CprE 381: Computer Organization and Assembly Level Programming

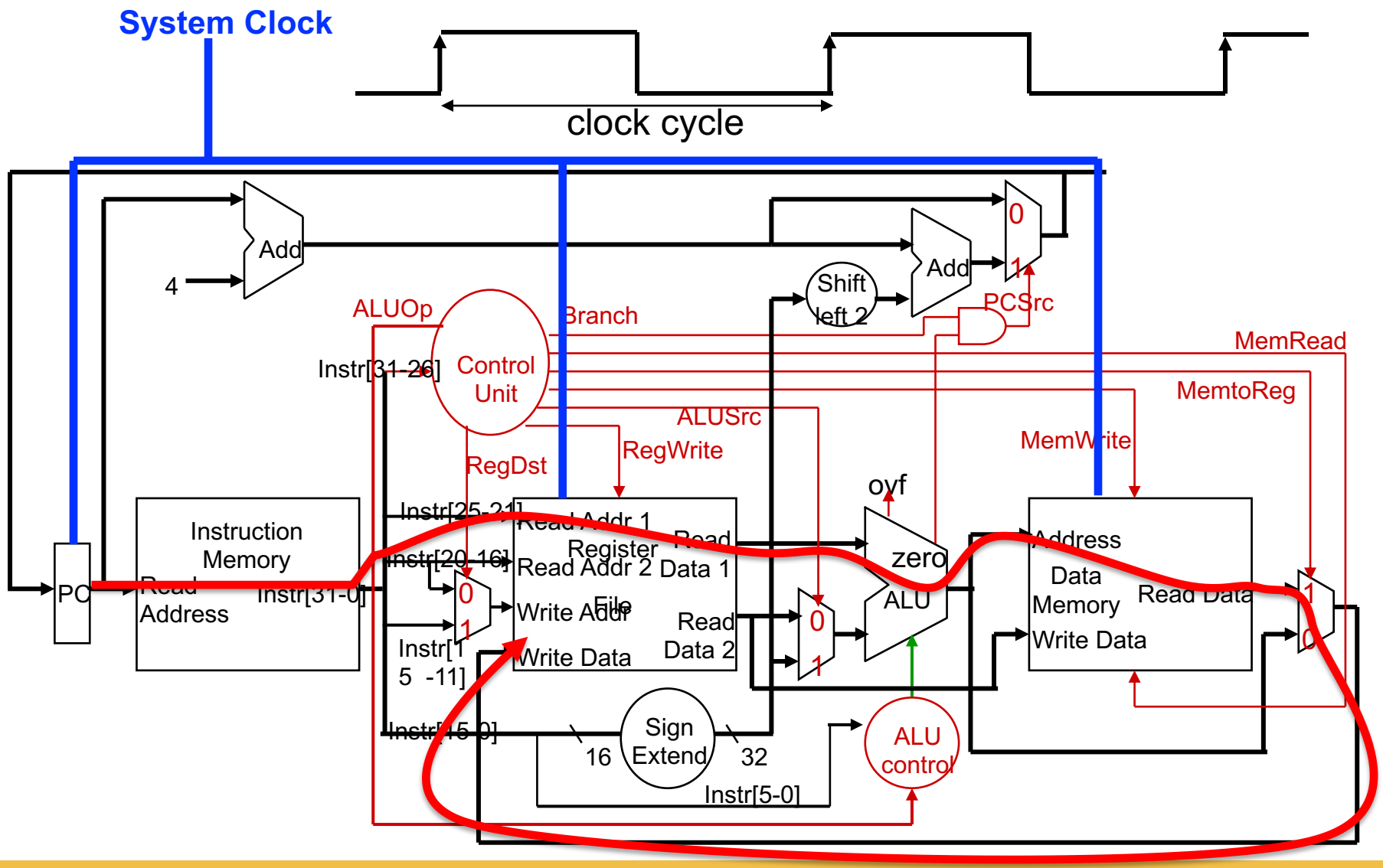
Control Hazards

Henry Duwe
Electrical and Computer Engineering
Iowa State University

Administrative

- Term Proj 2 due this week in lab
  - For keeps
- HW7
  - Due Mar 27 11:59pm
  - Last HW for exam 2 coverage
  - Question 3 best exam prep if done well
- Exam 2
  - When: April 1
  - Where: **TBD**
  - What: MIPS Arithmetic through pipelining (including hazards)

Review: Worst Case Timing (Load Instruction)



