Pipelined Architecture

Virendra Singh
Computer Architecture and Dependable Systems Lab
Department of Electrical Engineering
Indian Institute of Technology Bombay
http://www.ee.iitb.ac.in/~viren/
E-mail: viren@ee.iitb.ac.in

CS-683: Advanced Computer Architecture

Lecture 8 (16 Aug 2021)

CADSL

Ways to Handle Branch

- Stall or bubble
- Delayed branch
- Branch prediction:
  - Heuristics
    - Next instruction
    - Prediction based on statistics (dynamic)
    - Hardware decision (dynamic)
  - Prediction error: pipeline flush

Branch Prediction

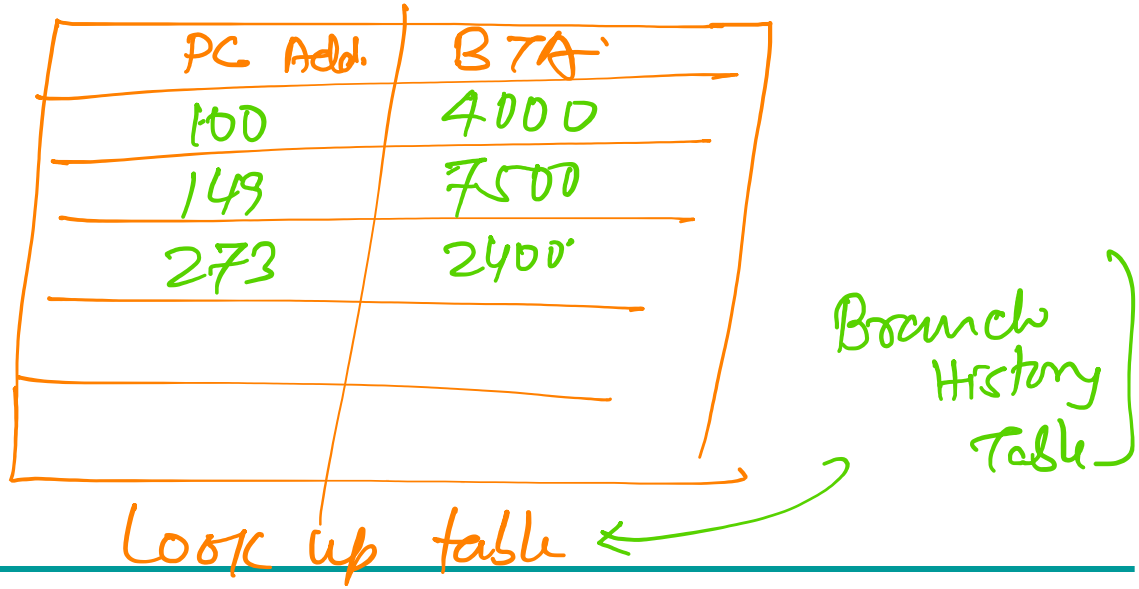
Statically taken

taken 70%. not taken 30%.

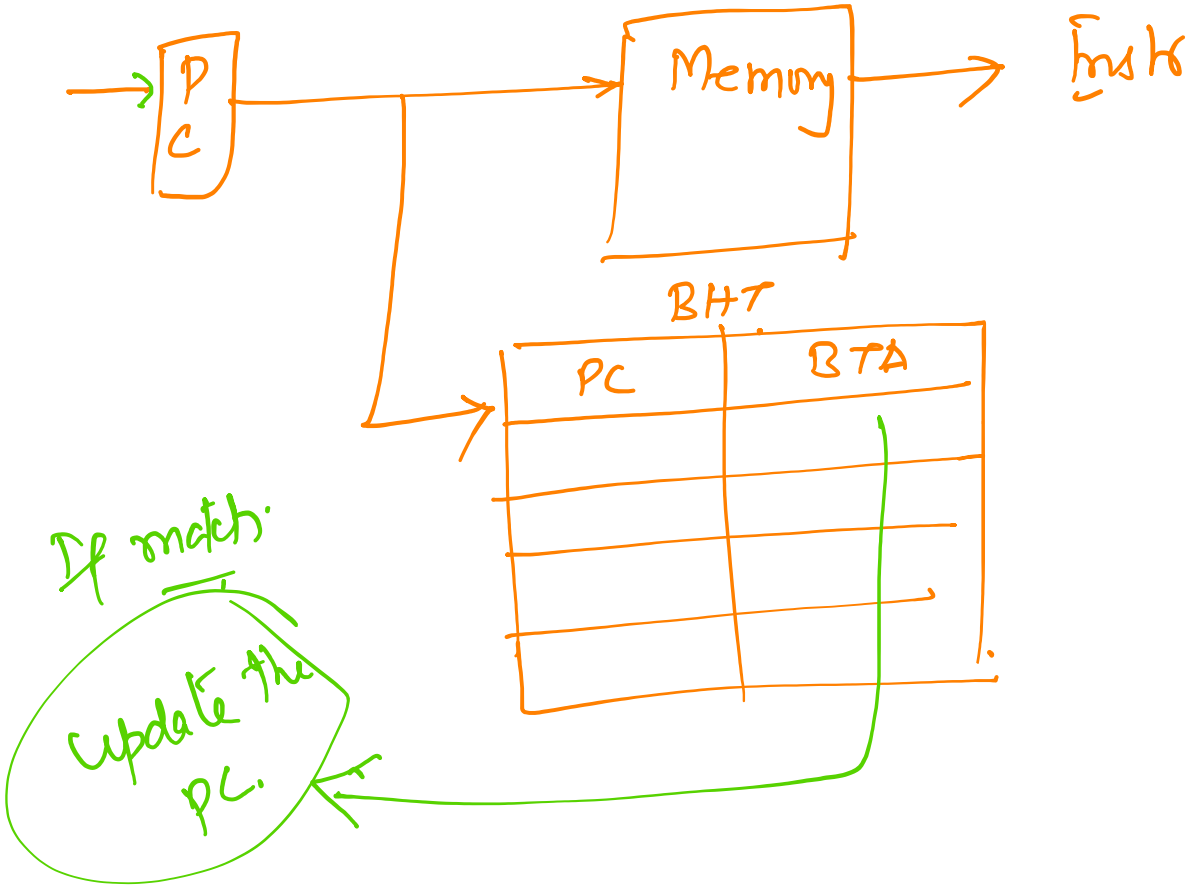
① Instruction is branch instruction] Fetch.

