

CSE 141 – Computer Architecture Spring 2005

Lectures 13 Pipeline Hazards

Pramod V. Argade
May 11, 2005

Announcements

- **Reading Assignment**
 - Chapter 6: Enhancing Performance with Pipelining
 - 6.1 - 6.3
- **Homework 5:**
 - 6.1, 6.2, 6.3, 6.6, 6.14, 6.17, 6.21, 6.22
 - Additional Problems: 6.20
- **Quiz**
 - When:** Mon., May 16th, First 10 minutes of the class
 - Topic:** Pipeline Hazards, Chapter 6 **Need:** Paper, pen

Course Schedule

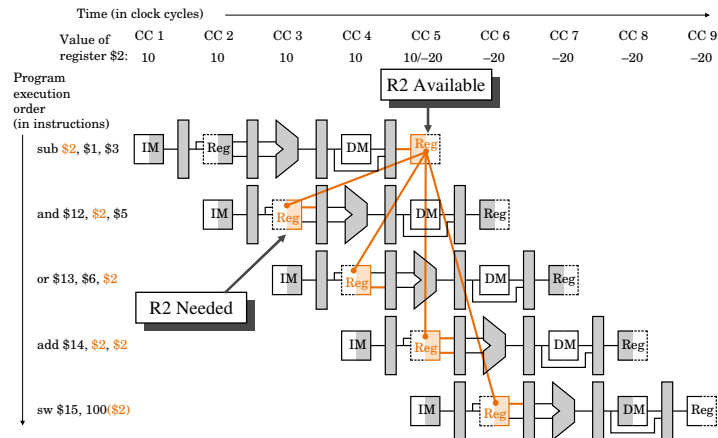
| Lecture # | Date | Day | Lecture Topic | Quiz Topic | Homework Due |
|-----------|------|-----------|---|------------------|--------------|
| 1 | 3/28 | Monday | Introduction, Ch. 1 | - | - |
| 2 | 3/30 | Wednesday | Performance, Ch. 4 | - | - |
| 3 | 4/4 | Monday | ISA, Ch. 2 | Performance | #1 |
| 4 | 4/6 | Wednesday | Arithmetic, Ch. 3 | - | - |
| 5 | 4/11 | Monday | Arithmetic, Ch. 3 | ISA | #2 |
| 6 | 4/13 | Wednesday | Single cycle CPU, Ch. 5 | - | - |
| 7 | 4/18 | Monday | Single cycle CPU, Ch. 5 | Arithmetic | #3 |
| 8 | 4/20 | Wednesday | Multi-cycle CPU, Ch. 5 | - | - |
| 9 | 4/25 | Monday | Multi-cycle CPU, Ch. 5 | Single Cycle CPU | #4 |
| 10 | 4/27 | Wednesday | Review for the Midterm | - | - |
| | 5/2 | Monday | Mid-term Exam Center 101 | - | - |
| 11 | 5/4 | Wednesday | Exceptions, Ch. 5 and Pipelining, Ch. 6 | - | - |
| 12 | 5/9 | Monday | Pipelining, Ch. 6 | - | - |
| 13 | 5/11 | Wednesday | Data and control hazards, Ch. 6 | - | - |
| 14 | 5/16 | Monday | Data and control hazards, Ch. 6 | Pipeline Hazards | #5 |
| 15 | 5/18 | Wednesday | Memory & cache design, Ch. 7 | - | - |
| 16 | 5/23 | Monday | Memory & cache design, Ch. 7 | Cache | #6 |
| 17 | 5/25 | Wednesday | Virtual Memory & cache design, Ch. 7 | - | - |
| No Class | 5/30 | Monday | Memorial Day Holiday | - | - |
| 18 | 6/1 | Wednesday | Course Review | - | - |
| | 6/10 | Friday | 7 - 10 PM Final Exam | - | - |

Would our pipeline design work in any case?

- What happens when...
 - add \$3, \$10, \$11
 - lw \$8, 1000(\$3)
 - sub \$11, \$8, \$7

Data Hazards

- When a result is needed in the pipeline before it is available, a “data hazard” occurs.