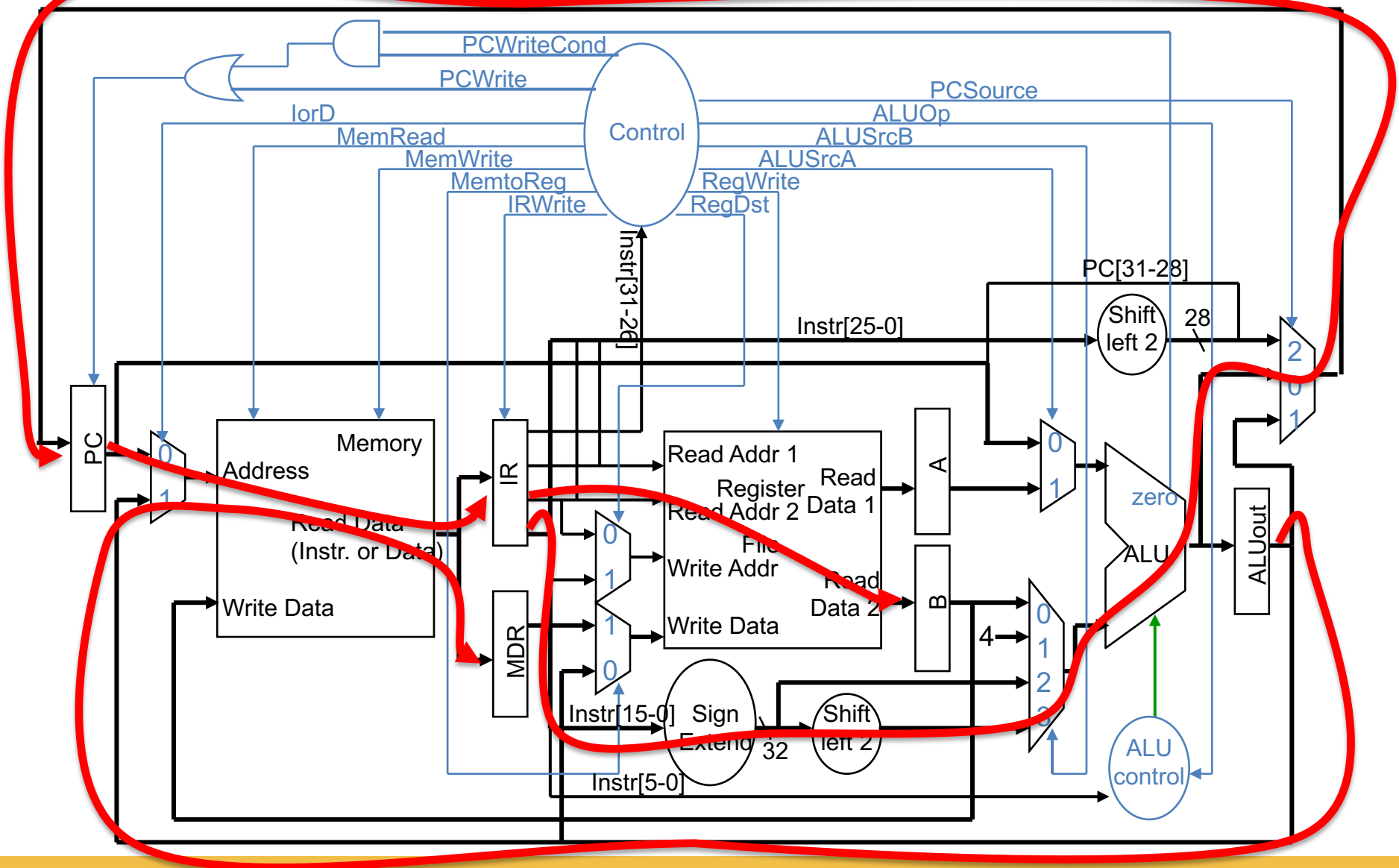
Review: Execution Time

- Drawing on the previous equation:

$$\textit{Execution Time} = \#\ \textit{Instructions} \times \frac{\textit{Cycles}}{\textit{Instruction}} \times \frac{\textit{Seconds}}{\textit{Cycle}}$$

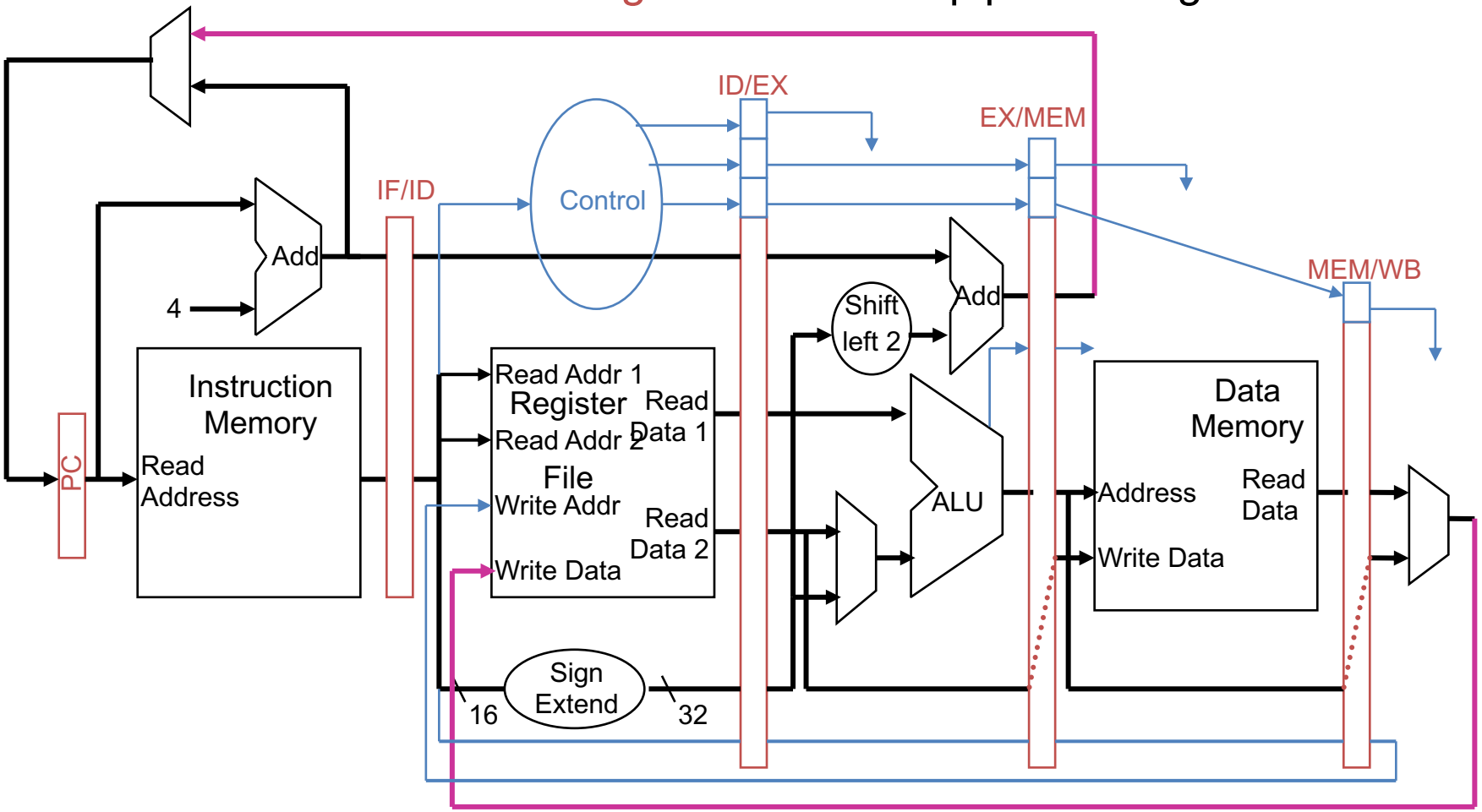
- To improve performance (i.e., reduce execution time)
  - Increase clock rate (decrease clock cycle time) OR
  - Decrease CPI OR
  - Reduce the number of instructions
- Designers balance cycle time against the number of cycles required
  - Improving one factor may make the other one worse...

Review: Multicycle Processor



Review: A Simple MIPS Pipeline

- All control signals can be determined during Decode
  - And held in the state registers between pipeline stages