② Target address

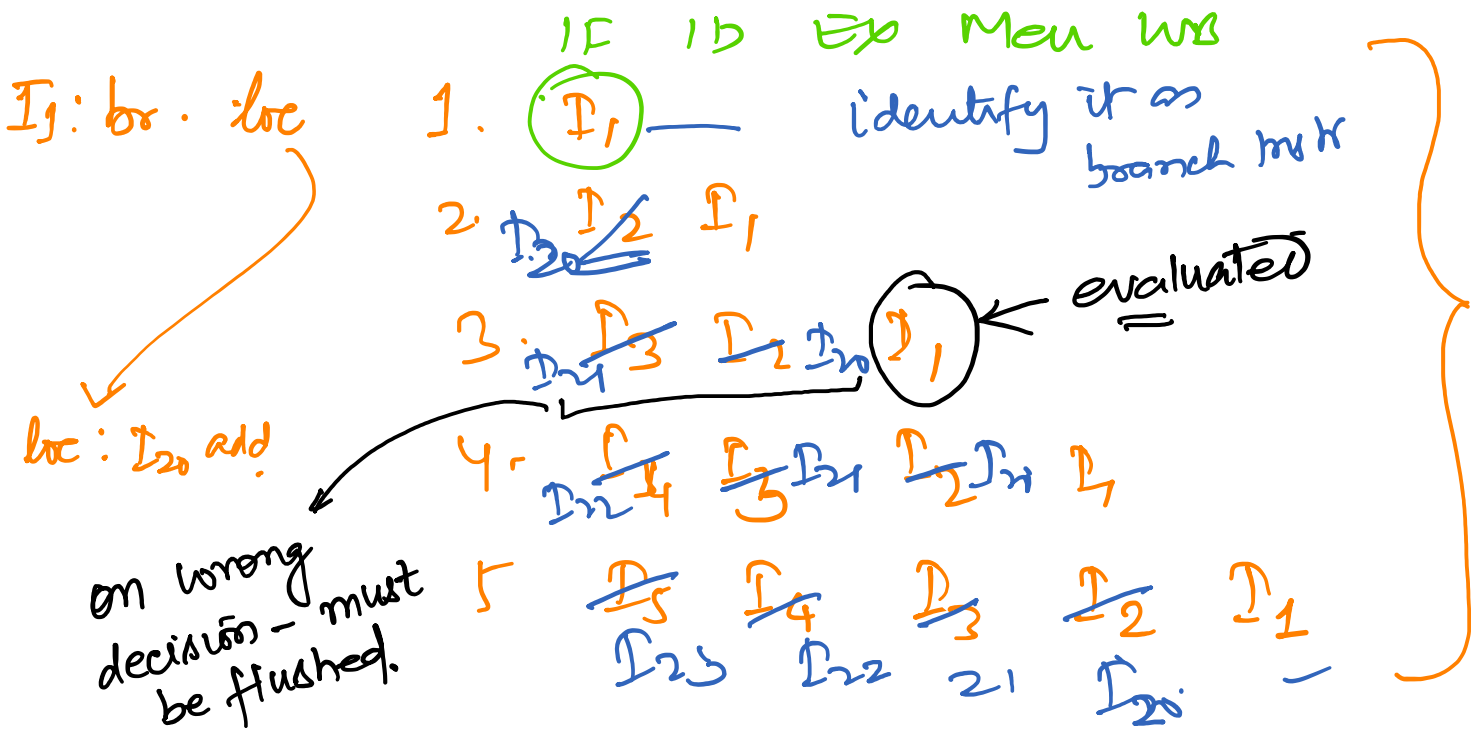
Record.



Branch Prediction



Branch Prediction



If decision is correct → NO PENALTY (70%)

20%

$CPI = 1 + .2 \times .3 \times 2 = 1.12$ ✓

How to improve.

Dynamic decision

– use the behaviour of last time

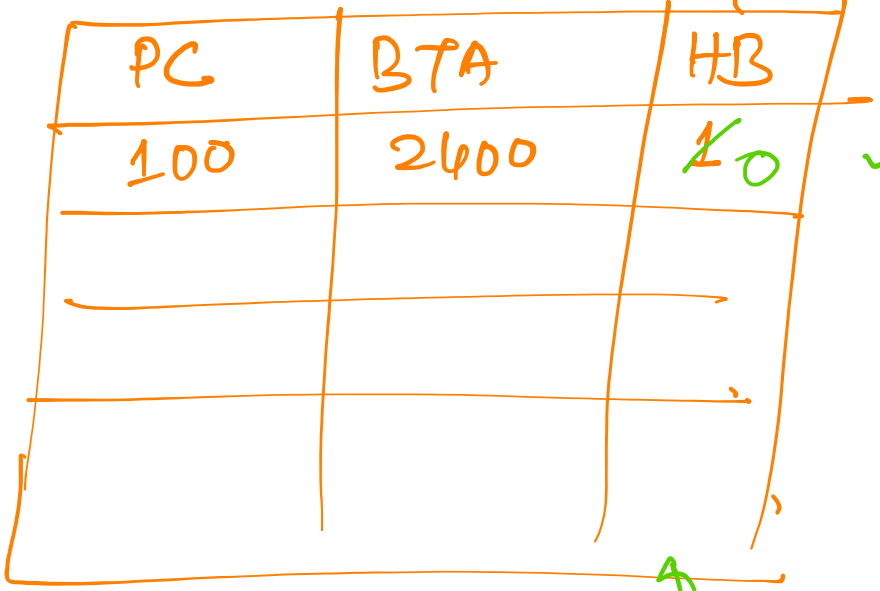
↑

Record the behaviour.

(in look up table – BHT)

if (i<j)

BHT



History bit

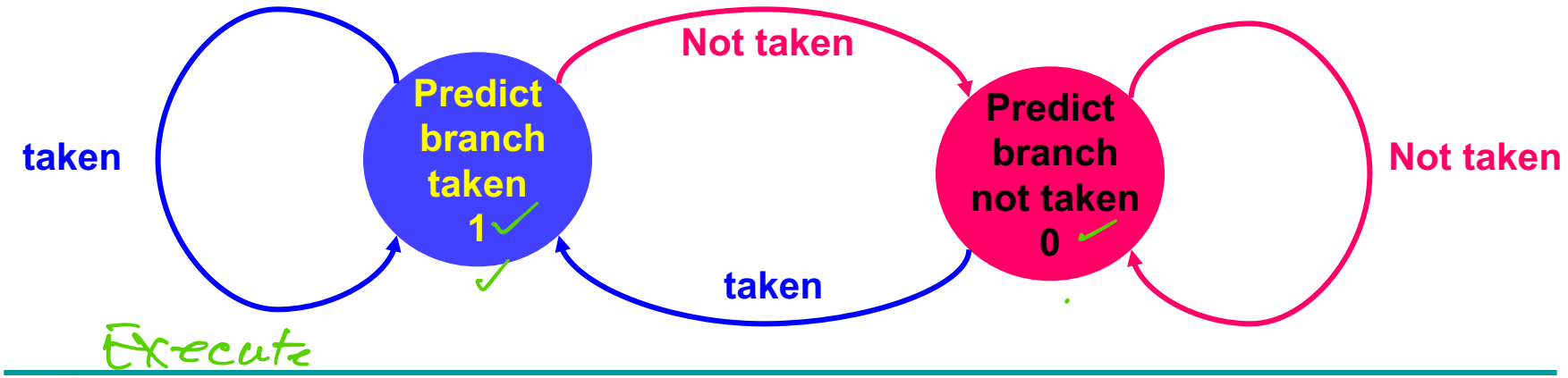
can capture the data dependent behaviour.