- Result of SUB instruction not available until CC5 or later!**

Software Solutions to Data Hazards

- Have compiler guarantee no hazards
 - Rearrange code to remove hazard
 - Not possible every time

- Insert “nops”

- 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)
    
```

nop
nop

←

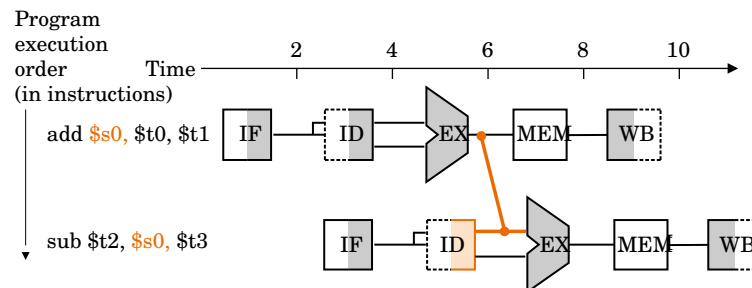
- Problem: Data hazards are very common!
 - “nops” really slows us down!

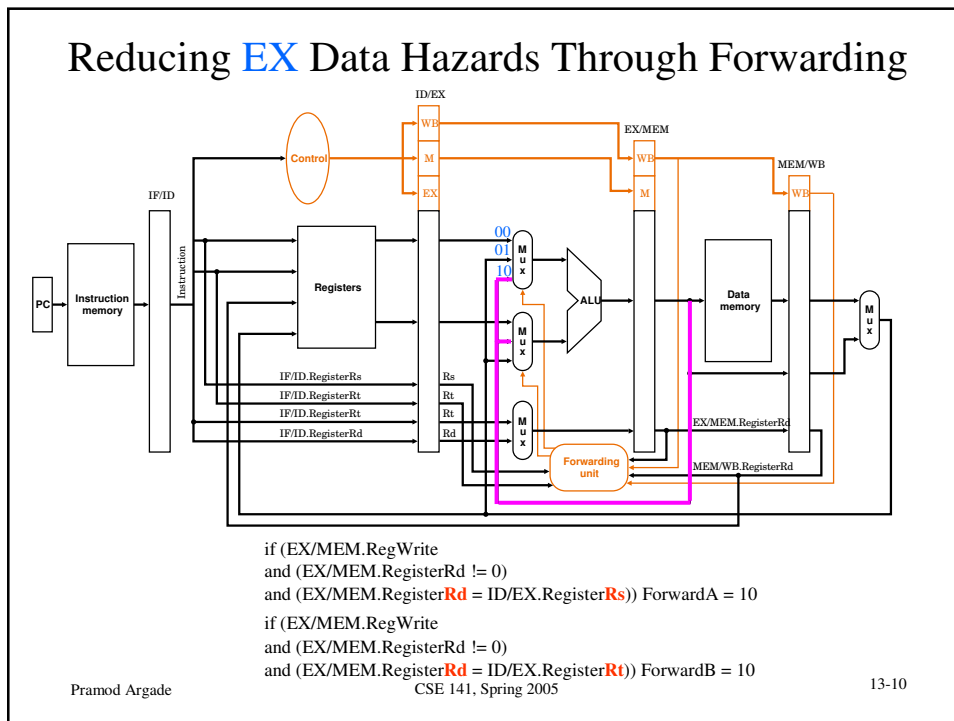
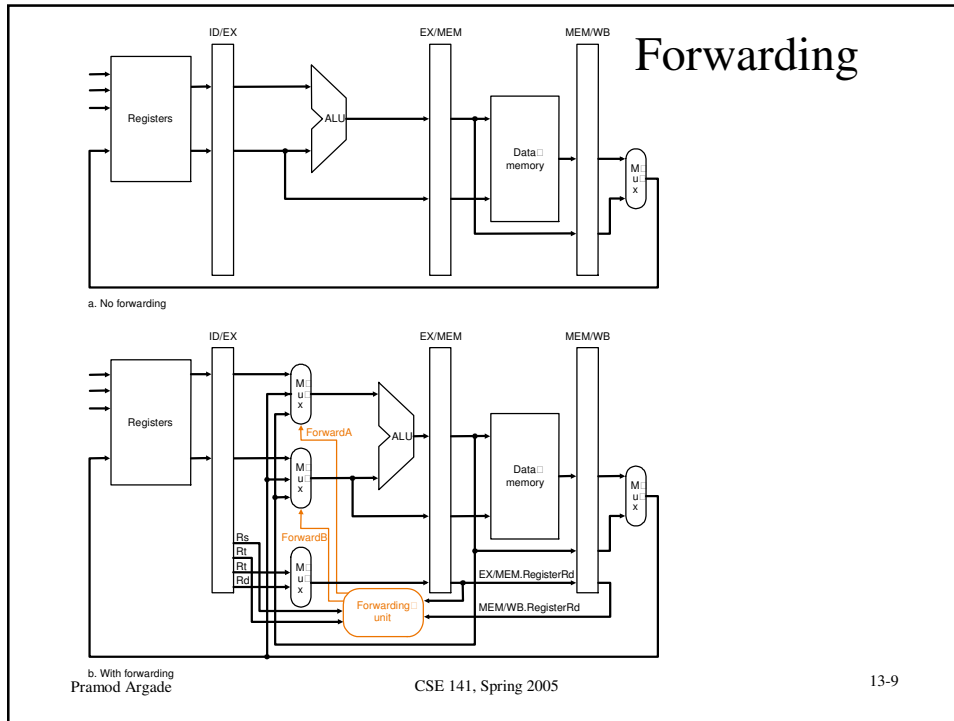
Hardware Solutions to Data Hazards

- Stall the pipeline (insert bubbles)
 - Data hazards are too common
 - Same as “nops”
 - Severe performance hit
- Forward the data as soon as it is available
 - Modify the pipeline to forward (bypass data)

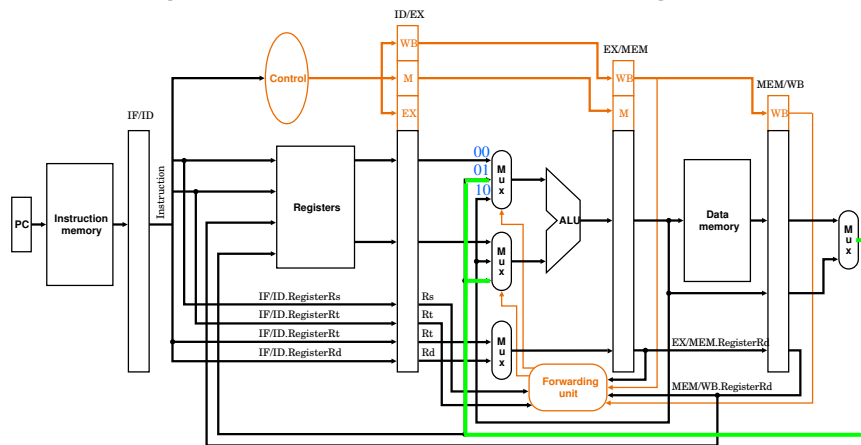
Forwarding

- Use temporary results, don't wait for them to be written





Reducing MEM Data Hazards Through Forwarding