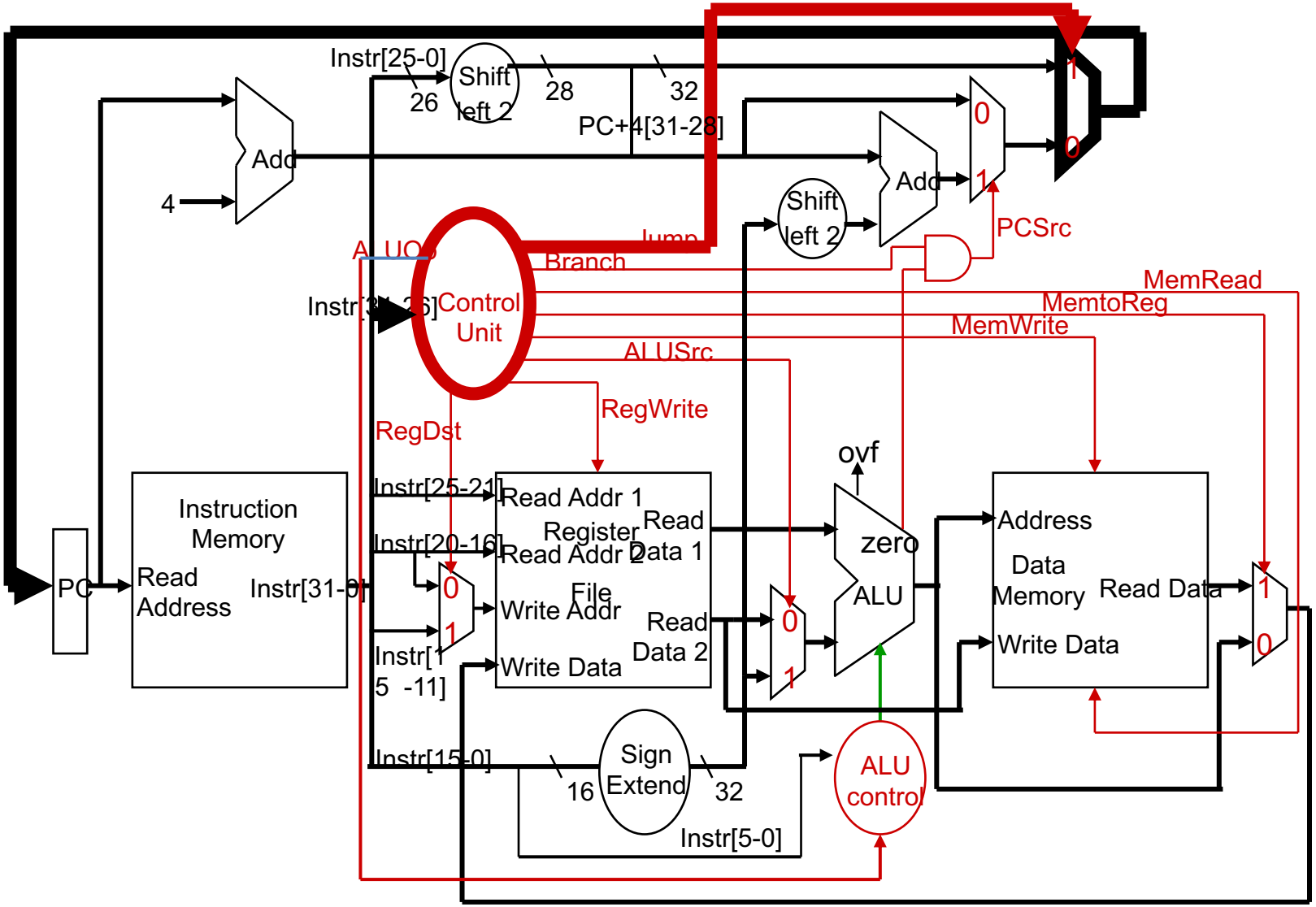
Review: Oh, the *Hazards* I've Seen

- Pipeline Hazards
  - Structural hazards: attempt to use the same resource by two different instructions at the same time
  - Data hazards: attempt to use data before it is ready
    - An instruction's source operand(s) are produced by a prior instruction still in the pipeline
  - Control hazards: attempt to make a decision about program control flow before the condition has been evaluated and the new PC target address calculated
    - Branch and jump instructions, exceptions
- Can always resolve hazards by **waiting**
  - Pipeline control must detect the hazard
  - And take action to resolve hazards

Control Hazards

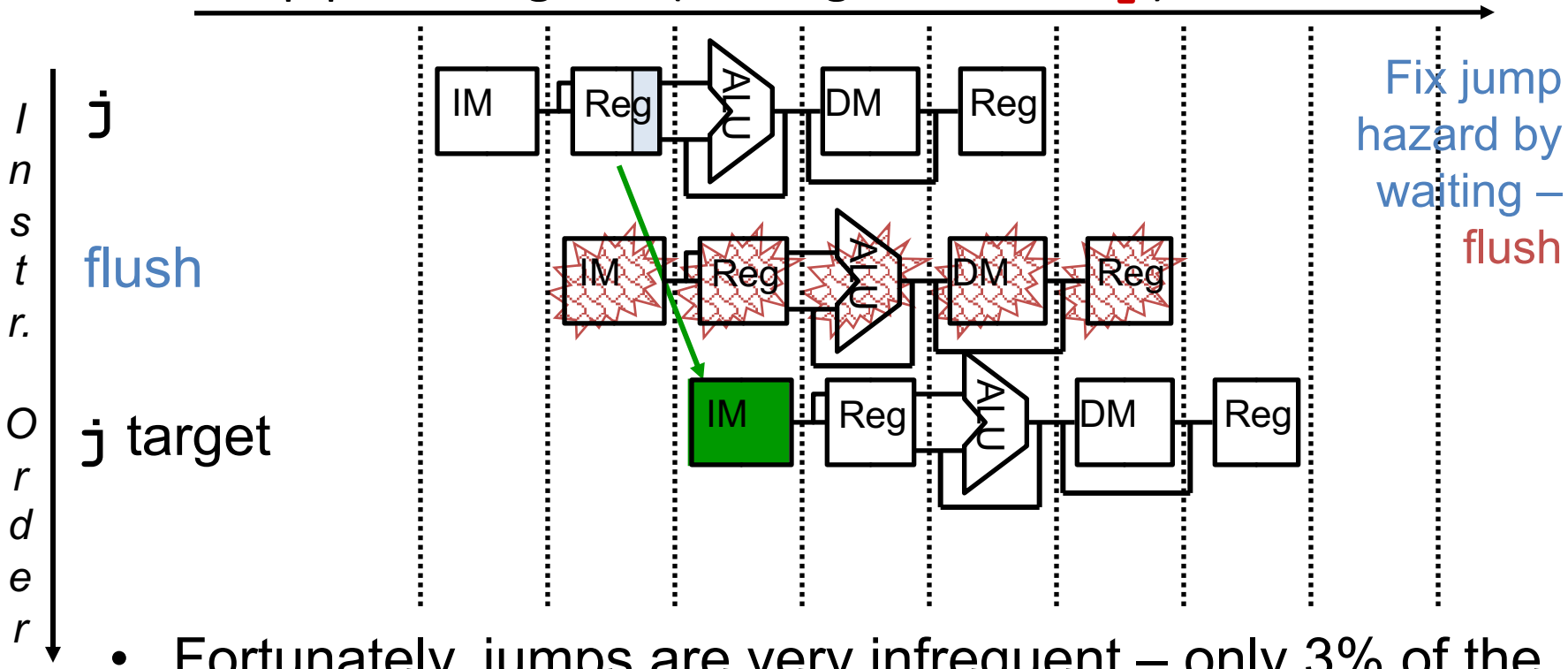
- When the flow of instruction addresses is not sequential (i.e., PC = PC + 4)
  - Conditional branches (`beq`, `bne`)
  - Unconditional branches (`j, jal, jr`)
  - Exceptions
- Possible "solutions"
  - Stall (impacts performance)
  - Move branch decision point as early in the pipeline as possible, thereby reducing the number of stall cycles
  - Delay decision (requires compiler support)
  - Predict and hope for the best!
- Control hazards occur less frequently than data hazards, but there is *nothing* as effective against control hazards as forwarding is for data hazards