> 80 %

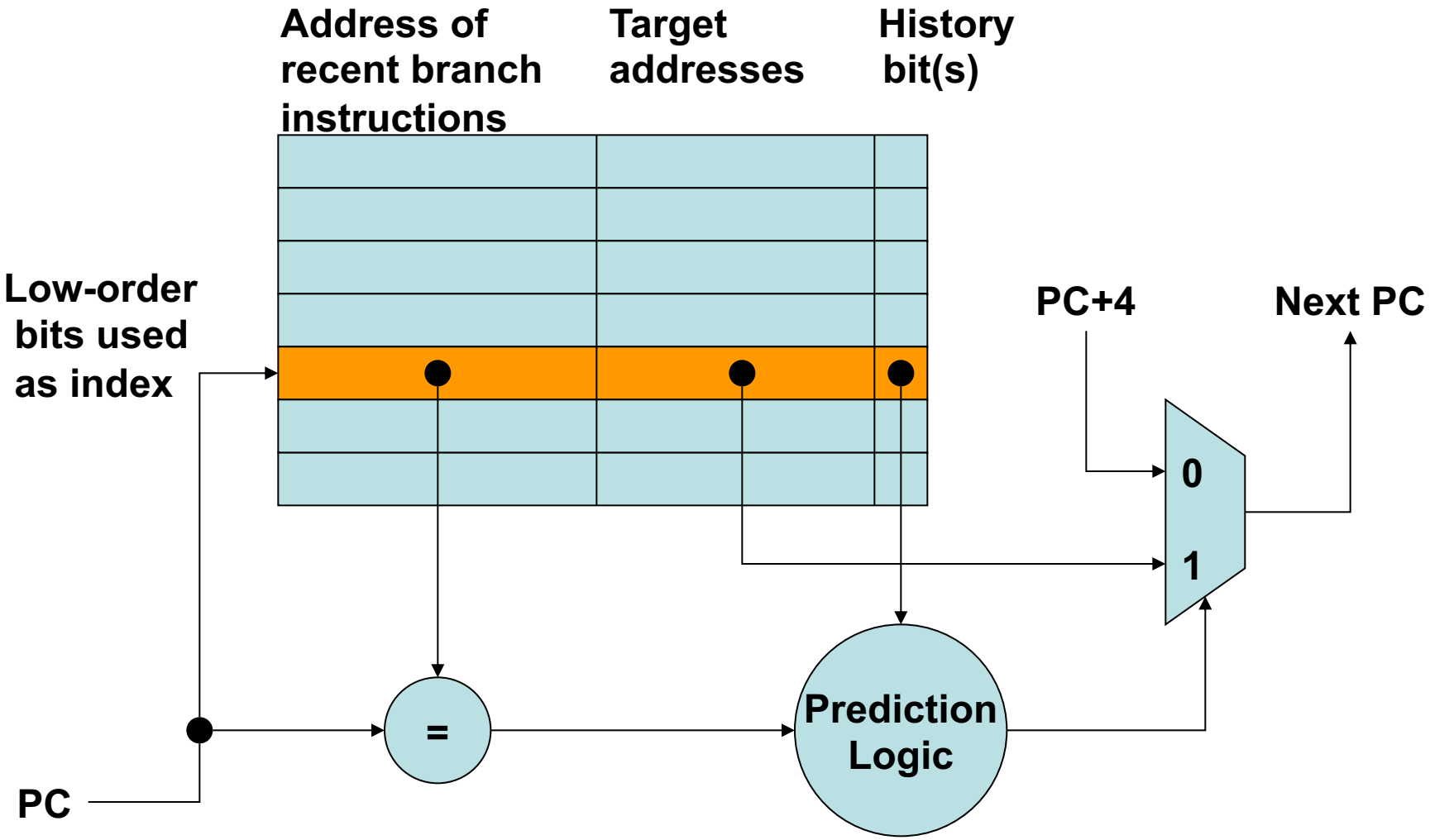
read at fetch & modify at execute

Branch Prediction

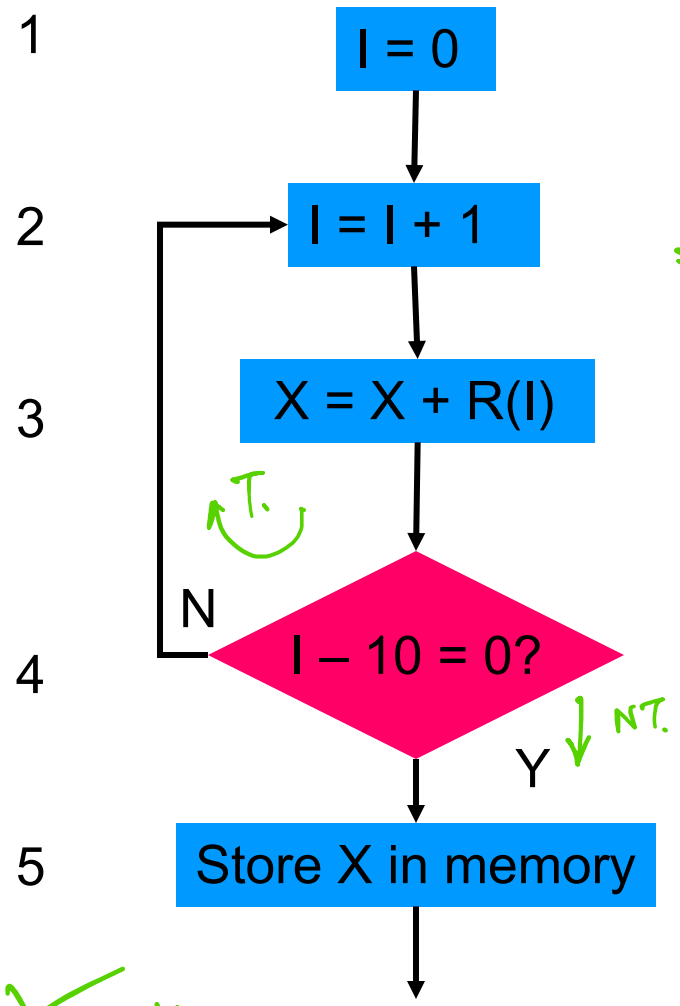
- Useful for program loops.
- A one-bit prediction scheme: a one-bit buffer carries a "history bit" that tells what happened on the last branch instruction
  - History bit = 1, branch was taken
  - History bit = 0, branch was not taken