```

if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd != 0)
and (MEM/WB.RegisterRd = ID/EX.RegisterRs)) ForwardA = 01
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd != 0)
and (MEM/WB.RegisterRd = ID/EX.RegisterRt)) ForwardB = 01
    
```

Pramod Argade

CSE 141, Spring 2005

13-11

Simultaneous EX/MEM Forwarding

- Consider following code

```

add $1, $1, $2
add $1, $1, $3
add $1, $1, $4
    
```

...

- Must forward from MEM stage
- Disable WB stage forwarding

```

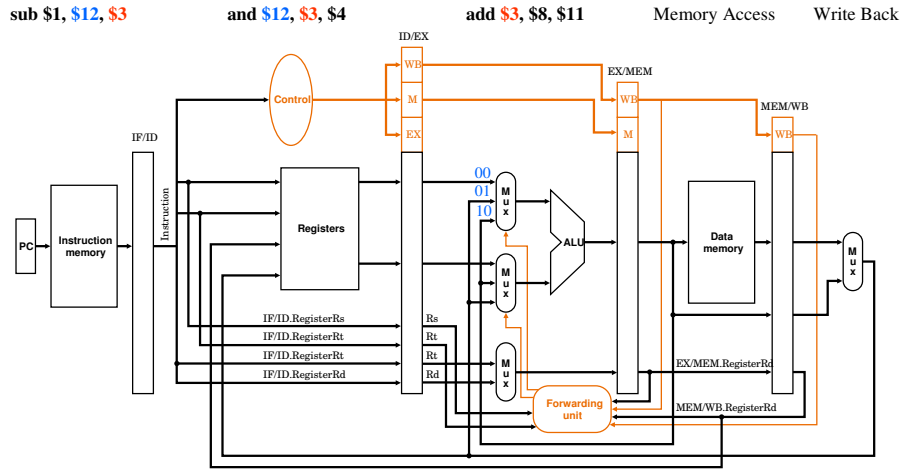
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd != 0)
and (EX/MEM.RegisterRd != ID/EX.RegisterRs)
and (MEM/WB.RegisterRd = ID/EX.RegisterRs)) ForwardA = 01
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd != 0)
and (EX/MEM.RegisterRd != ID/EX.RegisterRt)
and (MEM/WB.RegisterRd = ID/EX.RegisterRt)) ForwardB = 01
    