Remember: Adding the Jump Instruction



Jumps Incur One Stall

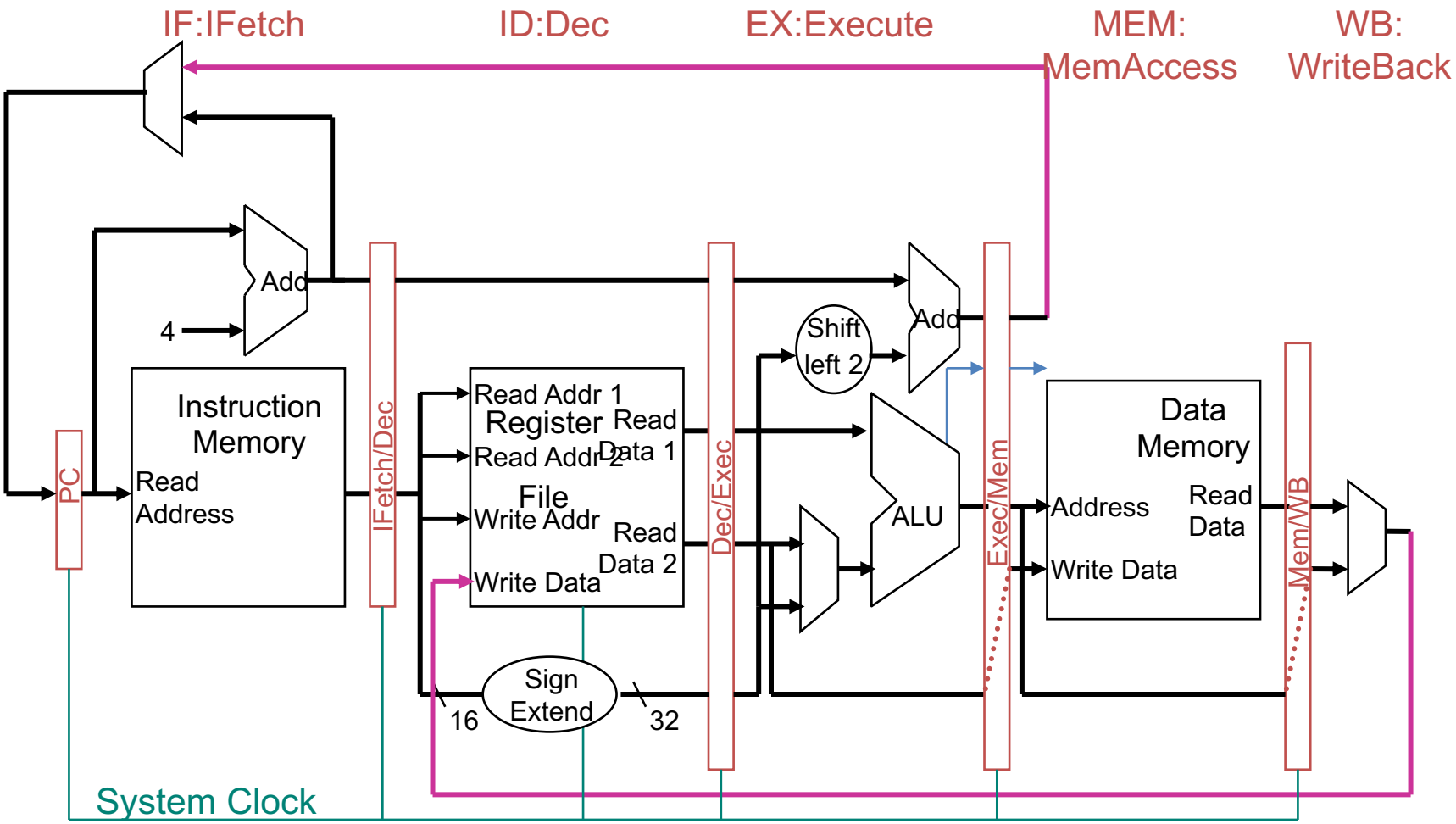
- Jumps not decoded until ID, so one flush is needed
  - To flush, set `IF.Flush` to zero the instruction field of the IF/ID pipeline register (turning it into a **`nop`**)



- Fortunately, jumps are very infrequent – only 3% of the SPECint instruction mix

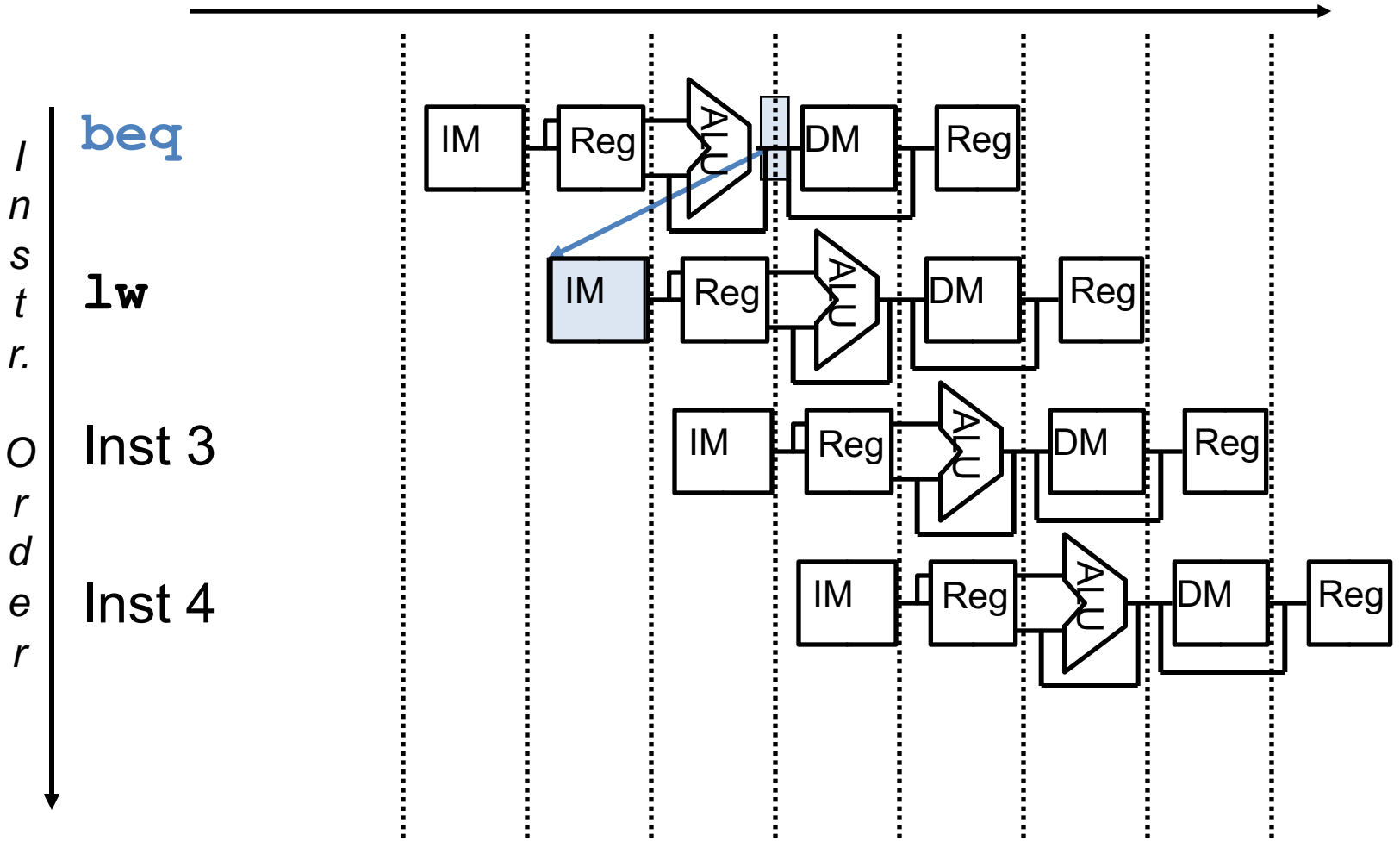
Remember: MIPS Pipeline Datapath Modifications