Branch Prediction



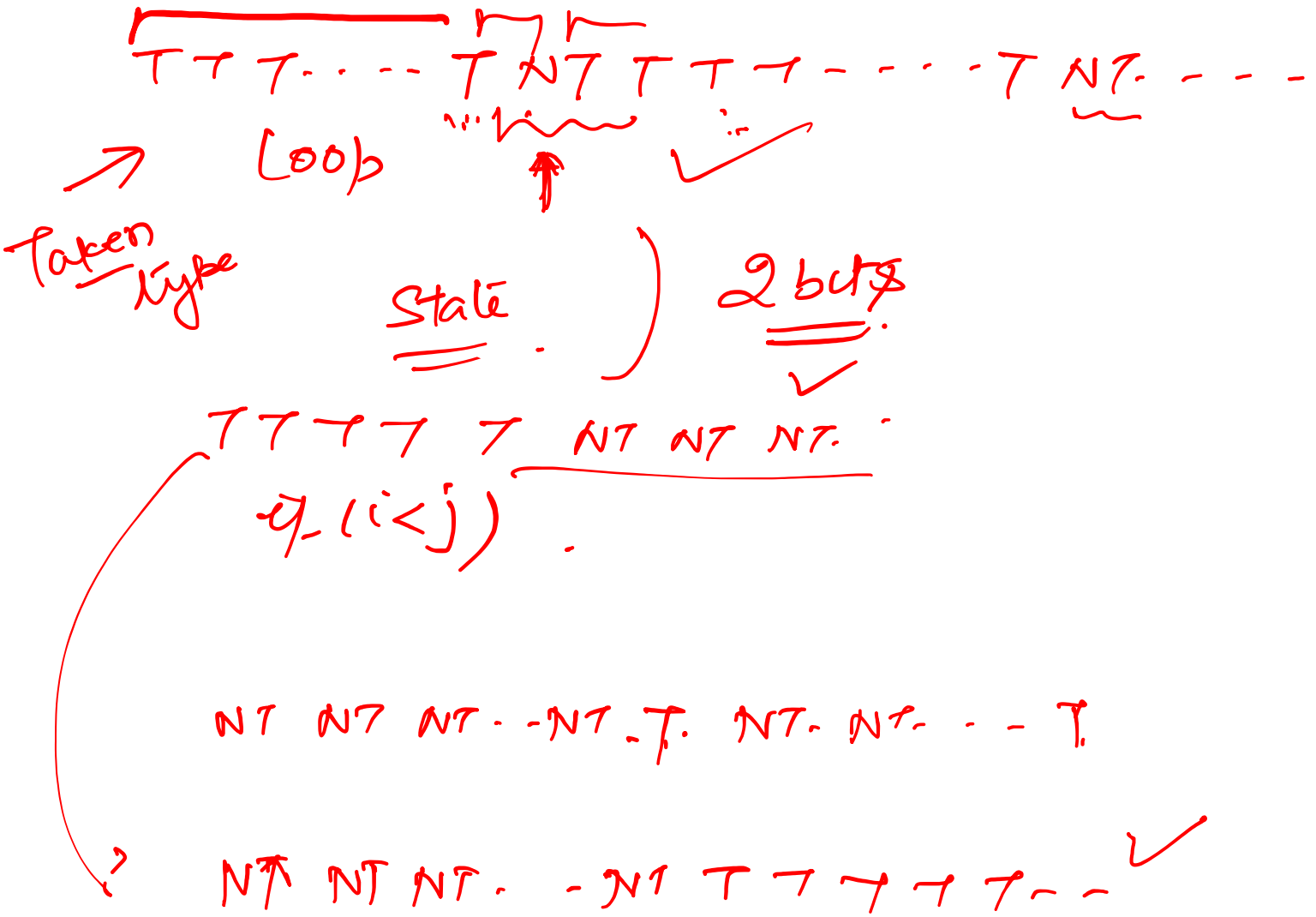
Branch Prediction for a Loop



Execution of Instruction 4

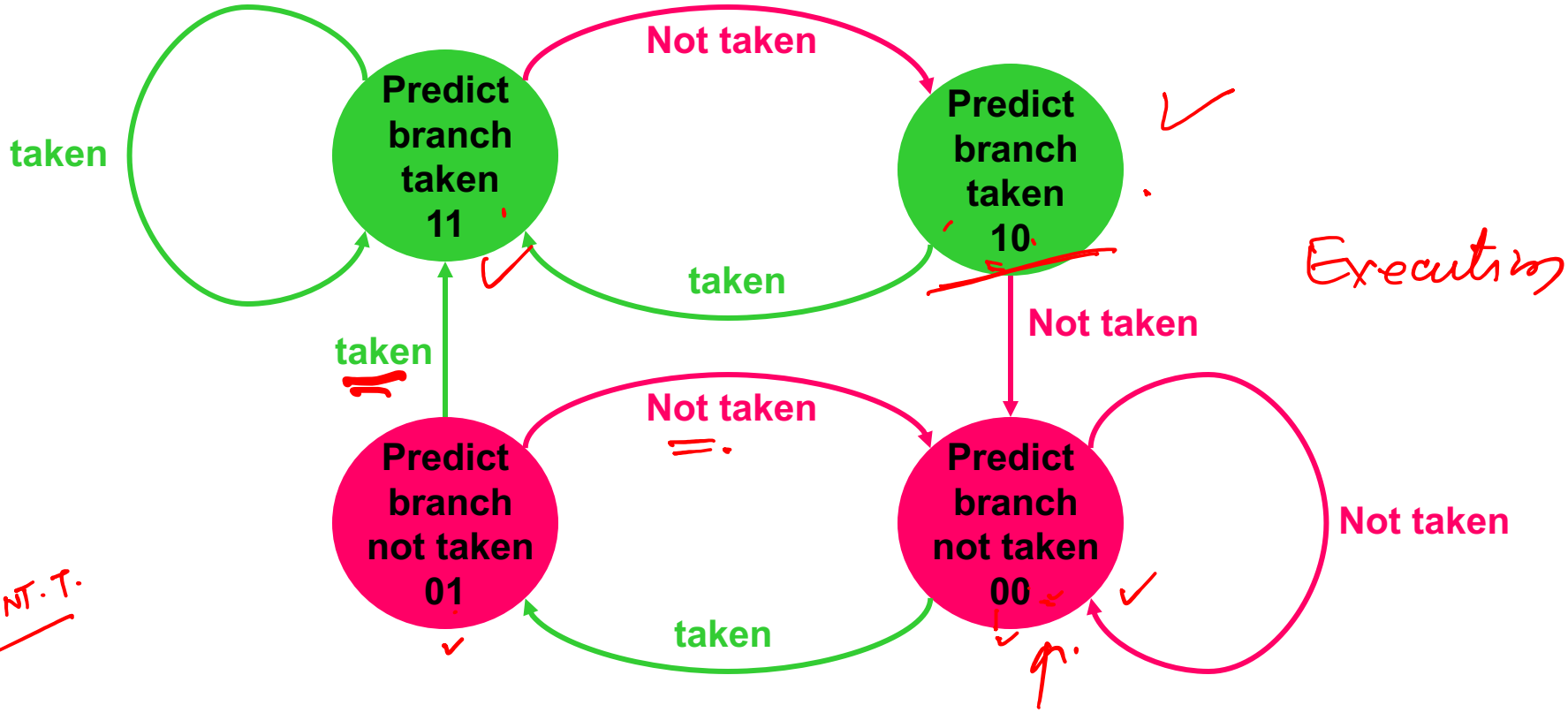
Execu-tion seq.	Old hist. bit	Next instr.			New hist. bit	Predi ction
		Pred.	I	Act.		
1	0	5	1	2	1	Bad
2	1	2	2	2	1	Good
3	1	2	3	2	1	Good
4	1	2	4	2	1	Good
5	1	2	5	2	1	Good
6	1	2	6	2	1	Good
7	1	2	7	2	1	Good
8	1	2	8	2	1	Good
