# CS305
# Computer Architecture

## Control Hazards in the Pipeline

Bhaskaran Raman
Room 406, KR Building
Department of CSE, IIT Bombay

http://www.cse.iitb.ac.in/~br

# Control Hazards

- Control hazard: pipeline cannot operate normally due to ((possibility of) non-sequential) control flow

| | CC1 | CC2 | CC3 | CC4 | CC5 | CC6 | CC7 | CC8 |
|---|---|---|---|---|---|---|---|---|
| `ADD R1, R2, R3` | IF | ID | EX | MEM | WB | | | |
| `BEQ R4, R5, LBL` | | IF | ID | EX | MEM | WB | | |
| `SUB R6, R7, R8` | | | IF | ID | EX | MEM | WB | |
| `LW R6, 0(R7)` | | | | IF | ID | EX | MEM | WB |

**Execution of SUB and LW should depend on branch condition: known only in CC4**

# Stalling “Solution” for Control Hazard

| | | CC1 | CC2 | CC3 | CC4 | CC5 | CC6 | CC7 | CC8 | |
|---|---|---|---|---|---|---|---|---|---|---|
| ADD | R1, R2, R3 | IF | ID | EX | MEM | WB | | | | |
| BEQ | R4, R5, LBL | | IF | ID | EX | MEM | WB | | | |
| SUB | R6, R7, R8 | | | IF | IF | IF | ID | EX | MEM | WB |
| LW | R6, 0(R7) | | | | | | IF | ID | EX | MEM |

Q: Suppose 10% instructions are branches, what is the performance implication?

A: CPI will increase from 1 to 1.2 due to control stalls: EXPENSIVE

## Techniques to Reduce Branch Penalty

- 2-stage branch completion
- Assume branch not taken
- Branch prediction
- Delayed branches
- Many advanced techniques (not in this course):
  - Correlating predictors
  - Branch target buffer
  - Special instructions for branch delay slot