- What do we need to add/modify in our MIPS datapath?
  - State registers between each pipeline stage to isolate them

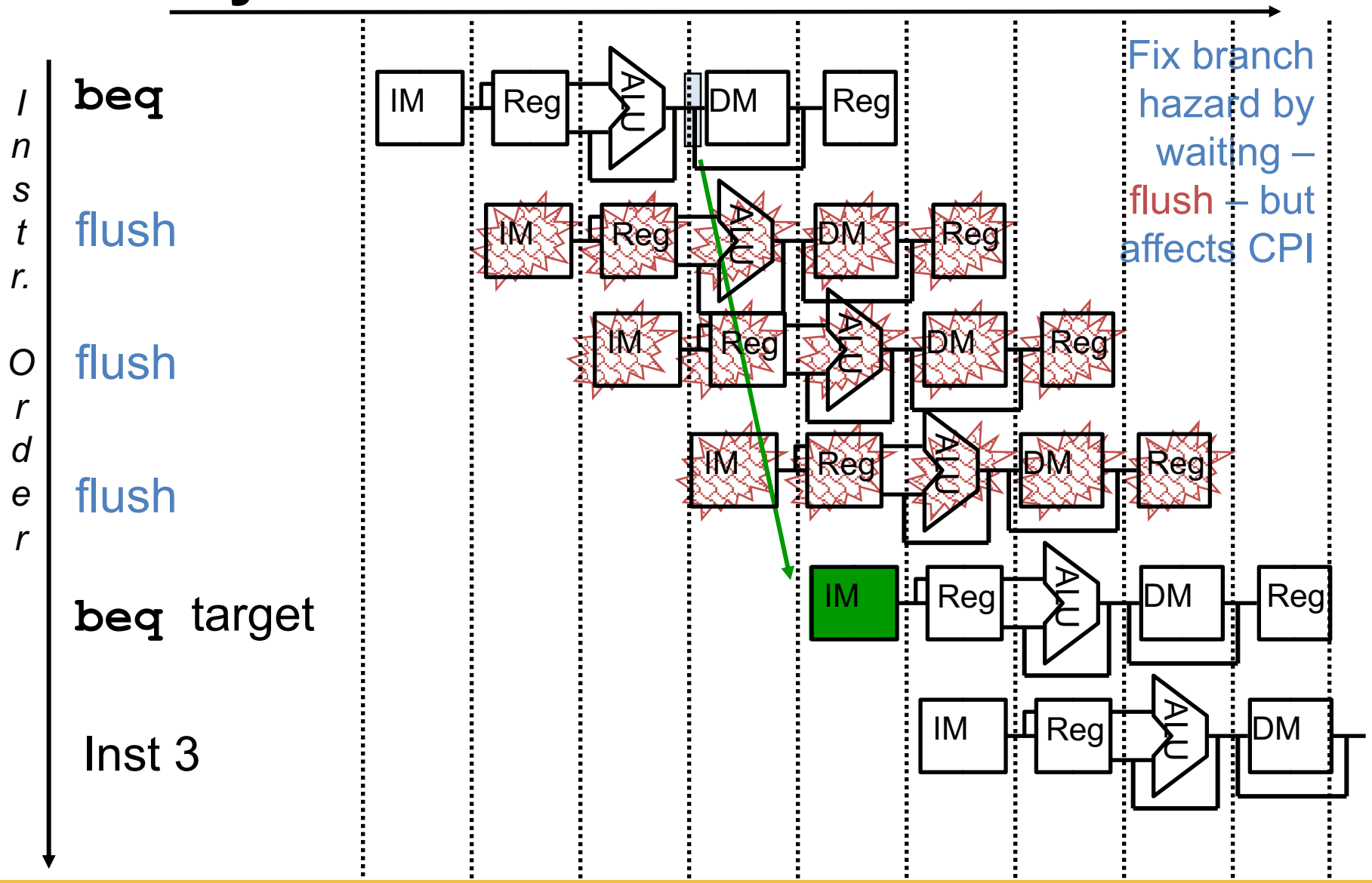


Branches Cause Control Hazards

- Dependencies backward in time cause hazards

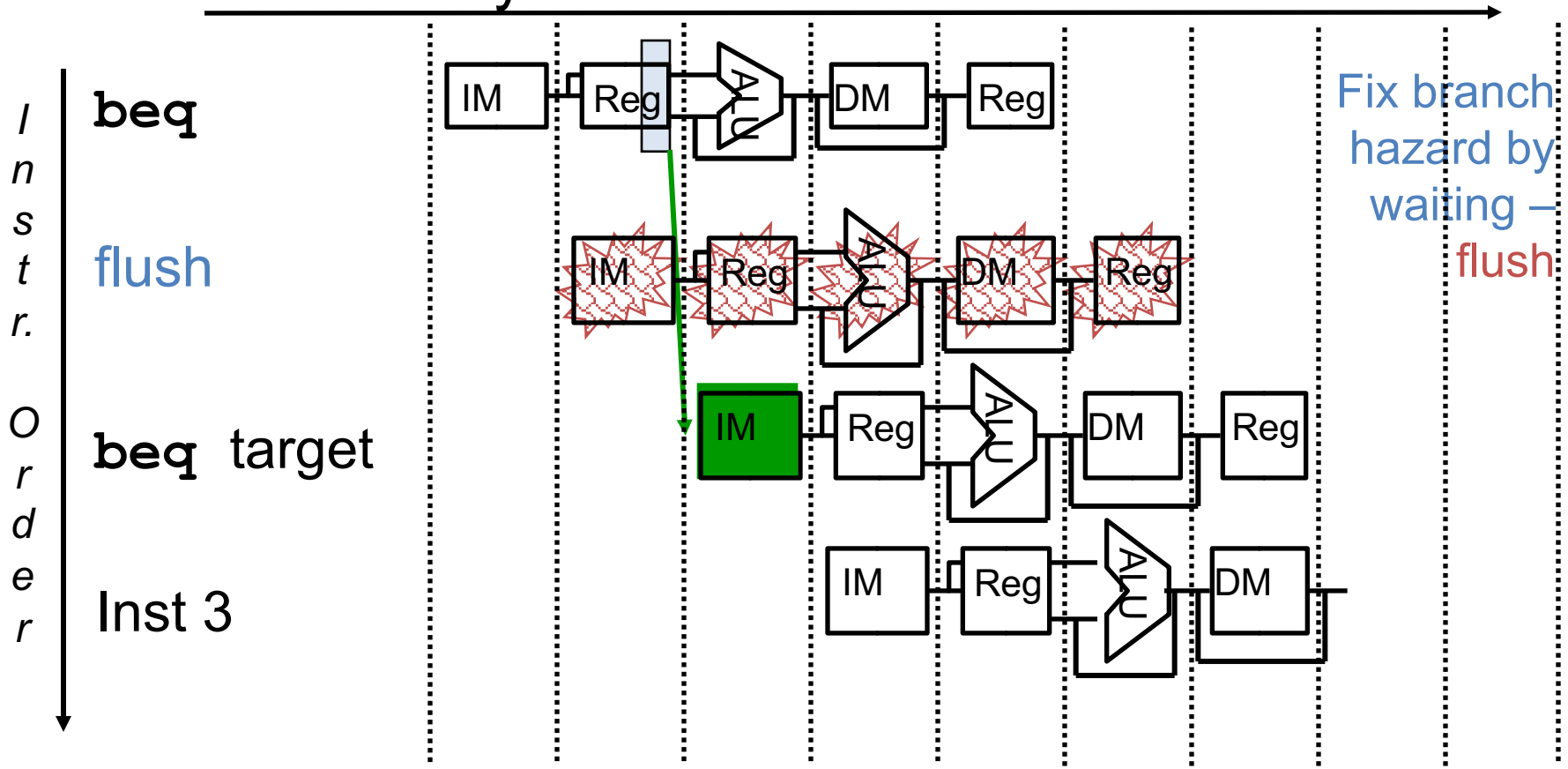


One Way to "Fix" a Branch Control Hazard

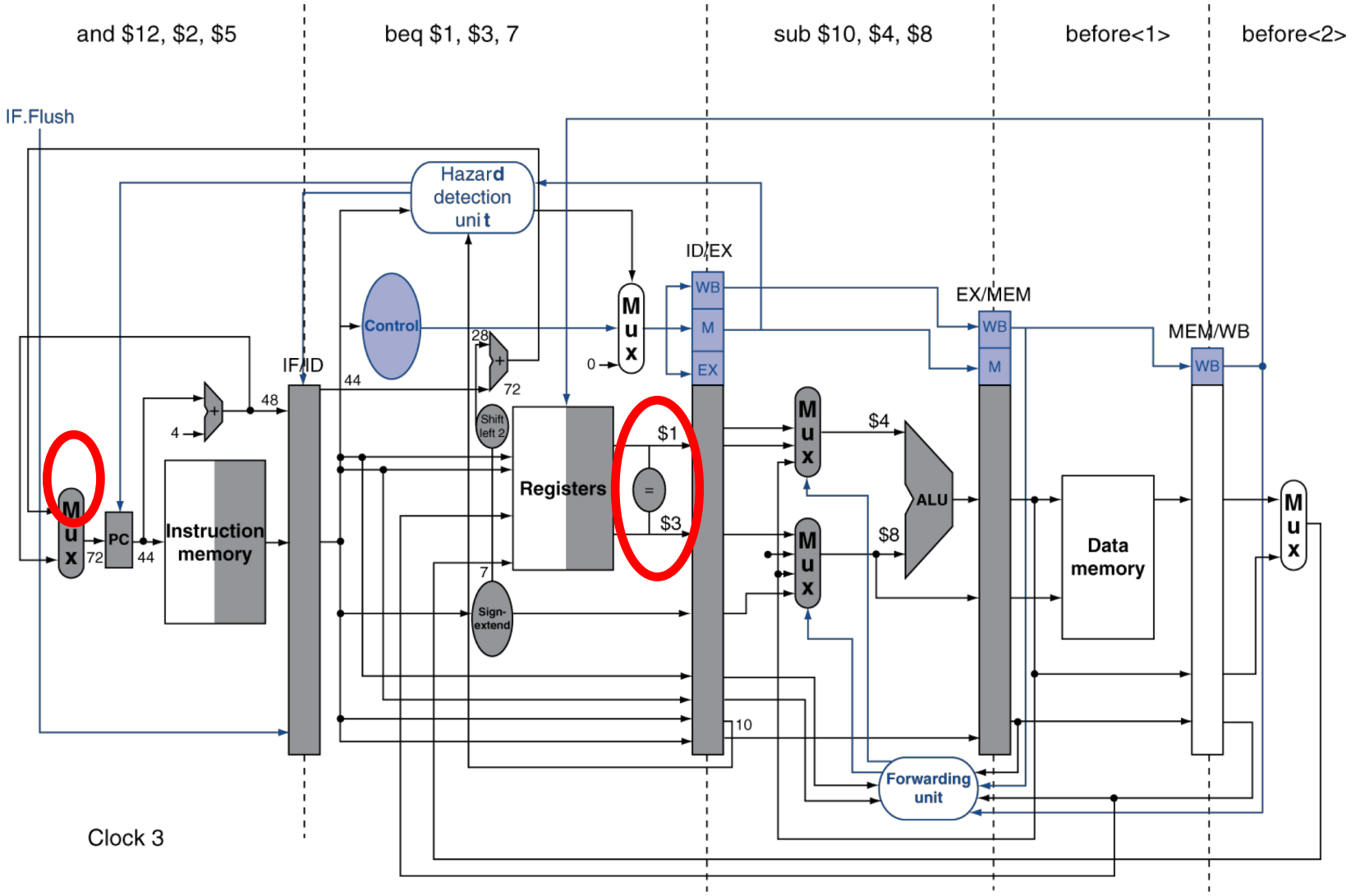


Another Method

- Move branch decision hardware back to as early in the pipeline as possible – i.e., during the decode cycle



Branch Comparison Hardware



Branch Stalls and Performance

- Need to stall for one cycle on every branch!
  - ID control point
- Consider the following case
  - The ideal CPI of the machine is 1 (applies to all instr not branches)
  - The branch causes a stall
- What is the new effective CPI after moving the branch decision if 15% of the instructions are branches?

Branch Stalls and Performance

- Need to stall for one cycle on every branch!
  - ID control point
-

In-class Assessment!

Access Code: QED

Note: sharing access code to those outside of classroom or using access while outside of classroom is considered cheating

- What is the new effective CPI after moving the branch decision if 15% of the instructions are branches?