9	1	2	9	2	1	Good
10	1	2	10	5	0	Bad

h.bit = 0 *branch not taken*, h.bit = 1 *branch taken*.

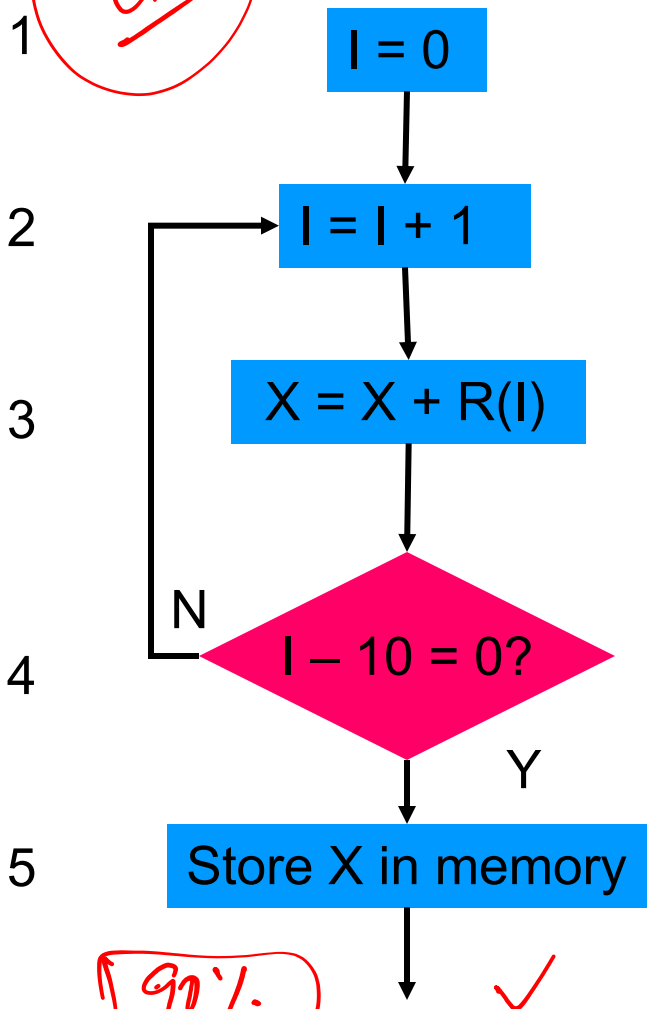

T T T . . . T NT T T T . . . T NT . . .
Loop
Taken type
State
2 bits
T T T T T NT NT NT .
q (i < j)
NT NT NT . . NT . T . NT . NT . . . T
NT NT NT . . NT T T T T T . .

Two-Bit Prediction Buffer

- Can improve correct prediction statistics.

