EE 3613: Computer Organization

Chapter 4: Pipelining - 2

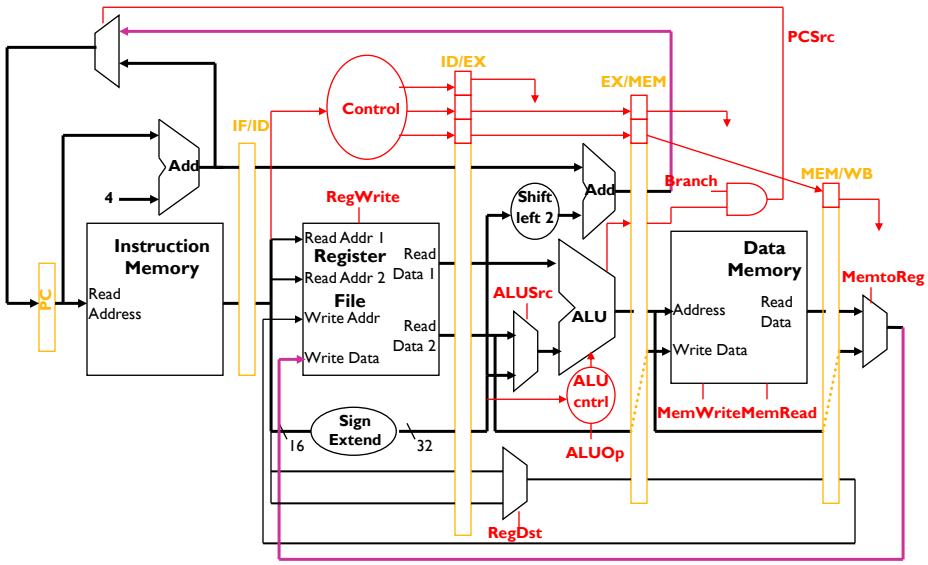
Avinash Karanth
Department of Electrical Engineering & Computer Science
Ohio University, Athens, Ohio 45701
E-mail: karanth@ohio.edu
Website: http://oucsace.cs.ohiou.edu/~avinashk/ee461a.htm
Acknowledgement: Mary J. Irwin, PSU; Srinivasan Ramasubramanian, UofA,

1

Course Administration

2

Review: MIPS Pipeline Data and Control Paths



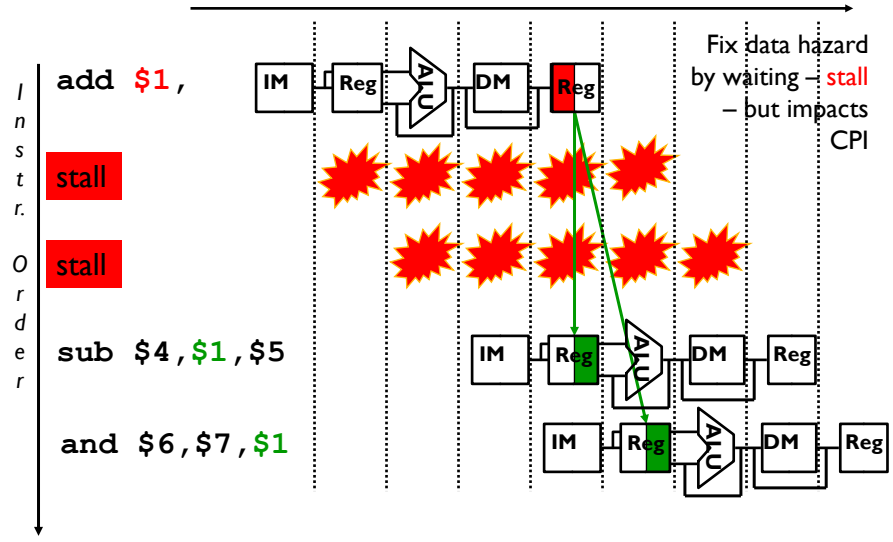
3

Control Settings

	EX Stage				MEM Stage			WB Stage	
	RegDst	ALU Op1	ALU Op0	ALUSrc	Brch	MemRead	Mem Write	RegWrite	Mem toReg
R	1	1	0	0	0	0	0	1	0
lw	0	0	0	1	0	1	0	1	1
sw	X	0	0	1	0	0	1	0	X
beq	X	0	1	0	1	0	0	0	X