Branch Stalls and Performance

- Need to stall for one cycle on every branch!
  - ID control point
- Consider the following case
  - The ideal CPI of the machine is 1 (applies to all instr not branches)
  - The branch causes a stall
- What is the new effective CPI after moving the branch decision if 15% of the instructions are branches?
- The new effective CPI is $1 + 1 \times 0.15 = 1.15$
- The old effective CPI was $1 + 3 \times 0.15 = 1.45$

Yet Another Solution: Delayed Branches

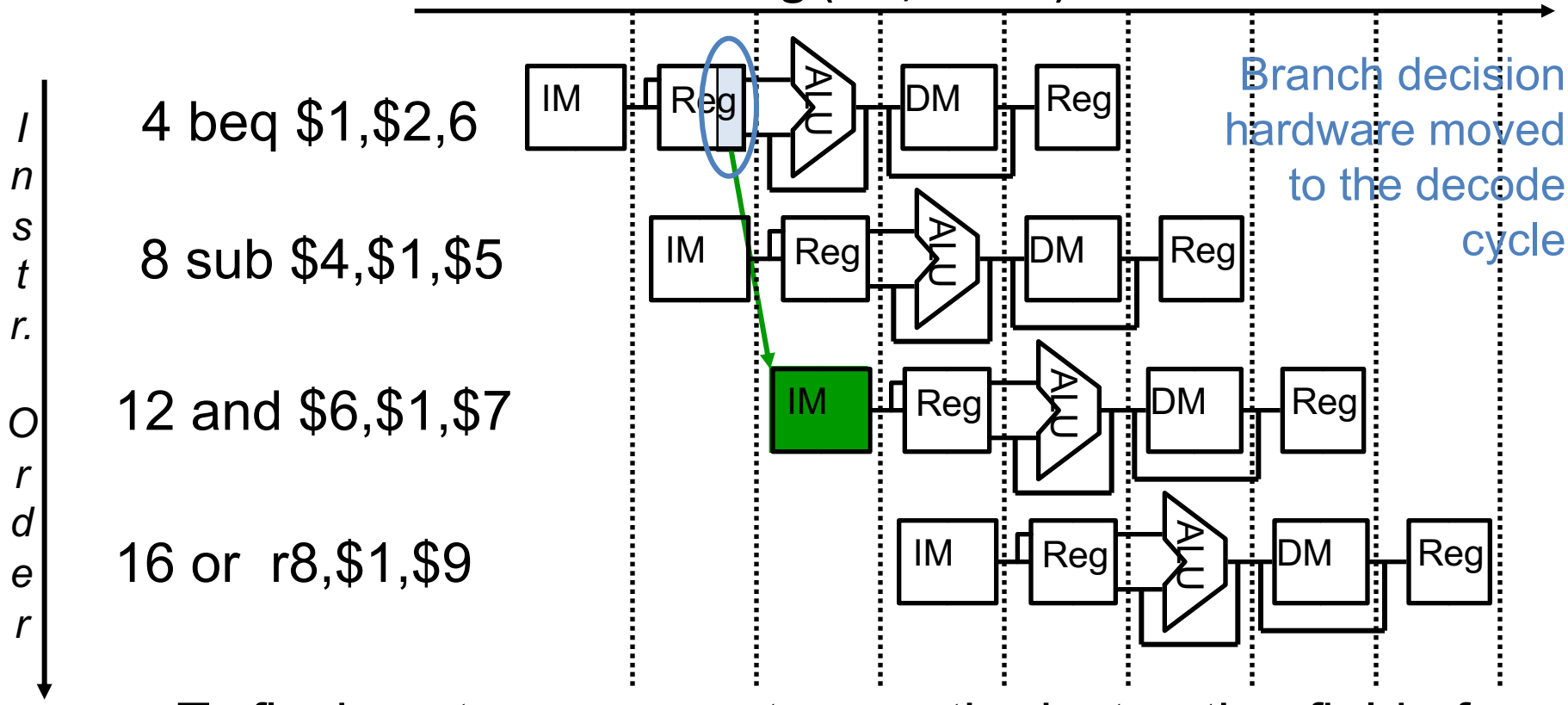
- Solution used in early MIPS machines
  - Had a branch delay of one
  - Branch does not take effect until the cycle after its execution
- Example:

```
beq r1, r2, L     # Branch instruction
sub r4, r1, r3    # instr ALWAYS is executed
and r6, r2, r7    # instr executes if not taken
```

- Worked well initially, but now is a pain
  - Compiler can fill one slot 50% of the time
  - Machines have many branch delay slots
  - Issue more than one instruction per cycle

Still More Fixes

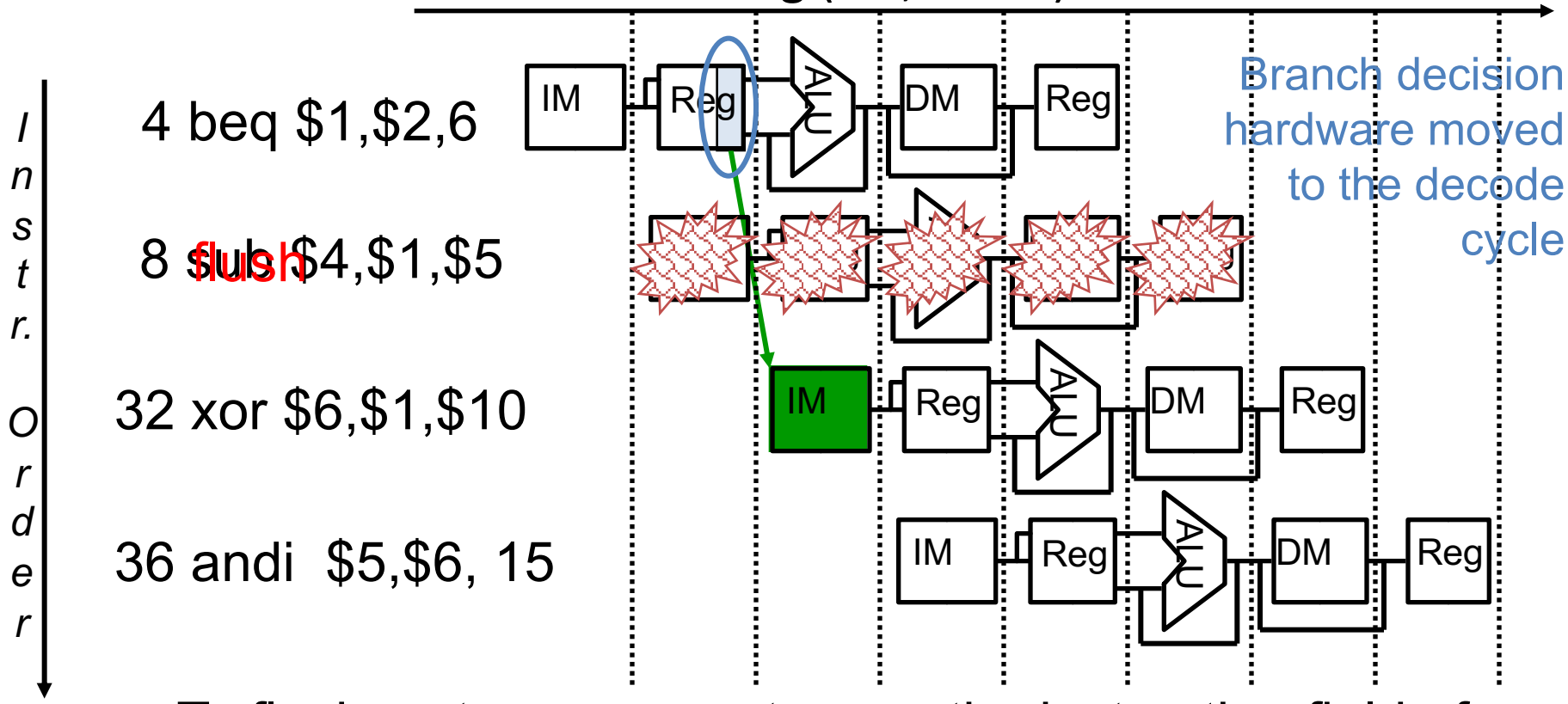
- Predict branches are always not taken – and take corrective action when wrong (i.e., taken)