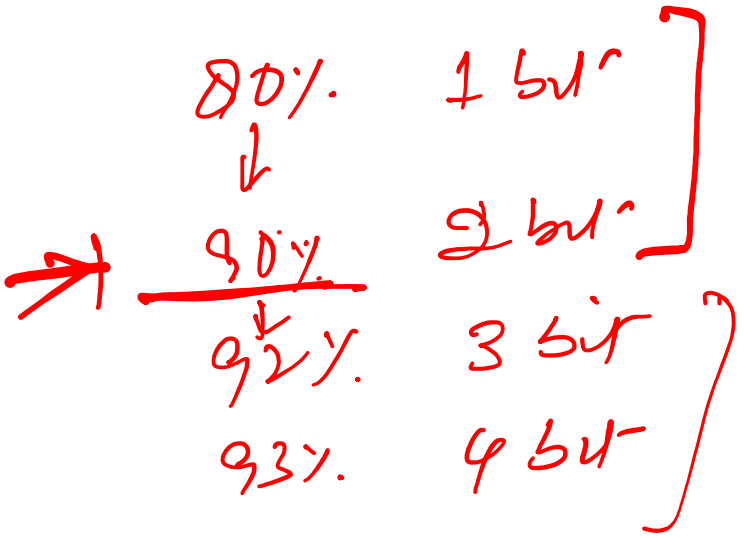


Branch Prediction for a Loop



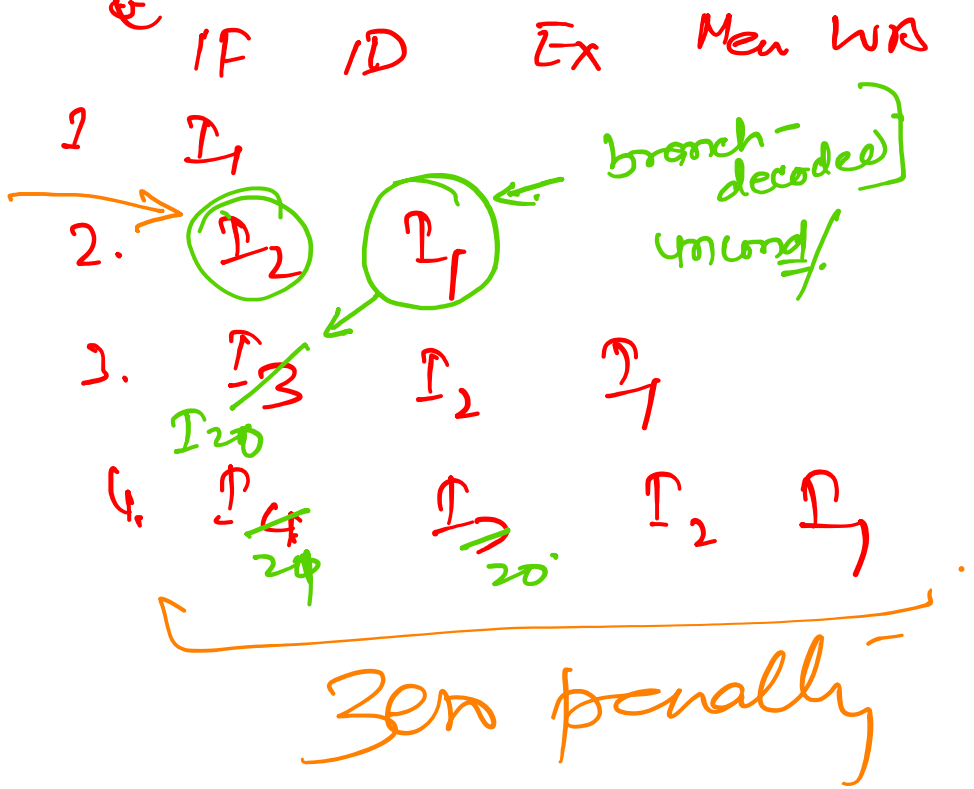
Execution of Instruction 4

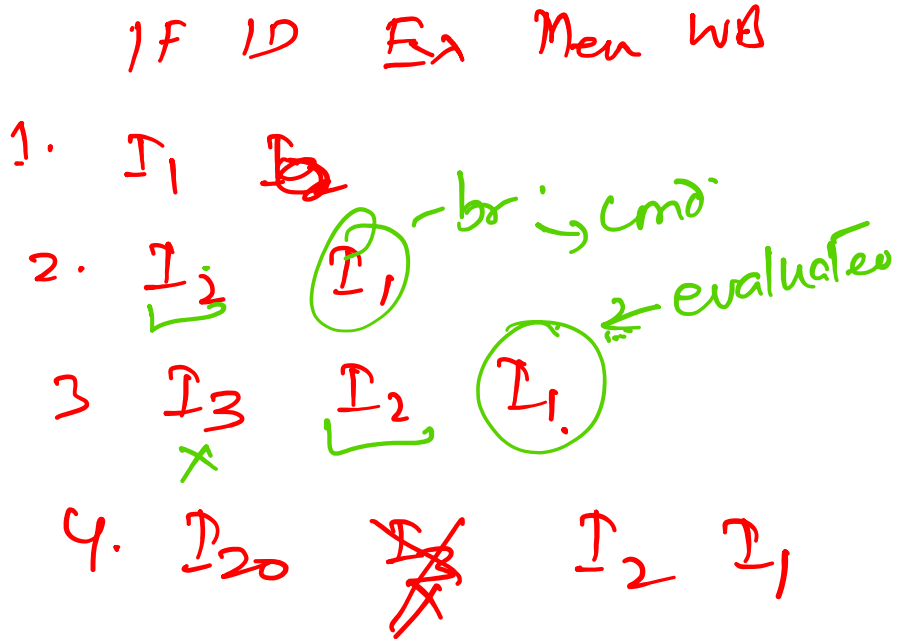
Execu-tion seq.	Old Pred. Buf	Next instr.			New pred. Buf	Predi ction
		Pred.	I	Act.		
1	10	2	1	2	11	Good
2	11	2	2	2	11	Good
3	11	2	3	2	11	Good
4	11	2	4	2	11	Good
5	11	2	5	2	11	Good
6	11	2	6	2	11	Good
7	11	2	7	2	11	Good
8	11	2	8	2	11	Good
9	11	2	9	2	11	Good
10	11	2	10	5	10	Bad

