ECE 2300
Digital Logic & Computer Organization
Spring 2018

More Pipelined Microprocessor
Announcements

• Prelim 2 tonight, 7:30-9:00pm, PHL 101
Example: Branch Delay Slot

BEQ R2, R3, X
NOP
OR R4, R1, R3
X: AND R6, R1, R2
ADDI R7, R2, 3
SUB R6, R2, R2
...

• Can you identify one instruction to fill the branch delay slot?
Review: Load Instructions and Forwarding

- LW $R1, 0(R2)$
- OR $R4, R1, R3$
- SUB $R5, R2, R1$
- AND $R6, R1, R2$
- ADDI $R7, R7, 3$
Pipeline Control Unit (CU)
Pipeline Control Requirements

• **Generate control signals for each stage**
  - **IF**: PCJ
  - **EX**: MB, F
  - **MEM**: MW, MD
  - **WB**: LD

• **Detect forwarding conditions and generate MUX control signals**

• **Detect data hazards and insert pipeline bubbles**
  - Assumes no load delay slot defined in ISA

• **Assume branch delay slot defined in ISA**
Generating Control for Each Stage
Forwarding Unit (Partial)
R-type to R-type Forwarding

- **ADD** \( R1, R2, R3 \)
- **OR** \( R4, R1, R3 \)
- **SUB** \( R5, R2, R1 \)
- **AND** \( R6, R1, R2 \)

- **MEM→EX** : \( \text{MEM.DR} = (\text{EX.SA} || \text{EX.SB}) \)
- **WB→EX** : 
  1. \( \text{WB.DR} = (\text{EX.SA} || \text{EX.SB}) \) and
  2. there is no forwarding from MEM→EX in this cycle
- **WB→ID** : \( \text{WB.DR} = (\text{ID.SA} || \text{ID.SB}) \)
R-type to R-type Forwarding

ADD R1, R2, R3
OR R1, R1, R3
SUB R5, R2, R1
AND R6, R1, R2
ADDI R7, R1, 3
R-type to R-type Forwarding

- **MEM→EX**
  - MEM.DR == (EX.SA || EX.SB)

- **WB→EX**
  - WB.DR == (EX.SA || EX.SB) and
  - MEM.DR != (EX.SA || EX.SB)

- **WB→ID**
  - WB.DR == (ID.SA || ID.SB)
R-type to Branch Forwarding

- ADD $R1, R2, R3$
- OR $R4, R6, R3$
- BGEZ $R1, X$
- AND $R6, R5, R2$
- X:ADDI $R7, R7, 3$

• Other forwarding conditions to handle as well
Pipeline with Fwding + Branch HW in ID

Diagram showing the pipeline stages and hardware components, including control signals and data flow through the pipeline stages (IF/ID, ID/EX, EX/MEM, MEM/WB) with specific components like the CPU, Adder, RF, and ALU.
Data Hazards Requiring Bubbles

- Occur when instructions are too close together for forwarding to work
- Requires adding bubbles in the pipeline
- Data hazard conditions to detect and handle
  - Load followed by R-type
  - Load followed by I-type ALU instruction
  - Load followed by Load
  - Load followed by Store (two cases)
  - Load followed by Branch
  - ALU instruction followed by Branch
Example: Data Hazards w/ Forwarding

- **Assume HW forwarding and NO delay slot for load**

- Identify all data hazards in the following instruction sequences by circling each source register that is read before the updated value is written back

  LW R2, 0(R1)
  ADDI R3, R2, 1
  BEQ R2, R1, X
  SW R3, 4(R1)

  X: …
Load Followed by R-type Instruction

if (EX.Load && ID.R-type) {                      // Inst in EX is a Load and inst in ID is R-type
    if (EX.DR == (ID.SA || ID.SB)) {           // DR of Load matches SA or SB of inst in ID
        Insert NOP into EX in next cycle // Insert bubble
        Don’t Load IF/ID in next cycle     // Hold instruction in ID for a cycle
        Don’t Load PC in next cycle       // Hold instruction in IF for a cycle
    }
}
Load Followed by R-type Instruction

Lecture 19: 17

LW R1, 0(R2)
OR R4, R1, R3
SUB R5, R2, R1
AND R6, R1, R2
Load Followed by R-type Instruction

OR R4, R1, R3  LW R1, 0(R2)
Load Followed by R-type Instruction

Detection of hazard
Load Followed by R-type Instruction

```
LW R1,0(R2)  
OR R4,R1,R3  
SUB R5,R2,R1  
AND R6,R1,R2
```

```
SUB R5,R2,R1  
OR R4,R1,R3  
NOP  
LW R1,0(R2)
```

Hold these instructions

Insert bubble

Lecture 19: 20
Load Followed by R-type Instruction

Pipeline continues normally
R1 value forwarded from WB to EX and ID
Hazard Detection Unit (Partial)

- IF/ID: Instruction Fetch/Decode
- ID/EX: Instruction ID/Execution
- EX/MEM: Execute/Memory
- MEM/WB: Memory Write-Back
- ALU: Arithmetic Logic Unit
- RF: Register File
- Data RAM
- PCJ: Program Counter Jaguar
- CU: Control Unit
- +2: Adder
- MUX: Multiplexer
- LD: Load
- SA: Store A
- SB: Store B
- DR: Data Register
- D_in: Data Input
- RF: Register File
- MB, F: Memory Bus, Flags
- D_IN: Data Input
- MW, MD: Memory Write
- LD: Load

‘Clear’ puts 0 into ID/EX
Pipeline with Datapath and Control
Load Followed by Branch Instruction

LW R1, 0(R2)

BEQ R4, R1, X

Two bubbles are needed

- Also need to handle Load followed by branch two instructions apart
R-type to Store Forwarding

- **SUB** $R3, R1, R2
- **SW** $R3, 4(R1)$
- **ADD** $R4, R3, R2
Pipeline without Forwarding for Store

R[SB] not forwarded
Slight Pipeline Improvement for Stores

R[SB] forwarded from MEM or WB
Next Time

- H&H 8-8.3

Caches