```

Pramod Argade

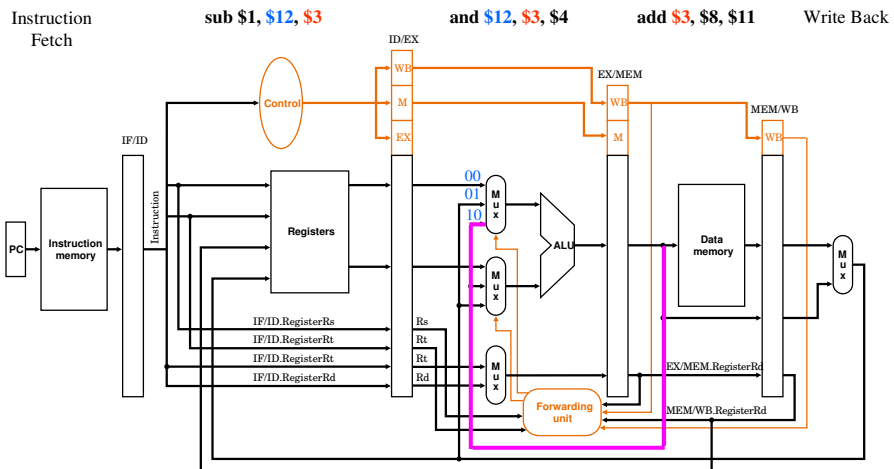
CSE 141, Spring 2005

13-12

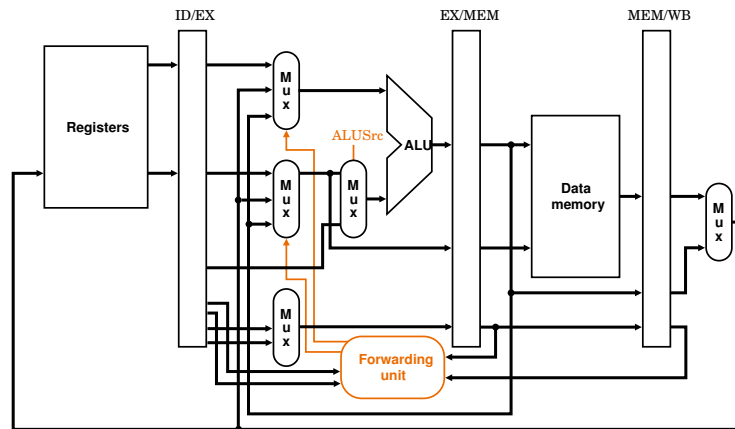
Forwarding in Action



Forwarding in Action



M \Rightarrow M Forwarding for LW \Rightarrow SW (Exercise 6.20 in the Textbook)

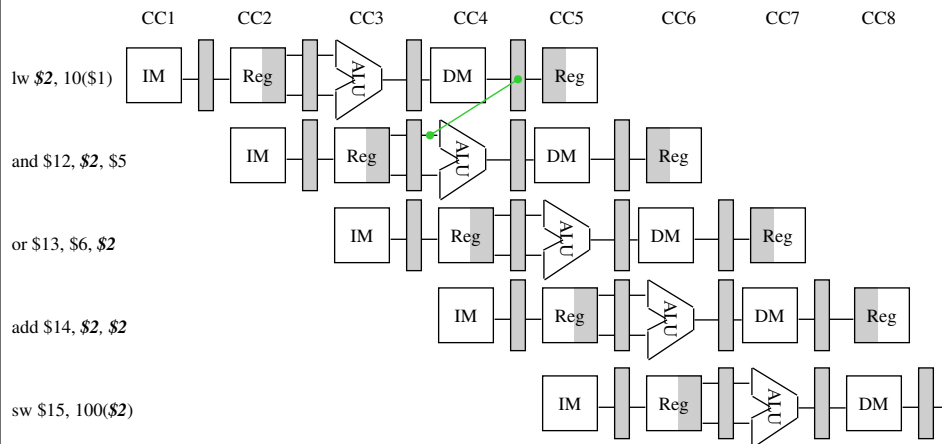


Forwarding does not eliminate Data Hazard in all cases

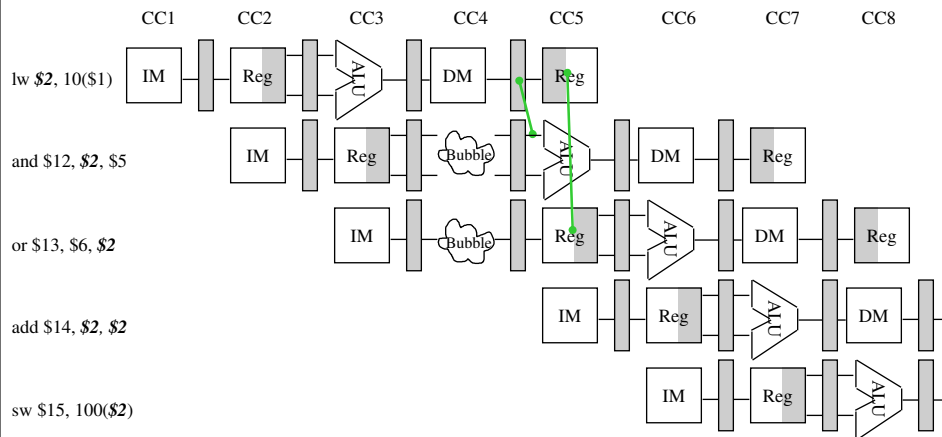
- Consider this code:

