle time w/out above limitations with n stage pipeline:
Obstacles to speedup in Pipelining

1. Uneven Stages
2. Pipeline Register Delay

Ideal cycle time w/out above limitations with n stage pipeline:

- OldCycleTime / n
Example

- Washing = 45
- Drying = 120
- Folding = 15
- Switching = 5

- What is the latency for one load of laundry?
- What is the latency for three loads?
Creating Stages

- Fetch – get instruction
- Decode – read registers
- Execute – use ALU
- Memory – access memory
- WriteBack – write registers
add $s0, $0, $0
lw $s1, 0($t0)
sw $s2, 0($t1)
or $s3, $s4, $t3
lw $s1, 0($t0)  add $s0, $0, $0

add $s0, $0, $0
lw $s1, 0($t0)
sw $s2, 0($t1)
or $s3, $s4, $t3
sw $s2, 0($t1)  lw $s1, 0($t0)  add $s0, $0, $0

add $s0, $0, $0
lw $s1, 0($t0)
sw $s2, 0($t1)
or $s3, $s4, $t3
or $s3, $s4, $t3  sw $s2, 0($t1)  lw $s1, 0($t0)  add $s0, $0, $0

add $s0, $0, $0
lw $s1, 0($t0)
sw $s2, 0($t1)
or $s3, $s4, $t3
add $s0, $0, $0
lw $s1, 0($t0)
sw $s2, 0($t1)
add $s0, $0, $0
or $s3, $s4, $t3
or $s3, $s4, $t3  sw $s2, 0($t1)  lw $s1, 0($t0)

Time->

add $s0, $0, $0
lw $s1, 0($t0)
sw $s2, 0($t1)
or $s3, $s4, $t3
IF    ID    MEM    WB

or $s3, $s4, $t3  sw $s2, 0($t1)

1       2       3       4       5        6      7       8

MEM    ID    MEM    WB

MEM    ID    MEM    WB

MEM    ID    MEM    WB

add $s0, $0, $0
lw $s1, 0($t0)
sw $s2, 0($t1)
or $s3, $s4, $t3
add $s0, $0, $0
lw $s1, 0($t0)
sw $s2, 0($t1)
or $s3, $s4, $t3

or $s3, $s4, $t3
add $s0, $0, $0
lw $s1, 0($t0)
sw $s2, 0($t1)
or $s3, $s4, $t3

The machine in cycle 4
add $s0, $0, $0
lw $s1, 0($t0)
sw $s2, 0($t1)
or $s3, $s4, $t3

The machine in cycle 5
In what cycle was $s1 written?

In what cycle was $s4 read?

In what cycle was the Add executed?

add $s0, $0, $0

lw $s1, 0($t0)

sw $s2, 0($t1)

or $s3, $s4, $t3
In what cycle was $s1 written? 6

In what cycle was $s4 read?

In what cycle was the Add executed?

```
add $s0, $0, $0
lw $s1, 0($t0)
sw $s2, 0($t1)
or $s3, $s4, $t3
```
In what cycle was $s1 written? 6

In what cycle was $s4 read? 5

In what cycle was the Add executed?

```
add $s0, $0, $0
lw $s1, 0($t0)
sw $s2, 0($t1)
or $s3, $s4, $t3
```
In what cycle was $s1 written? 6

In what cycle was $s4 read? 5

In what cycle was the Add executed? 3
Performance Analysis

- Measurements related to our machine
- Job = single instruction
- Latency: Time to finish a complete ______________, start to finish.
- Throughput: Average ______________ completed per unit time.

- Which is more important for reducing program execution time?
Performance Analysis

- Measurements related to our machine
- Job = single instruction
- Latency: Time to finish a complete instruction start to finish.
- Throughput: Average ______________ completed per unit time.

- Which is more important for reducing program execution time?
Performance Analysis

• Measurements related to our machine
• Job = single instruction
• Latency: Time to finish a complete instruction start to finish.
• Throughput: Average number of instructions completed per unit time.