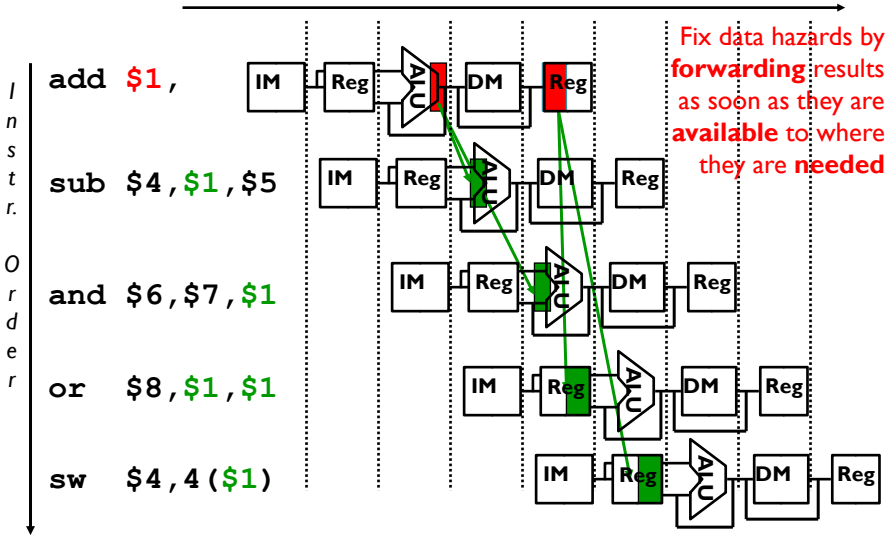
4

Review: One Way to "Fix" a Data Hazard



5

Review: Another Way to "Fix" a Data Hazard



6

Data Forwarding (aka Bypassing)

- Take the result from the earliest point that it exists in any of the pipeline state registers and forward it to the functional units (e.g., the ALU) that needs it in that cycle
- For ALU functional unit: the inputs can come from any pipeline register rather than just from ID/EX by
  - adding multiplexors to the inputs of the ALU
  - connecting the Rd write data in EX/MEM or MEM/WB to either (or both) of the EX's stage Rs and Rt ALU mux inputs
  - adding the proper control hardware to control the new muxes
- Other functional units may need similar forwarding logic (e.g., the DM)
- With forwarding can achieve a CPI of 1 even in the presence of data dependencies

7

Data Forwarding Control Conditions

1. EX/MEM hazard:

```
if (EX/MEM.RegWrite
and (EX/MEM.RegisterRd != 0)
and (EX/MEM.RegisterRd = ID/EX.RegisterRs))
      ForwardA = 10
if (EX/MEM.RegWrite
and (EX/MEM.RegisterRd != 0)
and (EX/MEM.RegisterRd = ID/EX.RegisterRt))
      ForwardB = 10
```

Forwards the result from the previous instr. to either input of the ALU

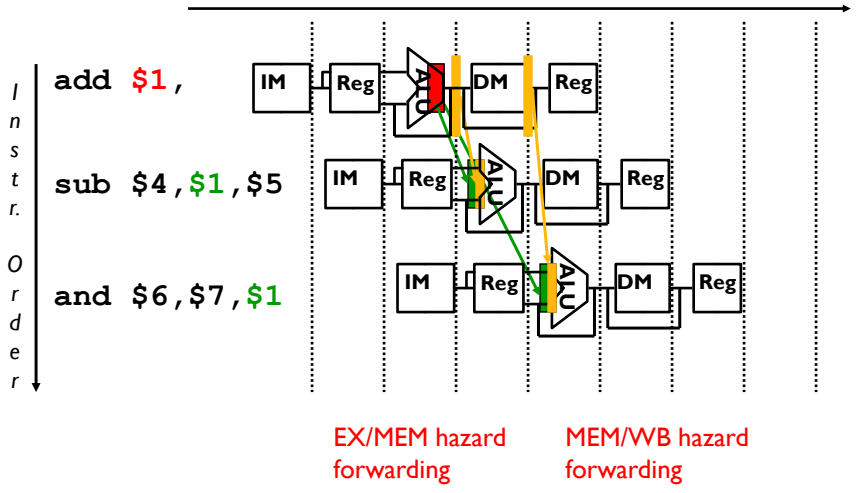
2. MEM/WB hazard:

```
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
```