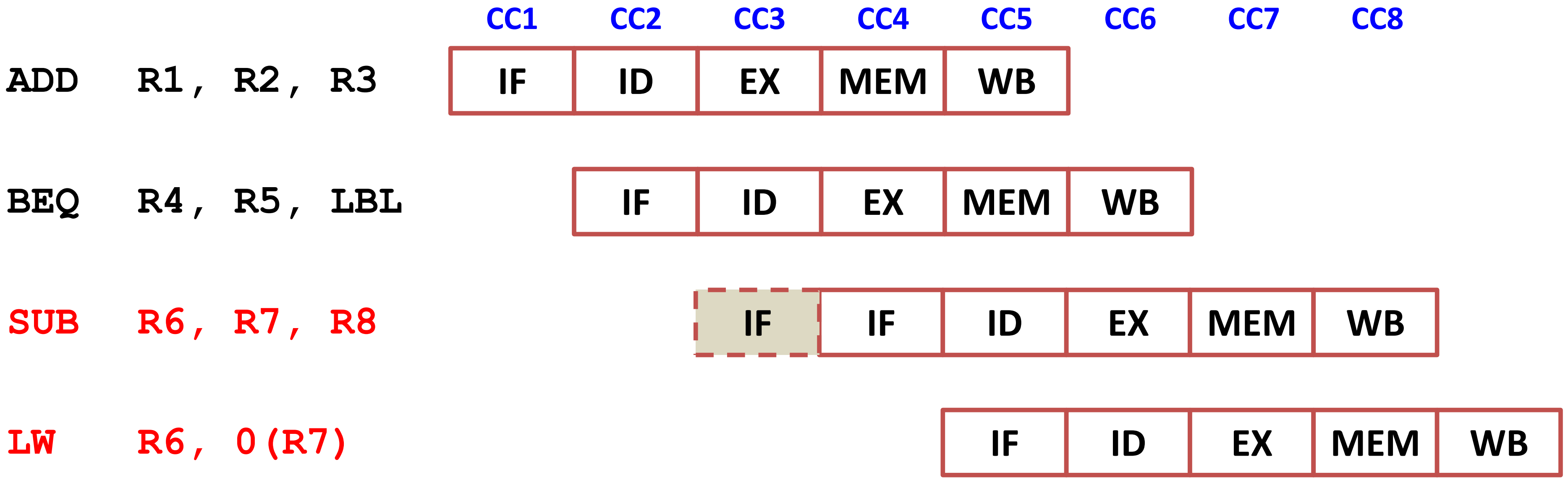
## Techniques to Reduce Branch Penalty

- 2-stage branch completion
- Assume branch not taken
- Branch prediction
- Delayed branches
- Many advanced techniques (not in this course):
  - Correlating predictors
  - Branch target buffer
  - Special instructions for branch delay slot

## Reducing Stalls: 2-Stage Branch Completion

- Extra hardware required:
  - Comparator in stage-2
  - Needs to complete in half a cycle!
- Data hazard implications:
  - Extra forwarding required: forwarding to ID stage, for branch instruction
  - **`beq`** itself may stall due to dependence on earlier instruction!

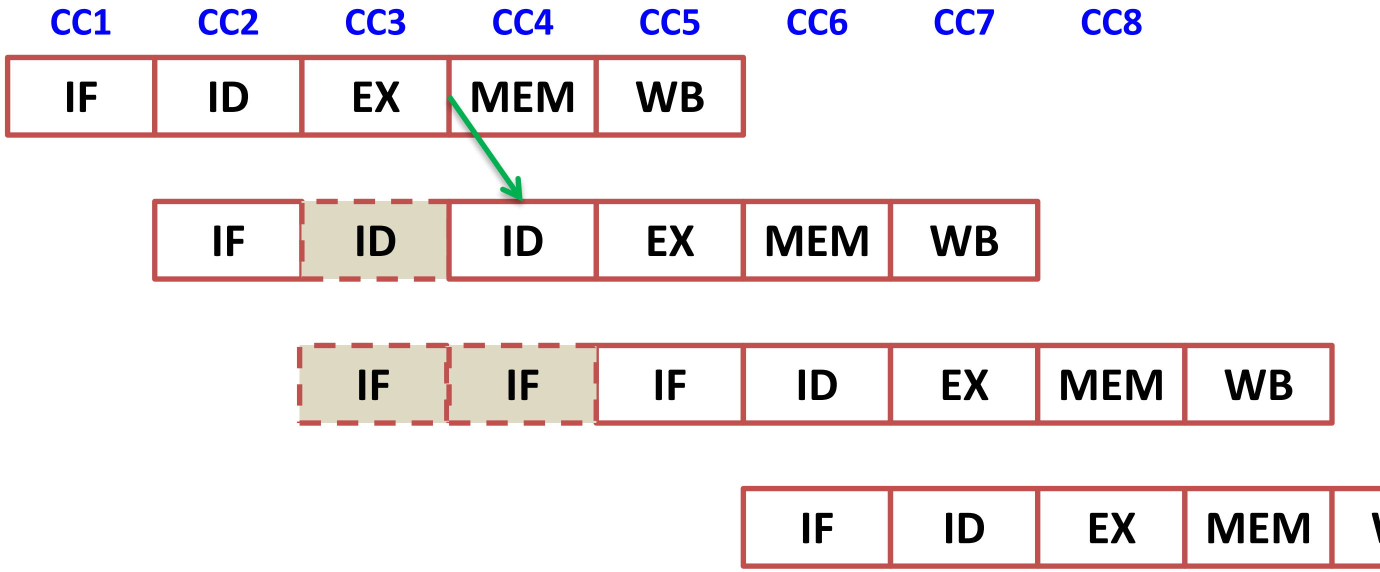
# Implication-1 of 2-Stage beq completion



Q: Suppose 10% instructions are branches, what is the performance implication?