• Which is more important for reducing program execution time?
Pipeline Registers

- Named for two stages they separate
- Store all data corresponding to lines that go through them

- **IF/ID**
  - 32b instruction
  - 32b nPC

- **ID/EX**
  - 32b register
  - 32b register
  - 32b immediate field
  - 32b nPC

- **EX/MEM**
  - Zero
  - 32b ALU result
  - 32b nPC
  - 32b register value

- **MEM/WB**
  - 32b ALU result
  - 32b memory value
Register File

- Only takes half of a cycle to read or write to register file
- Convention:
  - Read 2nd half of cycle
  - Write 1st half of cycle
Machine Comparison

<table>
<thead>
<tr>
<th>Fetch</th>
<th>Decode</th>
<th>Execute</th>
<th>Memory</th>
<th>WriteBack</th>
</tr>
</thead>
<tbody>
<tr>
<td>2ns</td>
<td>1ns</td>
<td>2ns</td>
<td>2ns</td>
<td>1ns</td>
</tr>
</tbody>
</table>

0.1 ns pipeline register delay

**Single-Cycle Implementation**

Clock cycle time: ______ ns
Latency of a single instruction: ______ ns
Throughput for machine: ______ inst/ns

**Pipelined Implementation**

Clock cycle time: ______ ns
Latency of a single instruction: ______ ns
Throughput for machine: ______ inst/ns
## Machine Comparison

<table>
<thead>
<tr>
<th>Fetch</th>
<th>Decode</th>
<th>Execute</th>
<th>Memory</th>
<th>WriteBack</th>
</tr>
</thead>
<tbody>
<tr>
<td>2ns</td>
<td>1ns</td>
<td>2ns</td>
<td>2ns</td>
<td>1ns</td>
</tr>
</tbody>
</table>

0.1 ns pipeline register delay

### Single-Cycle Implementation

- **Clock cycle time:** 8 ns
- **Latency of a single instruction:** ______ ns
- **Throughput for machine:** ______ inst/ns

### Pipelined Implementation

- **Clock cycle time:** ______ ns
- **Latency of a single instruction:** ______ ns
- **Throughput for machine:** ______ inst/ns
## Machine Comparison

<table>
<thead>
<tr>
<th>Fetch</th>
<th>Decode</th>
<th>Execute</th>
<th>Memory</th>
<th>Write Back</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 ns</td>
<td>1 ns</td>
<td>2 ns</td>
<td>2 ns</td>
<td>1 ns</td>
</tr>
</tbody>
</table>

0.1 ns pipeline register delay

### Single-Cycle Implementation

- **Clock cycle time:** 8 ns
- **Latency of a single instruction:** 8 ns
- **Throughput for machine:** _____ inst/ns

### Pipelined Implementation

- **Clock cycle time:** _____ ns
- **Latency of a single instruction:** _____ ns
- **Throughput for machine:** _____ inst/ns
## Machine Comparison

<table>
<thead>
<tr>
<th>Fetch</th>
<th>Decode</th>
<th>Execute</th>
<th>Memory</th>
<th>WriteBack</th>
</tr>
</thead>
<tbody>
<tr>
<td>2ns</td>
<td>1ns</td>
<td>2ns</td>
<td>2ns</td>
<td>1ns</td>
</tr>
</tbody>
</table>

0.1 ns pipeline register delay

### Single-Cycle Implementation

- Clock cycle time: 8 ns
- Latency of a single instruction: 8 ns
- Throughput for machine: 1/8 inst/ns

### Pipelined Implementation

- Clock cycle time: ______ ns
- Latency of a single instruction: ______ ns
- Throughput for machine: ______ inst/ns
Machine Comparison

<table>
<thead>
<tr>
<th>Fetch</th>
<th>Decode</th>
<th>Execute</th>
<th>Memory</th>
<th>WriteBack</th>
</tr>
</thead>
<tbody>
<tr>
<td>2ns</td>
<td>1ns</td>
<td>2ns</td>
<td>2ns</td>
<td>1ns</td>
</tr>
</tbody>
</table>

0.1 ns pipeline register delay

Single-Cycle Implementation

  Clock cycle time: 8 ns
  Latency of a single instruction: 8 ns
  Throughput for machine: 1/8 inst/ns

Pipelined Implementation

  Clock cycle time: 2.1 ns
  Latency of a single instruction: _____ ns
  Throughput for machine: _____ inst/ns
Machine Comparison

<table>
<thead>
<tr>
<th>Fetch</th>
<th>Decode</th>
<th>Execute</th>
<th>Memory</th>
<th>WriteBack</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 ns</td>
<td>1 ns</td>
<td>2 ns</td>
<td>2 ns</td>
<td>1 ns</td>
</tr>
</tbody>
</table>

0.1 ns pipeline register delay

Single-Cycle Implementation

- Clock cycle time: 8 ns
- Latency of a single instruction: 8 ns
- Throughput for machine: 1/8 inst/ ns

Pipelined Implementation

- Clock cycle time: 2.1 ns
- Latency of a single instruction: 2.1*5=10.5 ns
- Throughput for machine: _____ inst/ ns
Machine Comparison