Forwards the result from the second previous instr. to either input of the ALU

8

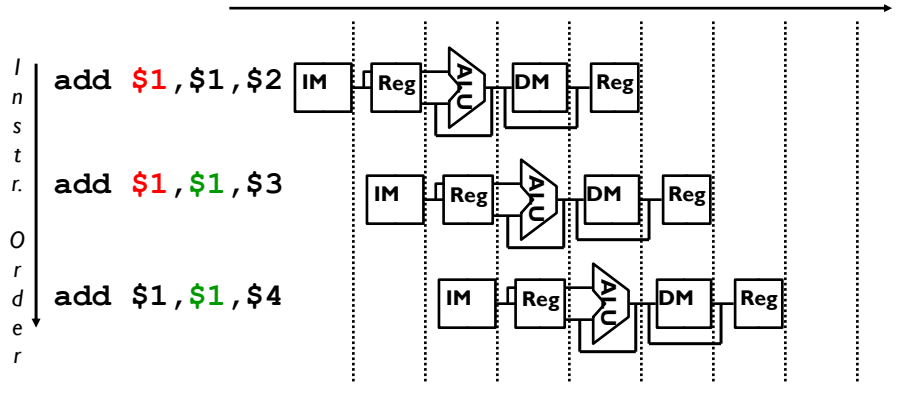
Forwarding Illustration



9

Yet Another Complication!

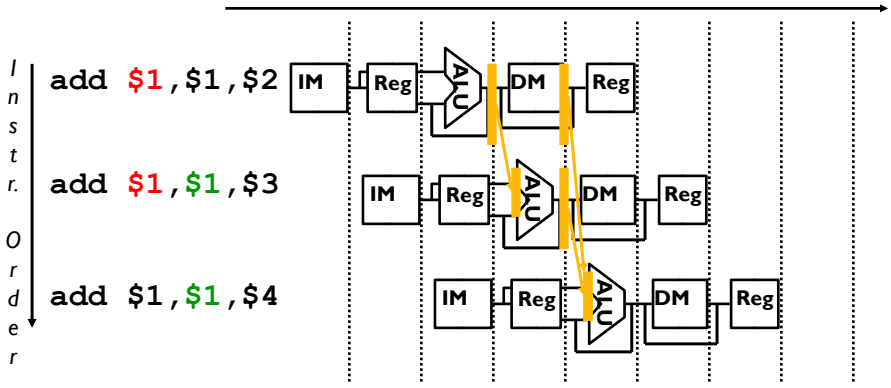
- Another potential data hazard can occur when there is a conflict between the result of the WB stage instruction and the MEM stage instruction – which should be forwarded?



10

Yet Another Complication!

- Another potential data hazard can occur when there is a conflict between the result of the WB stage instruction and the MEM stage instruction – which should be forwarded?