- To flush, set **`IF.Flush`** to zero the instruction field of the IF/ID pipeline register (turning it into a **`nop`**)

Still More Fixes

- Predict branches are always not taken – and take corrective action when wrong (i.e., taken)



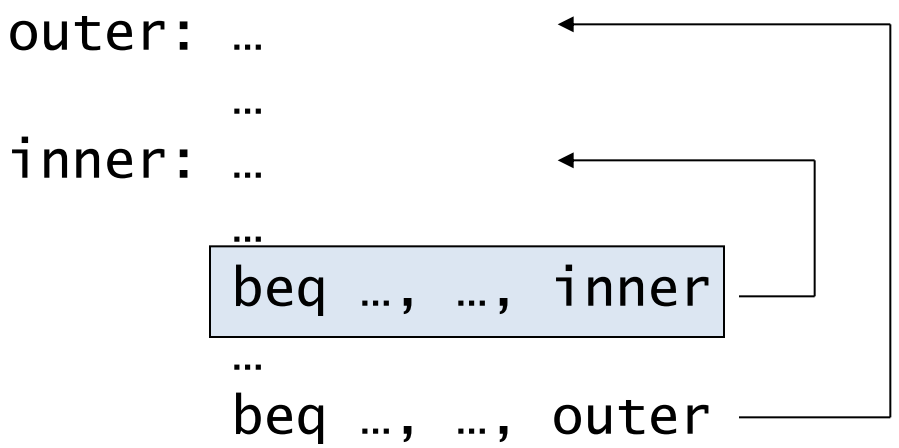
- To flush, set **`IF.Flush`** to zero the instruction field of the IF/ID pipeline register (turning it into a **`nop`**)

Dynamic Branch Prediction

- In deeper and superscalar pipelines, branch penalty is more significant
- Use dynamic prediction
  - Branch prediction buffer (aka branch history table)
  - Indexed by recent branch instruction addresses
  - Stores outcome (taken/not taken)
  - To execute a branch
    - Check table, expect the same outcome
    - Start fetching from fall-through or target
    - If wrong, flush pipeline and flip prediction

1-Bit Predictor: Shortcoming

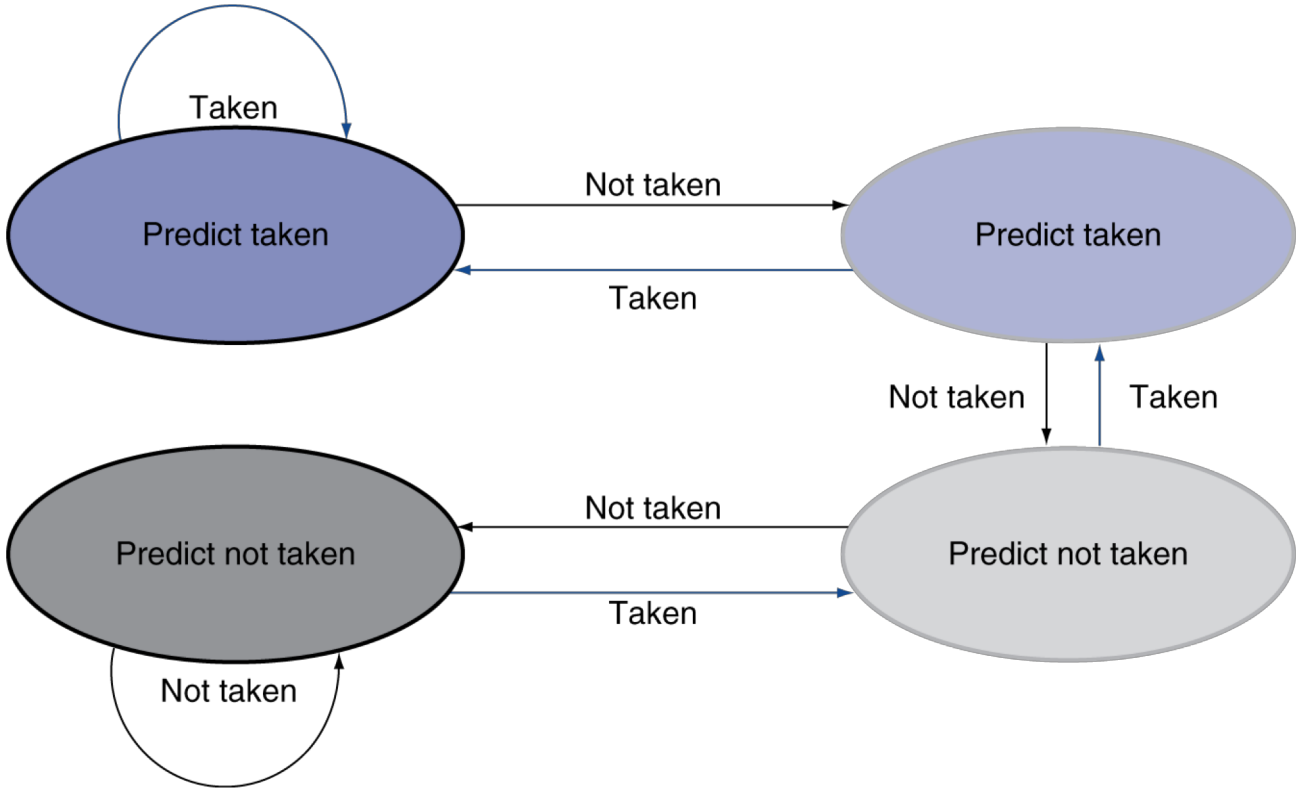
- Inner loop branches mispredicted twice!



- Mispredict as taken on last iteration of inner loop
- Then mispredict as not taken on first iteration of inner loop next time around

2-Bit Predictor

- Only change prediction on two successive mispredictions



Preview: Calculating the Branch Target

- Even with predictor, still need to calculate the target address
  - 1-cycle penalty for a taken branch

Acknowledgments

- These slides contain material developed and copyright by:
  - Joe Zambreno (Iowa State)
  - David Patterson (UC Berkeley)
  - Mary Jane Irwin (Penn State)
  - Christos Kozyrakis (Stanford)
  - Onur Mutlu (Carnegie Mellon)
  - Krste Asanović (UC Berkeley)