<table>
<thead>
<tr>
<th>Fetch</th>
<th>Decode</th>
<th>Execute</th>
<th>Memory</th>
<th>WriteBack</th>
</tr>
</thead>
<tbody>
<tr>
<td>2ns</td>
<td>1ns</td>
<td>2ns</td>
<td>2ns</td>
<td>1ns</td>
</tr>
</tbody>
</table>

0.1 ns pipeline register delay

Single-Cycle Implementation
- Clock cycle time: 8 ns
- Latency of a single instruction: 8 ns
- Throughput for machine: 1/8 inst/ns

Pipelined Implementation
- Clock cycle time: 2.1 ns
- Latency of a single instruction: 2.1*5=10.5 ns
- Throughput for machine: 1 / 2.1 inst/ns
Example 2 – How do we speed up pipelined machine?

<table>
<thead>
<tr>
<th>Fetch</th>
<th>Decode</th>
<th>Execute</th>
<th>Memory</th>
<th>Writeback</th>
</tr>
</thead>
<tbody>
<tr>
<td>6ns</td>
<td>4ns</td>
<td>8ns</td>
<td>10ns</td>
<td>4ns</td>
</tr>
<tr>
<td>0.1 ns pipelined register delay</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Single cycle: 1 / ns
Pipelined: 1 / ns
Example 2 – How do we speed up pipelined machine?

Fetch Decode Execute Memory Writeback
6ns  4ns  8ns  10ns  4ns
0.1 ns pipelined register delay

Single cycle:  1 / 32 inst / ns
Pipelined:  1 / 10.1 inst / ns
## Example 2 – Split more stages

<table>
<thead>
<tr>
<th>Stage</th>
<th>Delay (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fetch</td>
<td>6</td>
</tr>
<tr>
<td>Decode</td>
<td>4</td>
</tr>
<tr>
<td>Execute</td>
<td>8</td>
</tr>
<tr>
<td>Memory</td>
<td>10</td>
</tr>
<tr>
<td>Writeback</td>
<td>4</td>
</tr>
<tr>
<td>Pipelined</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Which stage(s) should we split?

___________ and ___________
Example 2 – Split more stages

Fetch  Decode  Execute  Memory  Writeback
6ns  4ns  8ns  10ns  4ns
0.1 ns pipelined register delay

Which stage(s) should we split?
Memory and __________
Example 2 – Split more stages

Fetch Decode Execute Memory Writeback
  6ns  4ns  8ns  10ns  4ns
0.1 ns pipelined register delay

Which stage(s) should we split?
Memory and Execute
Example 2 – After Split

F       D       X1     X2     M1     M2     WB
___ns  ___ns  ___ns  ___ns  ___ns  ___ns  ___ns  ___ns
0.1 ns pipelined register delay

Single cycle: 1 / ns
Pipelined: 1 / ns
### Example 2 – After Split

<table>
<thead>
<tr>
<th>F</th>
<th>D</th>
<th>X1</th>
<th>X2</th>
<th>M1</th>
<th>M2</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 ns</td>
<td>4 ns</td>
<td>___ns</td>
<td>___ns</td>
<td>___ns</td>
<td>___ns</td>
<td>4 ns</td>
</tr>
</tbody>
</table>

0.1 ns pipelined register delay

**Single cycle:** 1 / ns  
**Pipelined:** 1 / ns
### Example 2 – After Split

<table>
<thead>
<tr>
<th>F</th>
<th>D</th>
<th>X1</th>
<th>X2</th>
<th>M1</th>
<th>M2</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>6  ns</td>
<td>4  ns</td>
<td>4  ns</td>
<td>4  ns</td>
<td>___ns</td>
<td>___ns</td>
<td>4  ns</td>
</tr>
</tbody>
</table>

0.1 ns pipelined register delay

**Single cycle:** \( \frac{1}{\text{ns}} \)

**Pipelined:** \( \frac{1}{\text{ns}} \)
Example 2 – After Split

<table>
<thead>
<tr>
<th>F</th>
<th>D</th>
<th>X1</th>
<th>X2</th>
<th>M1</th>
<th>M2</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 ns</td>
<td>4 ns</td>
<td>4 ns</td>
<td>4 ns</td>
<td>5 ns</td>
<td>5 ns</td>
<td>4 ns</td>
</tr>
</tbody>
</table>

0.1 ns pipelined register delay

Single cycle: 1 / ns
Pipelined: 1 / ns
Example 2 – After Split

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>X1</th>
<th></th>
<th>X2</th>
<th></th>
<th>M1</th>
<th></th>
<th>M2</th>
<th></th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>D</td>
<td>6</td>
<td>ns</td>
<td>4</td>
<td>ns</td>
<td>4</td>
<td>ns</td>
<td>4</td>
<td>ns</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>ns</td>
<td>5</td>
<td>ns</td>
<td>4</td>
<td>ns</td>
<td>4</td>
<td>ns</td>
<td>0.1</td>
</tr>
</tbody>
</table>