A: CPI will increase from 1 to 1.1 due to control stalls: still not so good

# Implications-2,3 of 2-Stage beq completion



| | CC1 | CC2 | CC3 | CC4 | CC5 | CC6 | CC7 | CC8 | |
|---|---|---|---|---|---|---|---|---|---|
| ADD R1, R2, R3 | IF | ID | EX | MEM | WB | | | | |
| BEQ R1, R7, LBL | | IF | ID | ID | EX | MEM | WB | | |
| SUB R6, R7, R8 | | | IF | IF | IF | ID | EX | MEM | WB |
| LW R4, 0(R5) | | | | | IF | ID | EX | MEM | |

## Techniques to Reduce Branch Penalty

- 2-stage branch completion
- Assume branch not taken
- Branch prediction
- Delayed branches
- Many advanced techniques (not in this course):
  - Correlating predictors
  - Branch target buffer
  - Special instructions for branch delay slot

# Reducing Stalls: Assume Branch Not Taken

| | CC1 | CC2 | CC3 | CC4 | CC5 | CC6 | CC7 | CC8 |
|---|---|---|---|---|---|---|---|---|
| `ADD R1, R2, R3` | IF | ID | EX | MEM | WB | | | |
| `BEQ R4, R5, LBL` | | IF | ID | EX | MEM | WB | | |
| `SUB R6, R7, R8` | | | IF | ID | EX | MEM | WB | |
| `LW R6, 0(R7)` | | | | IF | ID | EX | MEM | WB |

In CC4, cancel out the instruction in ID stage, if branch taken

Does not help much for loops: branch taken in most cases

Price paid in this approach: increased control complexity

## Techniques to Reduce Branch Penalty

- 2-stage branch completion
- Assume branch not taken
- Branch prediction
- Delayed branches
- Many advanced techniques (not in this course):
  - Correlating predictors
  - Branch target buffer
  - Special instructions for branch delay slot

## Reducing Stalls: Branch Prediction

- Idea: remember whether branch was taken last time

| Last few bits of PC | Prediction (taken=1, not taken=0) |
|---|---|
| … | 1 |
| … | 0 |

  - Only last few bits of PC needed: need to deal with branch mis-prediction anyway
- Single-bit predictor: loops are mis-predicted twice
- 2-bit predictor: predict based on last two bits

## Techniques to Reduce Branch Penalty

- 2-stage branch completion
- Assume branch not taken
- Branch prediction
- Delayed branches
- Many advanced techniques (not in this course):
  - Correlating predictors
  - Branch target buffer
  - Special instructions for branch delay slot

# Reducing Branch Penalty: Delayed Branch, Branch Delay Slot

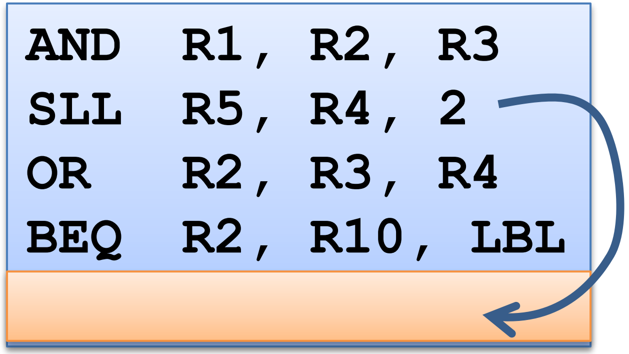
- Change semantic of branch in the ISA
  - Instruction after branch WILL BE executed, even if branch is taken
  - Such an instruction is said to be in the branch delay slot

| | 1 | 2 | 3 | 4 | 5 | | |
|---|---|---|---|---|---|---|---|
| `ADD R1, R2, R3` | IF | ID | EX | MEM | WB | | |
| `BEQ R4, R5, LBL` | | IF | ID | EX | MEM | WB | |
| `SUB R6, R7, R8` | | | IF | ID | EX | MEM | WB |