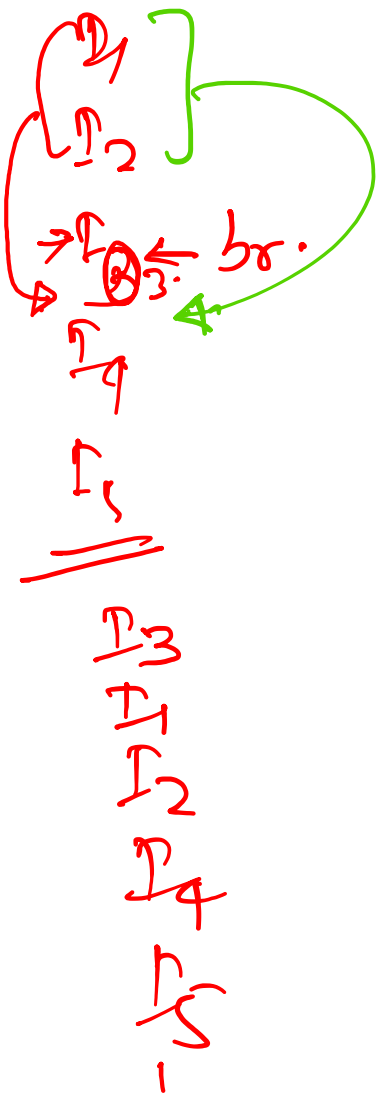

80%.
1 bit
90%.
2 bit
92%.
3 bit
93%.
4 bit

Delayed Branch Execution

ISA




IF ID Ex Mem WB
1. I1
2. I2
I1
-br → cmd
evaluated
3 I3
I2
I1
4. I20
I2 I1



1. I_3
2. I_3
3. I_2 I_1 I_3 — evaluated
4. I_{20} I_2 I_1 I_3
5. I_{21} I_{20} I_2 I_1 I_3

no - penalty

Delayed Branch Example

- Stall on branch

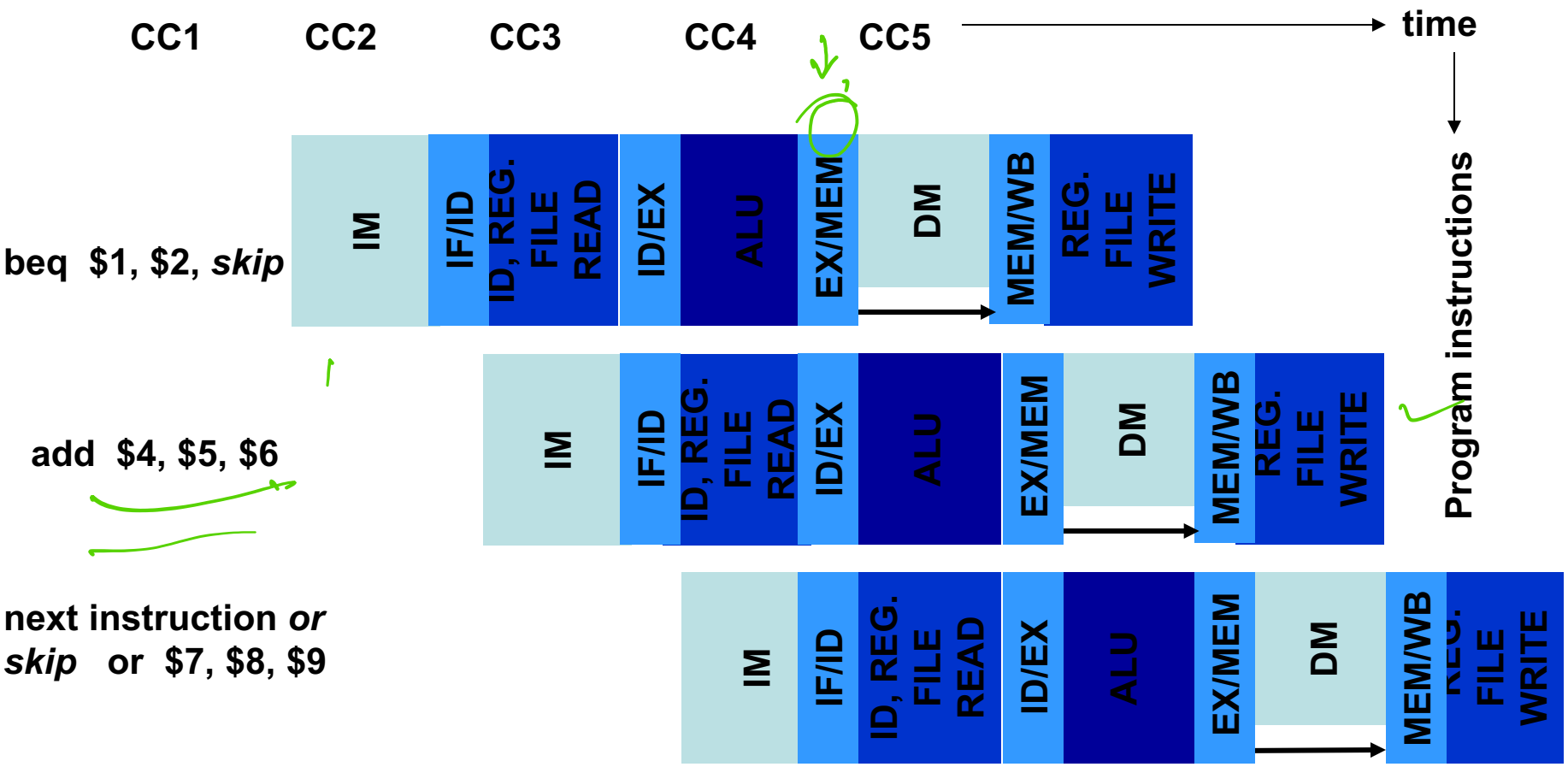
```
        add $4, $5, $6
        beq $1, $2, skip
        next instruction
        . . .
skip    or $7, $8, $9
```

- Delayed branch

```
        beq $1, $2, skip
        add $4, $5, $6
        next instruction
        . . .
skip    or $7, $8, $9
```