0.1 ns pipelined register delay

Single cycle: 1 / 32 ns
Pipelined: 1 / ns
### Example 2 – After Split

<table>
<thead>
<tr>
<th>F</th>
<th>D</th>
<th>X1</th>
<th>X2</th>
<th>M1</th>
<th>M2</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 ns</td>
<td>4 ns</td>
<td>4 ns</td>
<td>4 ns</td>
<td>4 ns</td>
<td>5 ns</td>
<td>5 ns</td>
</tr>
</tbody>
</table>

0.1 ns pipelined register delay

Single cycle: 1 / 32 ns  
Pipelined: 1 / 6.1 ns
In what cycle does the add write $s0?
In what cycle does the or read $s0?

add $s0, $0, $0
or $s3, $s0, $t3
sw $s2, 0($t1)
and $s6, $s4, $t3

Incorrect Execution
Easy Right? Not so fast.

In what cycle does the add write $s0?  
1st half of cycle 5

In what cycle does the or read $s0?

1

add $s0, $0, $0
or $s3, $s0, $t3
sw $s2, 0($t1)
and $s6, $s4, $t3
Easy Right? Not so fast.

In what cycle does the add write $s0? 1^{st} \text{ half of cycle } 5
In what cycle does the or read $s0? 2^{nd} \text{ half of cycle } 3

add $s0, 0, 0$ or $s3, s0, s3$
sw $s2, 0(t1)$ and $s6, s4, t3$
Easy Right? Not so fast.

In what cycle does the `add $s0, $0, $0` write $s0? 1st half of cycle 5

In what cycle does the `or $s3, $s0, $t3` read $s0? 2nd half of cycle 3

Ahhhh! Values cannot pass backwards in time
Correct, Slow Execution

Easy Right? Not so fast.

In what cycle does the add write $s0? 1\text{st} \text{ half of cycle 5}
In what cycle does the or read $s0? 2\text{nd} \text{ half of cycle 5}

Stall - wasted cycles

add $s0, 0, 0
or $s3, $s0, $t3
sw $s2, 0($t1)
and $s6, $s4, $t3

Time->
Easy Right?  Not so fast.

In what cycle does the add write $s0? 1^{st} half of cycle 5
In what cycle does the or read $s0? 2^{nd} half of cycle 5

Stall - wasted cycles

add $s0, $0, $0
or $s3, $s0, $t3
sw $s2, 0($t1)
and $s6, $s4, $t3

Only Register File rd/wr in half a cycle. All other stages take a full cycle – this is because of shared hardware
Pipelined Machine

Fetch

Instruction Memory

Read Addr Out Data

Decode

Register File

src1 src1data

src2 src2data

destreg

destdata

op/fun

rs

rts

rd

imm

Execute

Memory

Pipeline Register

(Writeback)

Addr Out Data

Data Memory

In Data

PC

<< 2

<< 2

4

<< 2

32

16

Sign Ext

32
Incorrect Execution caused by Data Hazard

In what cycle does the `lw` write `$s0`?
In what cycle does the `or` read `$s0`?

```
lw  $s0, 0($t4)
and $s6, $s4, $t3
sw  $s2, 0($t1)
```

or `$s3, $s0, $t3`
and `$s6, $s4, $t3`
Incorrect Execution caused by Data Hazard

In what cycle does the lw write $s0? 1st half of cycle 5
In what cycle does the or read $s0?

lw $s0, 0($t4)
or $s3, $s0, $t3
sw $s2, 0($t1)
and $s6, $s4, $t3
Incorrect Execution caused by Data Hazard

In what cycle does the lw write $s0? 1\text{st} \text{ half of cycle} \ 5
In what cycle does the or read $s0? 2\text{nd} \text{ half of cycle} \ 3
Incorrect Execution caused by Data Hazard

In what cycle does the lw write $s0?  1\text{st half of } 5$
In what cycle does the or read $s0?  2\text{nd half of } 3$

lw $s0, 0($t4)
or $s3, $s0, $t3$
sw $s2, 0($t1)
and $s6, $s4, $t3$

Arrow to the left is information passed backwards in time
Data Hazard

In what cycle does the lw write $s0?  1st half of cycle.
In what cycle does the or read $s0?  2nd half of cycle.