11

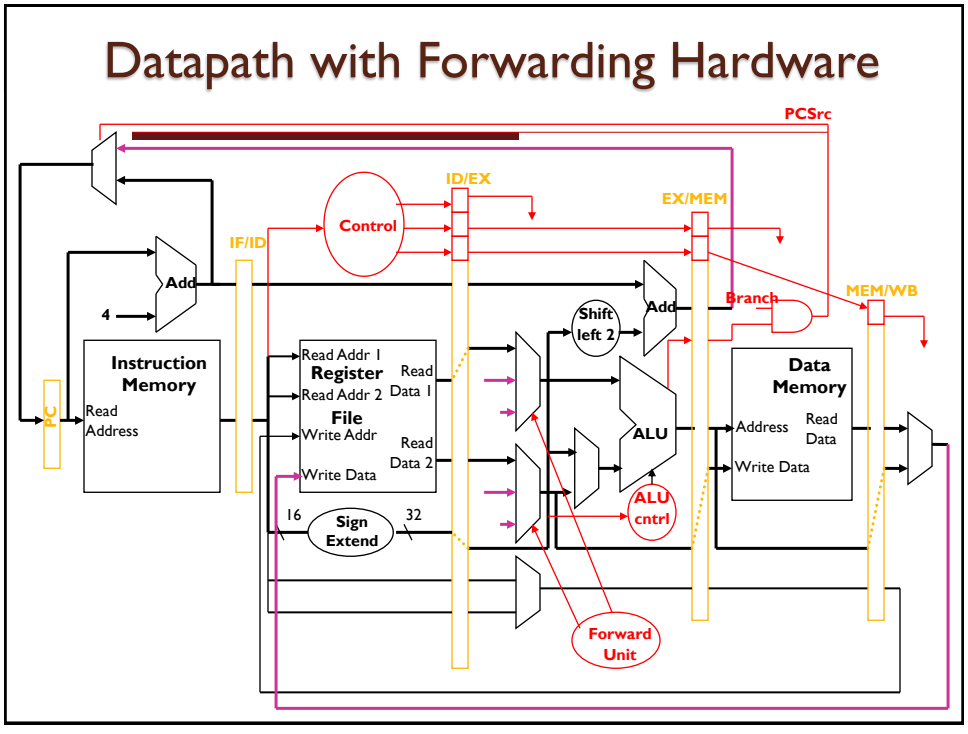
Corrected Data Forwarding Control Conditions

2. MEM/WB hazard:

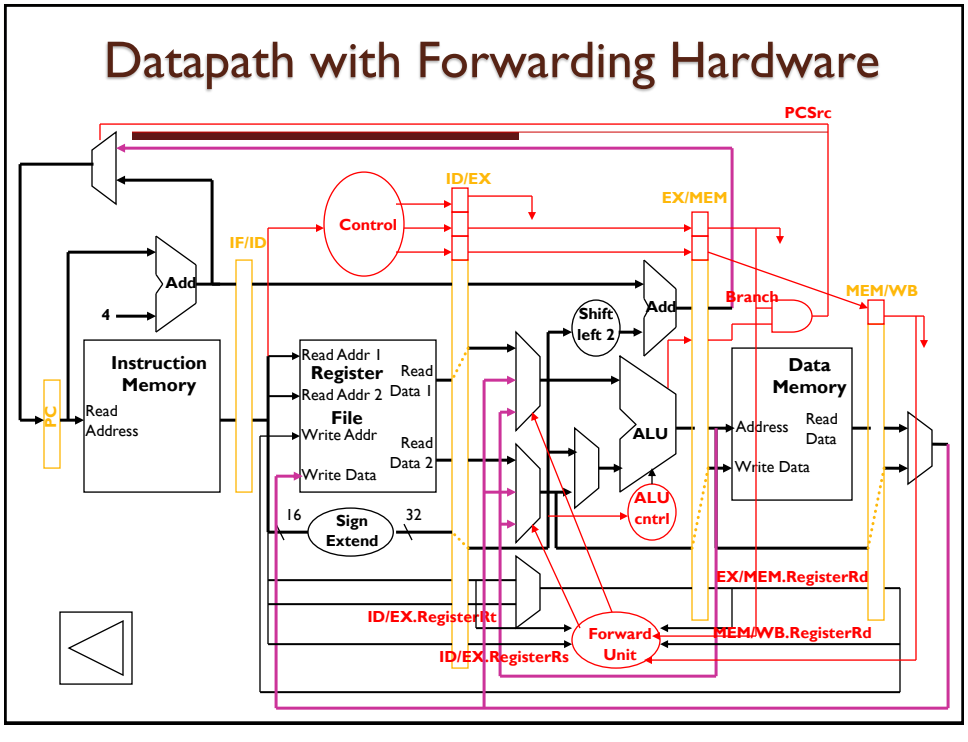
```
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd != 0)
and (EX/MEM.RegisterRd != ID/EX.RegisterRs)
and (MEM/WB.RegisterRd = ID/EX.RegisterRs))
      ForwardA = 01

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
```