Instruction executed irrespective of branch decision

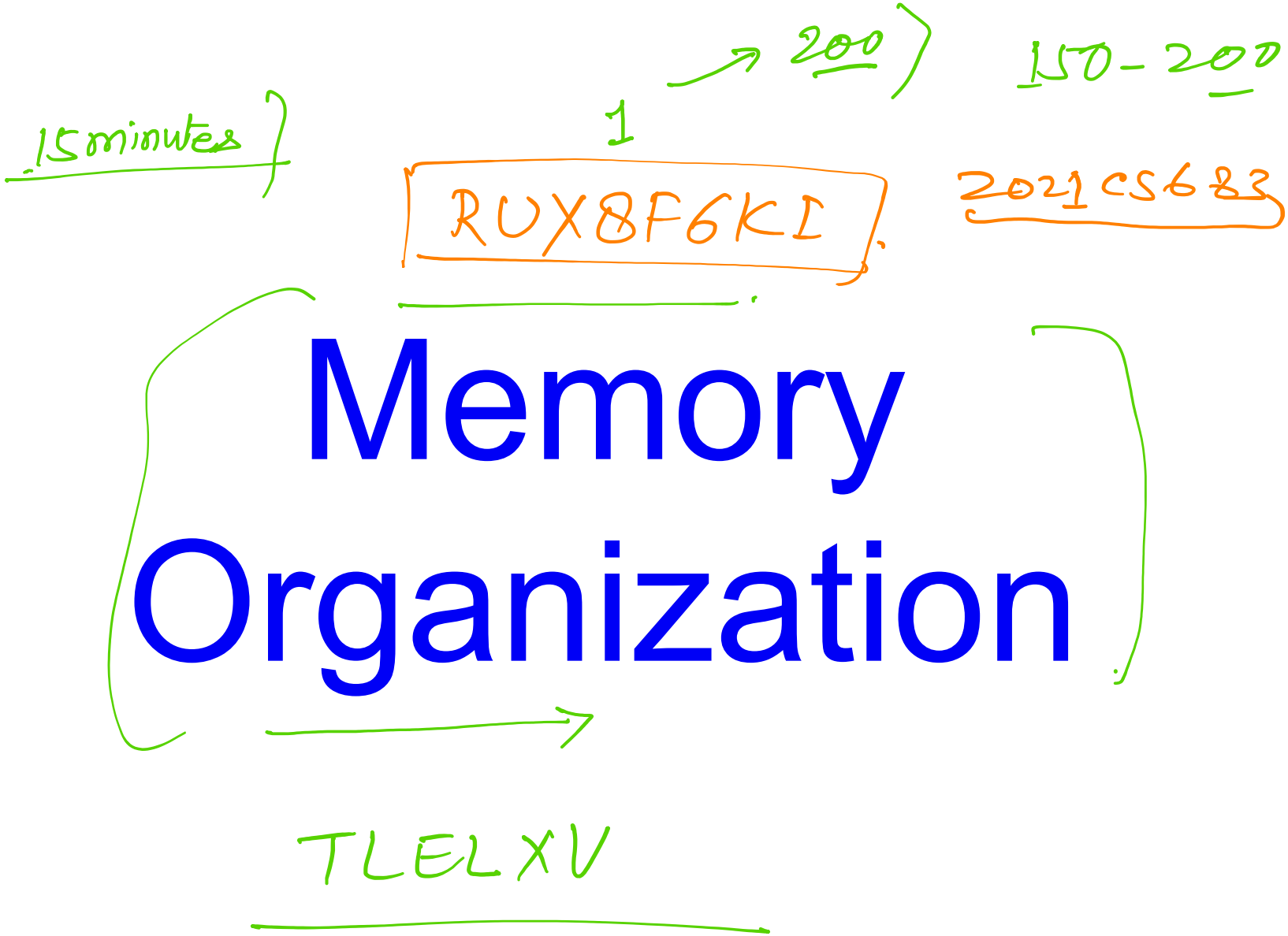
Delayed Branch



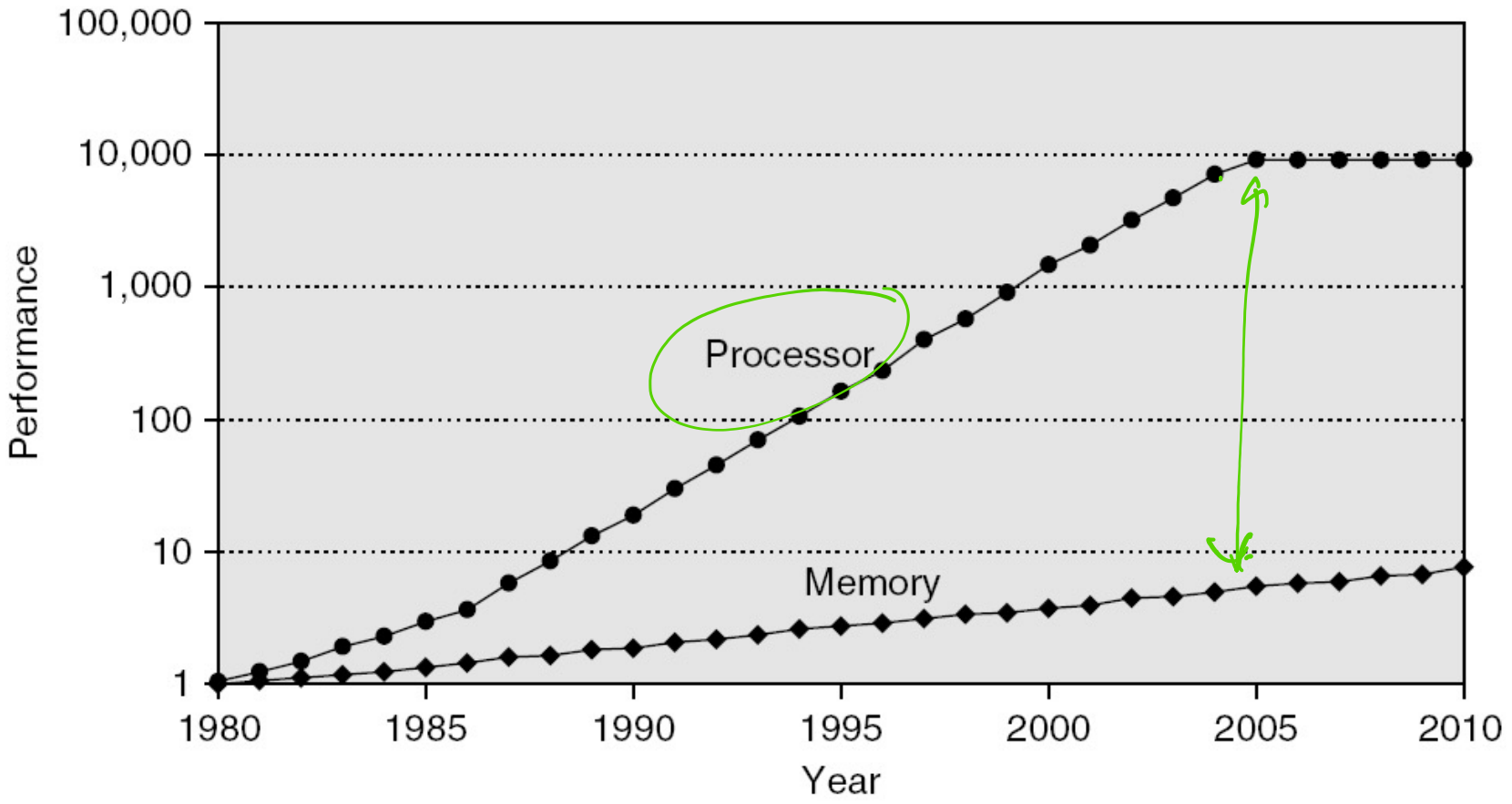
Summary: Hazards

- Structural hazards
  - Cause: resource conflict
  - Remedies: (i) hardware resources, (ii) stall (bubble)
- Data hazards
  - Cause: data unavailablity
  - Remedies: (i) forwarding, (ii) stall (bubble), (iii) code reordering
- Control hazards
  - Cause: out-of-sequence execution (branch or jump)
  - Remedies: (i) stall (bubble), (ii) branch prediction/pipeline flush, (iii) delayed branch/pipeline flush

Memory Organization



Memory Performance Gap



Why Memory Hierarchy?

- Need lots of bandwidth

$$BW = \frac{1.0inst}{cycle} \times \left[\frac{1Ifetch}{inst} \times \frac{4B}{Ifetch} + \frac{0.4Dref}{inst} \times \frac{4B}{Dref} \right] \times \frac{1Gcycles}{\text{sec}}$$
$$= \frac{5.6GB}{\text{sec}}$$

- Need lots of storage
  - 64MB (minimum) to multiple TB
- Must be cheap per bit
  - (TB x anything) is a lot of money!
- These requirements seem incompatible

Thank You