lw $s0, 0($t4)
or $s3, $s0, $t3
sw $s2, 0($t1)
and $s6, $s4, $t3

Remember!!! Only WB and ID take ½ cycle. All other stages take a full cycle!!!!!!!!!
Barriers to pipelined performance

- Uneven stages
- Pipeline register delays
Barriers to pipelined performance

- Uneven stages
- Pipeline register delays
- Data Hazards
Barriers to pipeline performance

• Uneven stages
• Pipeline register delays
• **Data Hazards**
  - An instruction depends on the result of a previous instruction still in the pipeline
Default: Stall

In what cycle does the lw write $s0? 1^{st} half of cycle 5

In what cycle does the or read $s0? 2^{nd} half of cycle 5

Stall - wasted cycles

Remember!!! Only WB and ID take $\frac{1}{2}$ cycle. All other stages take a full cycle!!!!!!!!!
Solution 1: Data Forwarding

In what cycle is $s0 calculated in the machine?
In what cycle is $s0 used in the machine?

1w $s0, 0($t4)

or $s3, $s0, $t3

sw $s2, 0($t1)

and $s6, $s4, $t3
Solution 1: Data Forwarding

In what cycle is $s0 **calculated** in the machine? **End** of cycle 4

In what cycle is $s0 **used**?

1. lw $s0, 0($t4)
2. or $s3, $s0, $t3
3. sw $s2, 0($t1)
4. and $s6, $s4, $t3
In what cycle is $s0 calculated in the machine? **End** of cycle 4
In what cycle is $s0 used? **beginning** of cycle 4

1w $s0, 0($t4)
or $s3, $s0, $t3
sw $s2, 0($t1)
and $s6, $s4, $t3

**Solution 1: Data Forwarding**
In what cycle is $s0 calculated in the machine? **end** of cycle 4
In what cycle is $s0 used? **beginning** of cycle 5

1w $s0, 0($t4)
or $s3, $s0, $t3
sw $s2, 0($t1)
and $s6, $s4, $t3

Solution 1: Data Forwarding
Data-Forwarding
Where are those wires?

Fetch

Decode

Execute

Memory

Instruction Memory

PC

Register File

destreg
destdata

src1 src1data

src2 src2data

op/fun

rs

rt

rd

imm

Pipeline Register

(Writeback)

addr

out data

Addr Out Data

Sign Ext

16

32

Data Memory

In Data

<< 2

<< 2
Data-Forwarding
Where are those wires?

Fetch | Decode | Execute | Memory

Instruction Memory
- PC
- Read Addr
- Out Data

Register File
- src1
- src1data
- src2
- src2data
- rs
- rt
- rd
- imm
- destreg
- destdata
- op/fun

Data Memory
- Addr
- Out Data
- In Data

Pipeline Register (Writeback)
- 16 Sign Ext
- 32
Data Forwarding
Example 2

Draw the timing diagram with data forwarding
Draw arrows to indicate data passing through forwarding

lw $t0, 0($s0)
addi $t0, $t0, 1
add $s2, $s2, $t0
sw $s2, 0($s0)
Solution 2: Instruction Reordering (Before reordering)

Stall - wasted cycles

1w $s0, 0($t4)  
or $s3, $s0, $t3  
sw $s2, 0($t1)  
and $s6, $s4, $t3
Solution 2: Instruction Reordering (After Reordering)

1w $s0, 0($t4)
sw $s2, 0($t1)
and $s6, $s4, $t3
or $s3, $s0, $t3
Who reorders instructions?

- Static scheduling
  - Compiler
  - Simpler, but does not know when caches miss or loads/stores are to the same locations

- Dynamic scheduling
  - Hardware
  - More complicated, but has all knowledge
Solution 2: Instruction Reordering

lw $s0, 0($t4)
or $s3, $s0, $t3
sw $s3, 0($t1)
and $s0, $s4, $t3
Solution 2: Instruction Reordering

lw $s0, 0($t4)
sw $s3, 0($t1)
and $s0, $s4, $t3
or $s3, $s0, $t3

Is this the same execution?!?
Solution 2: Instruction Reordering

Is this the same execution?!?

lw $s0, 0($t4)
sw $s3, 0($t1)
and $s0, $s4, $t3
or $s3, $s0, $t3
Pipelined Machine

Fetch  Decode  Execute  Memory

Instruction Memory

register

src1  src1data

src2  src2data

Registerr File
destreg
destdata

op/fun

rs

rt

rd

imm

<< 2

<< 2

Pipeline Register

(Writeback)

Addr

Out Data

Data Memory

In Data

Sign Ext

16

32

PC

Read Addr

Out Data

4

<< 2