12


Datapath with Forwarding Hardware
PCSrc
ID/EX
EX/MEM
MEM/WB
IF/ID
Control
Add
4
Branch
Shift left 2
Add
Instruction Memory
Read Address
PC
Read Addr 1
Register
Read Addr 2
File
Write Addr
Write Data
Read Data 1
Read Data 2
ALU
ALU cntrl
Data Memory
Address
Read Data
Write Data
16
Sign Extend
32
Forward Unit

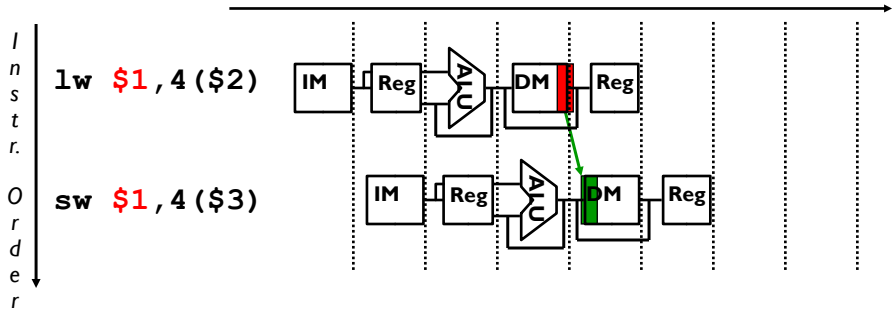
13


Datapath with Forwarding Hardware
PCSrc
ID/EX
EX/MEM
MEM/WB
IF/ID
Control
Add
4
Branch
Shift left 2
Add
Instruction Memory
Read Address
PC
Read Addr 1
Register
Read Addr 2
File
Write Addr
Write Data
Read Data 1
Read Data 2
ALU
ALU cntrl
Data Memory
Address
Read Data
Write Data
16
Sign Extend
32
EX/MEM.RegisterRd
ID/EX.RegisterRt
MEM/WB.RegisterRd
ID/EX.RegisterRs
Forward Unit

14

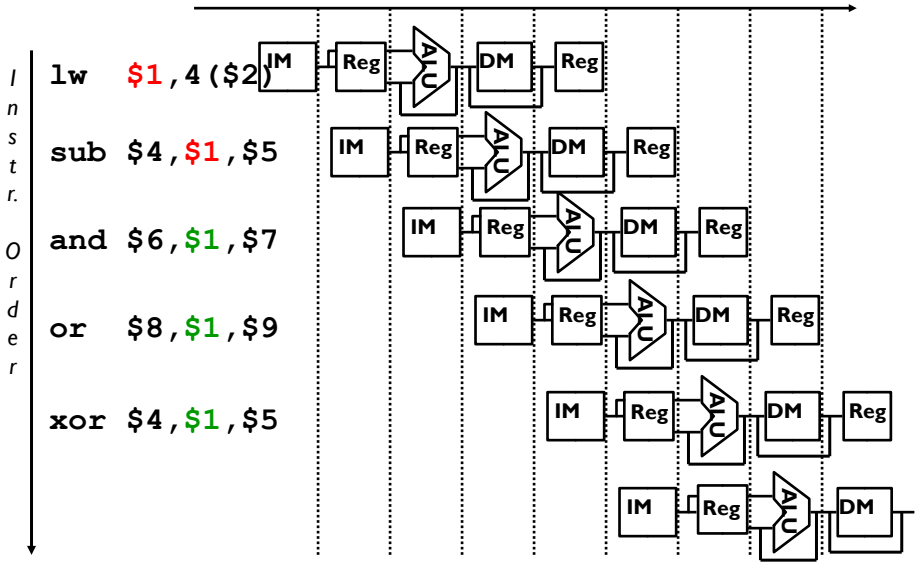
Memory-to-Memory Copies

- For loads immediately followed by stores (memory-to-memory copies) can avoid a stall by adding forwarding hardware from the MEM/WB register to the data memory input.
  - Would need to add a Forward Unit and a mux to the memory access stage