**Q: Who will fill the branch delay slot?**

**A: Compiler has to do it: yet another important role for the compiler**

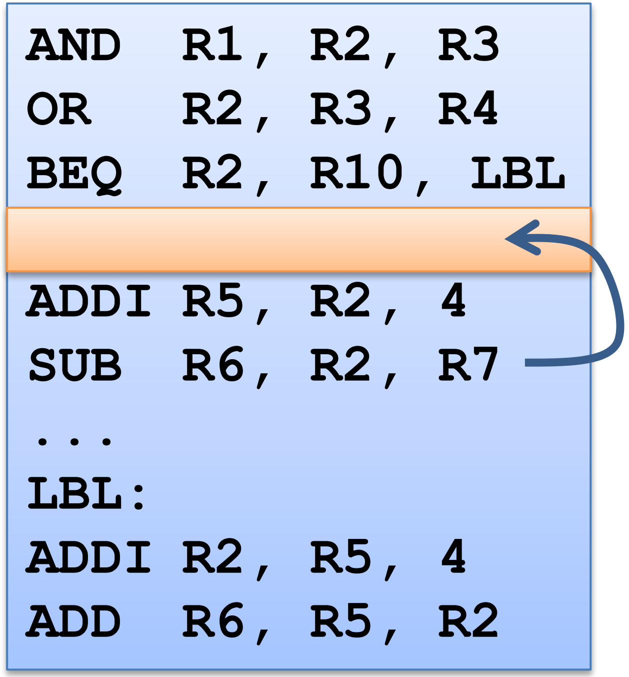
# Filling the Branch Delay Slot



```
AND  R1, R2, R3
SLL  R5, R4, 2
OR   R2, R3, R4
BEQ  R2, R10, LBL

```

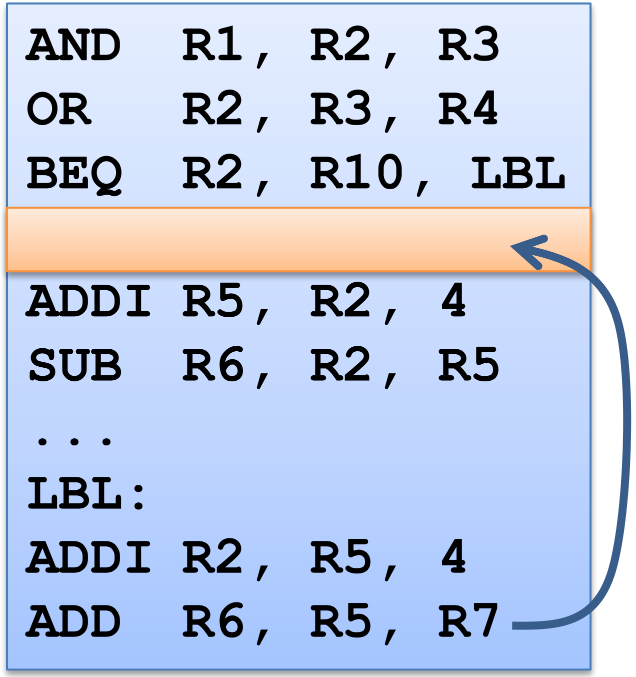
From before branch



```
AND  R1, R2, R3
OR   R2, R3, R4
BEQ  R2, R10, LBL

ADDI R5, R2, 4
SUB  R6, R2, R7
...
LBL:
ADDI R2, R5, 4
ADD  R6, R5, R2
```

From branch fall through



```
AND  R1, R2, R3
OR   R2, R3, R4
BEQ  R2, R10, LBL

ADDI R5, R2, 4
SUB  R6, R2, R5
...
LBL:
ADDI R2, R5, 4
ADD  R6, R5, R7
```

From branch target

- Desirable to fill delay slot from before branch: why? A: always executed
- It may not always be possible to fill delay slot: fill nop

# Techniques to Reduce Branch Penalty

- 2-stage branch completion
- Assume branch not taken
- Branch prediction
- Delayed branches
- Many advanced techniques (not in this course):
  - Correlating predictors
  - Branch target buffer
  - Special instructions for branch delay slot

Q: Which of these techniques are complementary? That is, can be used together with one another?

## Summary

- Control hazard: non-sequential control flow in program causes disruption in normal pipeline
- Stalling is expensive
- A variety of solutions to reduce branch penalty
  - 2-stage branch
  - Branch prediction
  - Delay slot: filled with compiler’s help
  - Many, many advanced techniques!