```
lw $2, 10($1)
and $12, $2, $5
or $13, $6, $2
add $14, $2, $2
sw $15, 100($2)
```

Data Hazard: Load followed by R-type



Eliminating Data Hazards via Forwarding and stalling

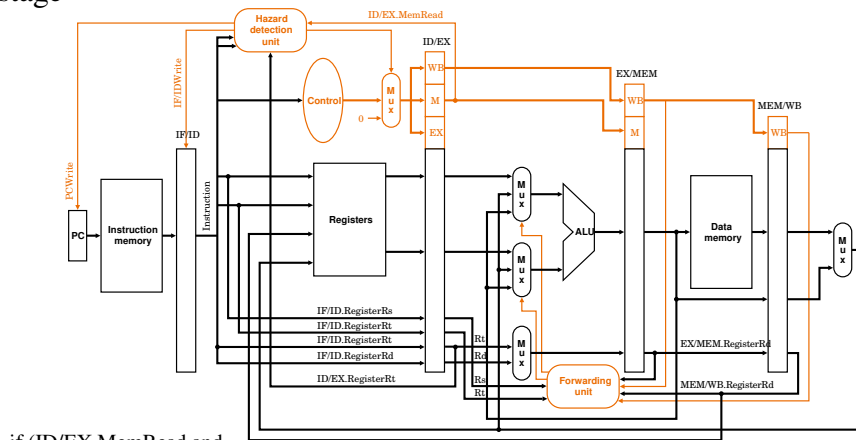


Pipeline Interlocks

- Not all data hazards can be handled by forwarding
- Pipeline Interlock or Hazard Detection Unit
 - detects a hazard and stalls the pipeline until the hazard is clear
- A stall creates a pipeline bubble:
 - Preventing the IF and ID stages from proceeding
 - don't write the PC (PCWrite = 0)
 - don't rewrite IF/ID register (IF/IDWrite = 0)
 - Inserting “nops”
 - set all control signals propagating to EX/MEM/WB to zero (inserts a no-op instruction)

Hazard Detection Unit

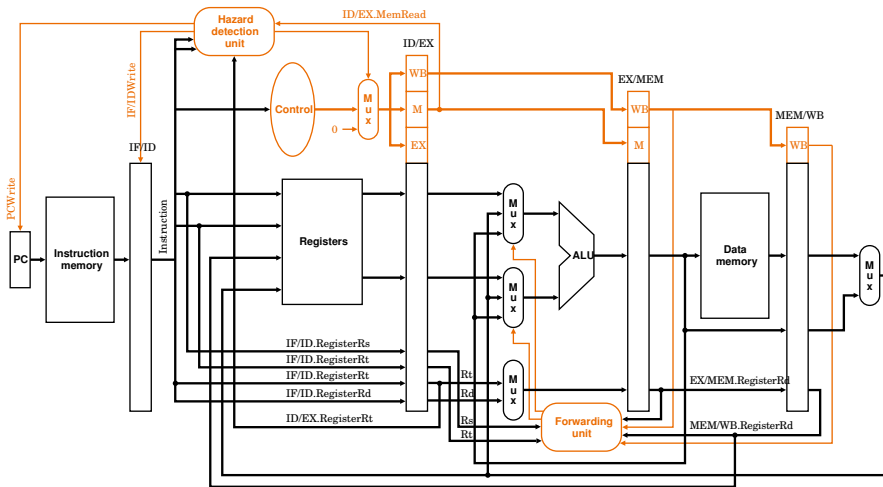
- We can stall the pipeline by keeping an instruction in the same stage