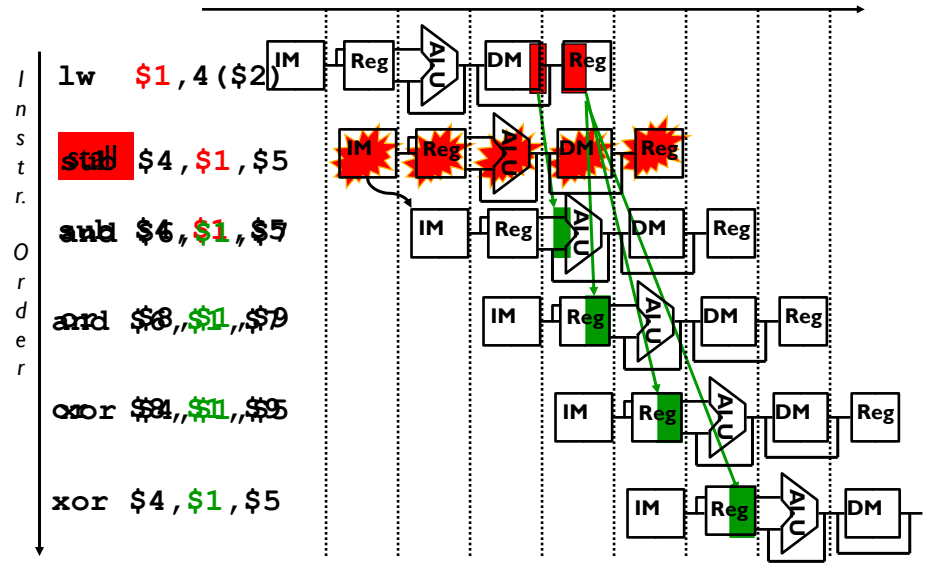
15

Forwarding with Load-use Data Hazards



16

Forwarding with Load-use Data Hazards



17

Load-use Hazard Detection Unit

- Need a Hazard detection Unit in the ID stage that inserts a stall between the load and its use

2. ID Hazard Detection

```
if (ID/EX.MemRead
and ((ID/EX.RegisterRt = IF/ID.RegisterRs)
or  (ID/EX.RegisterRt = IF/ID.RegisterRt)))
stall the pipeline
```

- The first line tests to see if the instruction now in the EX stage is a `lw`; the next two lines check to see if the destination register of the `lw` matches either source register of the instruction in the ID stage (the load-use instruction)
- After this one cycle stall, the forwarding logic can handle the remaining data hazards

