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 – SingleCycle

Load

Minutes

0 30 60 90 120 150 180 210 240 270

1 2 3
Laundry-o-Matic – SingleCycle

Load

1
2
3

Minutes

0 30 60 90 120 150 180 210 240 270

W D F
Laundry-o-Matic – SingleCycle

Load

1

2

3

Minutes

0 30 60 90 120 150 180 210 240 270

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

0  30  60  90  120  150  180  210  240  270
Two loads can not be in Dryer at the same time.
Pipelined Laundry

Switch all loads at the same time
Pipelined Laundry

Minutes

Load

Minutes
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. 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:

\[ \text{OldCycleTime} / n \]
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
Pipelined Machine

Fetch

Instruction Memory

Decode

Register File

External Memory

Execute

Pipeline Register

(writeback)
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
IF    ID    MEM    WB

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

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

Fetch

Instruction Memory

Read Addr Out Data

Decode

Register File

src1 src1data

src2 src2data

destreg

destdata

op/fun

rs

rt

rd

imm

Execute

Memory

Pipeline Register

(Writeback)

Data Memory

Addr Out Data

In Data

PC
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>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1 ns pipeline register delay</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **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>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: _____ 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>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: ____ inst/ns
## Machine Comparison

<table>
<thead>
<tr>
<th></th>
<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>
<td></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:** $\frac{1}{8}$ inst/ns

### Pipelined Implementation

- **Clock cycle time:** 2.1 ns
- **Latency of a single instruction:** $2.1 \times 5 = 10.5$ ns
- **Throughput for machine:** $\frac{1}{2.1}$ inst/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 / ns
Pipelined: 1 / 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>
</tbody>
</table>

0.1 ns pipelined register delay

Single cycle: \( \frac{1}{32} \text{ inst / ns} \)
Pipelined: \( \frac{1}{10.1} \text{ inst / ns} \)
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?
_________ 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

<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>___ns</td>
<td>___ns</td>
<td>___ns</td>
<td>___ns</td>
<td>___ns</td>
<td>___ns</td>
<td>___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>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>___ 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:** 1 / ns
- **Pipelined:** 1 / ns
Example 2 – After Split

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

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>5 ns</td>
<td>5 ns</td>
<td>4 ns</td>
</tr>
</tbody>
</table>

0.1 ns pipelined register delay

Single cycle: \( \frac{1}{32} \) 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:** $\frac{1}{32}$ ns  
**Pipelined:** $\frac{1}{6.1}$ ns
Incorrect Execution

Easy Right? Not so fast.

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
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?

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} \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, $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 3

Ahhhh! Values can not pass backwards in time
Correct, Slow Execution

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

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

Stall - wasted cycles

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

Correct, Slow Execution

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

- What can we try to reduce data hazards or their effect?
Default (do nothing): Stall

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? 2nd half of cycle 5

Stall - wasted cycles

add $s0, $0, $0
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?
In what cycle is $s0 used in the machine?

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

Time->
In what cycle is $s0 \textbf{calculated} in the machine? End of cycle 4
In what cycle is $s0 \textbf{used}?

1w $s0, 0($t4)
or $s3, $s0, $t3
sw $s2, 0($t1)
and $s6, $s4, $t3
In what cycle is $s0 \textbf{calculated} in the machine? \textbf{End} of cycle 4

In what cycle is $s0 \textbf{used}?

\textbf{beginning} of cycle 4

\textbf{Solution 1: Data Forwarding}

lw $s0, 0($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

lw $s0, 0($t4)
or $s3, $s0, $t3
sw $s2, 0($t1)
and $s6, $s4, $t3
Data-Forwarding
Where are those wires?
Data-Forwarding
Where are those wires?

Fetch

Decode

Execute

Memory

Pipeline Register

(Writeback)
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

lw $s0, 0($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

1w $s0, 0($t4)
or $s3, $s0, $t3
sw $s3, 0($t1)
and $s0, $s4, $t3
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
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
How Are Data Hazards Detected

A.K.A. The Official Lawson Johnson Aside
First: Pipelined Control
Recall: 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
Pipelined Control

• PC is written on each clock cycle
• No need for signals to pipeline registers, since they are written on each clock cycle
• Observation: Each control line associated with a component active in only one stage of the pipeline
  • So we can divide control lines into five groups, according to pipeline stage
Pipelined Control
The Hazard Detection Pseudocode

• “Code” operates during ID stage
• Looks something like this

```c
if (ID/EX.Memread AND
    ((ID/EX.RegisterRt = IF/ID.RegisterRs) OR
    (ID/EX.RegisterRt = IF/ID.RegisterRt))) {
    Stall the pipeline
}
```
The Hazard Detection Pseudocode

• “Code” operates during ID stage
• Looks something like this

```c
if (ID/EX.Memread AND ((ID/EX.RegisterRt = IF/ID.RegisterRs) OR (ID/EX.RegisterRt = IF/ID.RegisterRt))) {
    Stall the pipeline
}
```
How is the Pipeline Stalled?

- If ID stage is stalled, IF stage must also be stalled
  - Otherwise we lose the instruction that was to be fetched next
  - In concrete terms: PC and IF/ID pipeline registers are prevented from changing
How is the Pipeline Stalled?

• And “back half” (EX/M/WB) phases must be doing “nothing
  • Like restarting the wash but letting dryer continue to tumble empty
  • Accomplished with NOP instruction(s)
  • Turns out: deasserting all control signals into EX/M/WB states creates a NOP instruction in those stages
  • Note these are percolated forward (doing nothing), which is what we want