if (ID/EX.MemRead and
 ((ID/EX.RegisterRt = IF/ID.RegisterRs) or
 (ID/EX.RegisterRt = IF/ID.RegisterRt))
 then stall the pipeline → PCWrite = 0, IF/IDWrite = 0, EX/M/WB = 0

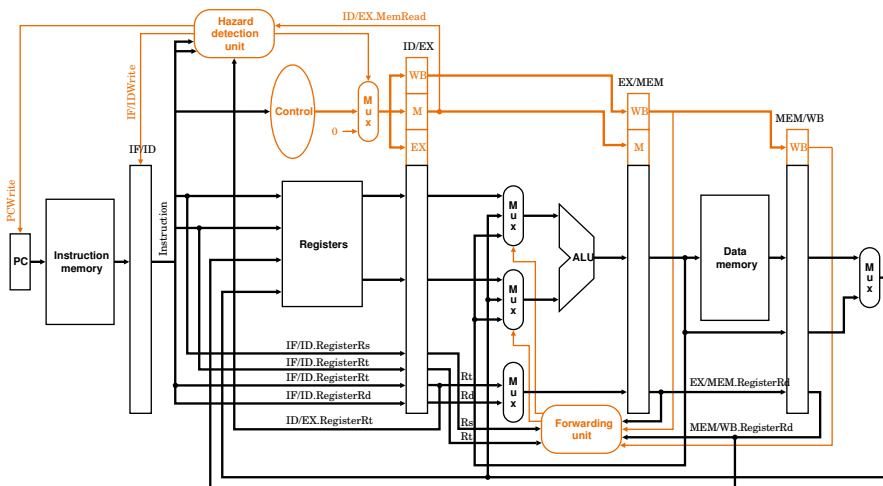
Hazard Detection Unit

and \$4, \$2, \$5 lw \$2, 20(\$1)



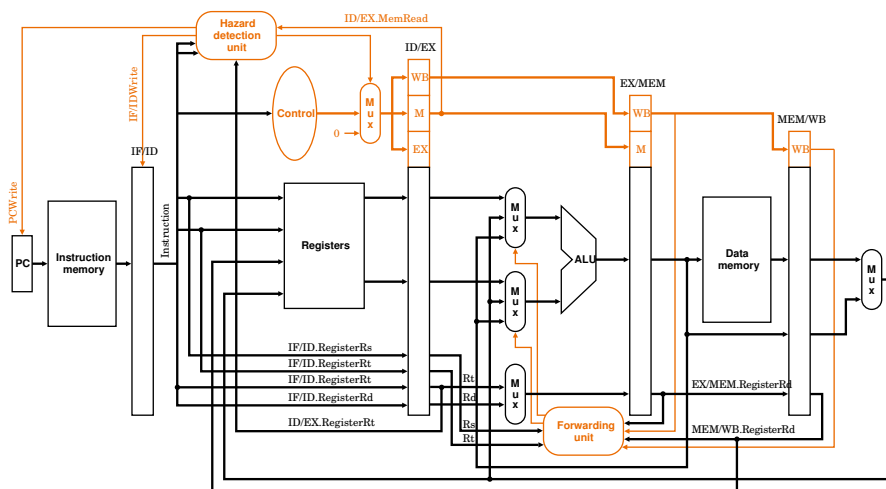
Hazard Detection Unit

and \$4, \$2, \$5 bubble lw \$2, 20(\$1)



Hazard Detection Unit

and \$4, \$2, \$5 bubble lw \$2, 20(\$1)



Pramod Argade

CSE 141, Spring 2005

13-25

Data Hazard Key Points

- Pipelining provides high throughput
- Data dependencies cause *data hazards*
- Data hazards can be solved by:
 - Software (nops)
 - Hardware data forwarding
 - Hardware pipeline stalling
- Our processor, and indeed all modern processors, use a combination of forwarding and stalling

Pramod Argade

CSE 141, Spring 2005

13-26