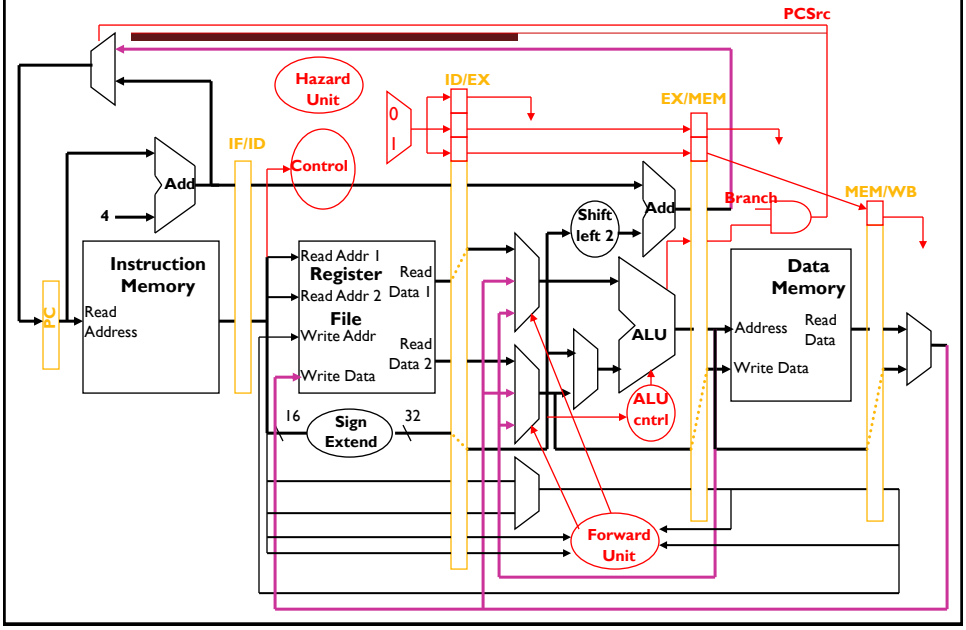
18

Stall Hardware

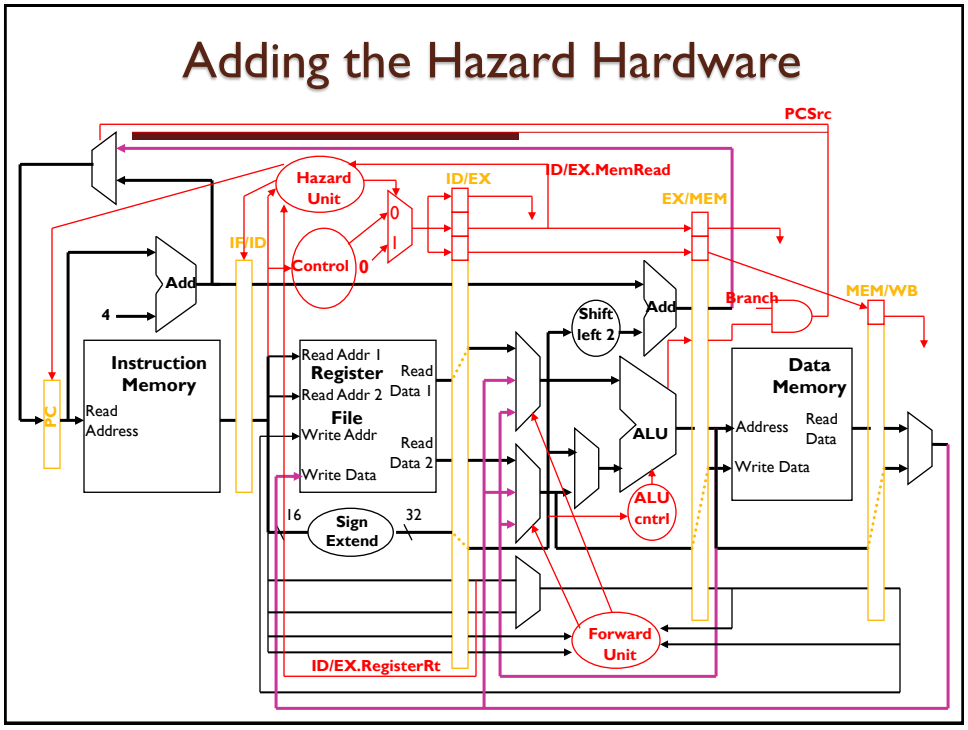
- Along with the Hazard Unit, we have to implement the stall
- Prevent the instructions in the IF and ID stages from progressing down the pipeline – done by preventing the PC register and the IF/ID pipeline register from changing
  - Hazard detection Unit controls the writing of the PC (`PC.write`) and IF/ID (`IF/ID.write`) registers
- Insert a "bubble" between the `lw` instruction (in the EX stage) and the load-use instruction (in the ID stage) (i.e., insert a `noop` in the execution stream)
  - Set the control bits in the EX, MEM, and WB control fields of the ID/EX pipeline register to 0 (`noop`). The Hazard Unit controls the mux that chooses between the real control values and the 0's.
- Let the `lw` instruction and the instructions after it in the pipeline (before it in the code) proceed normally down the pipeline

19

Adding the Hazard Hardware



20


Adding the Hazard Hardware
PCSrc
Hazard Unit
ID/EX
ID/EX.MemRead
EX/MEM
IF/ID
Control
0
0
1
Add
4
Instruction Memory
Read Address
Read Addr 1
Register
Read Addr 2
File
Write Addr
Write Data
Read Data 1
Read Data 2
16
Sign Extend
32
Shift left 2
Add
Branch
MEM/WB
ALU
ALU cntrl
Data Memory
Address
Read Data
Write Data
Forward Unit
ID/EX.RegisterRt

21