RISC Design
Memory System

Virendra Singh
Professor
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

EE-739: Processor Design

Lecture 24 (17 March 2021)

CADSL

Control Hazard

(1) Stall. (2) Branch delay slot

(3) Branch prediction — 98-99% accuracy

Track history → take the same decision which we took last time. → bi-

Branch History TABLE

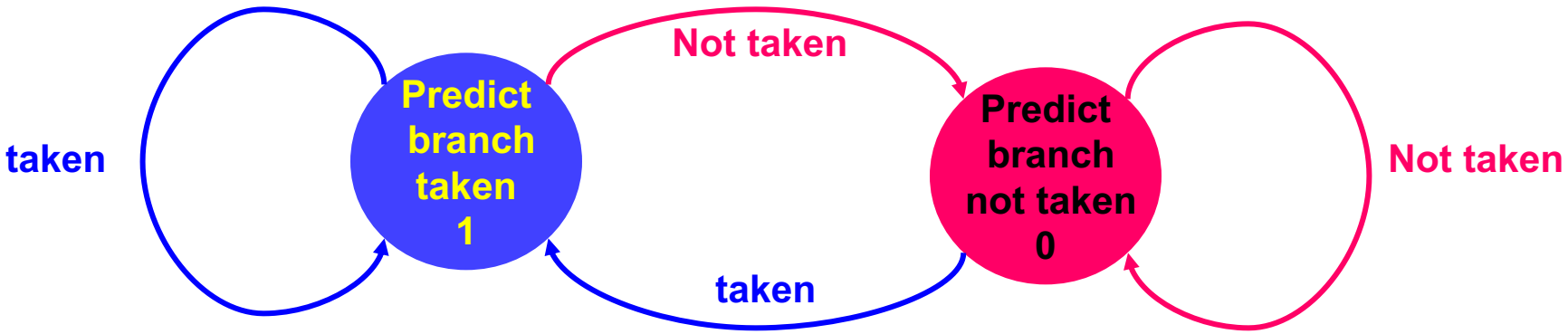
PC	BTA	HB
200	575	1

← modified from execution stage

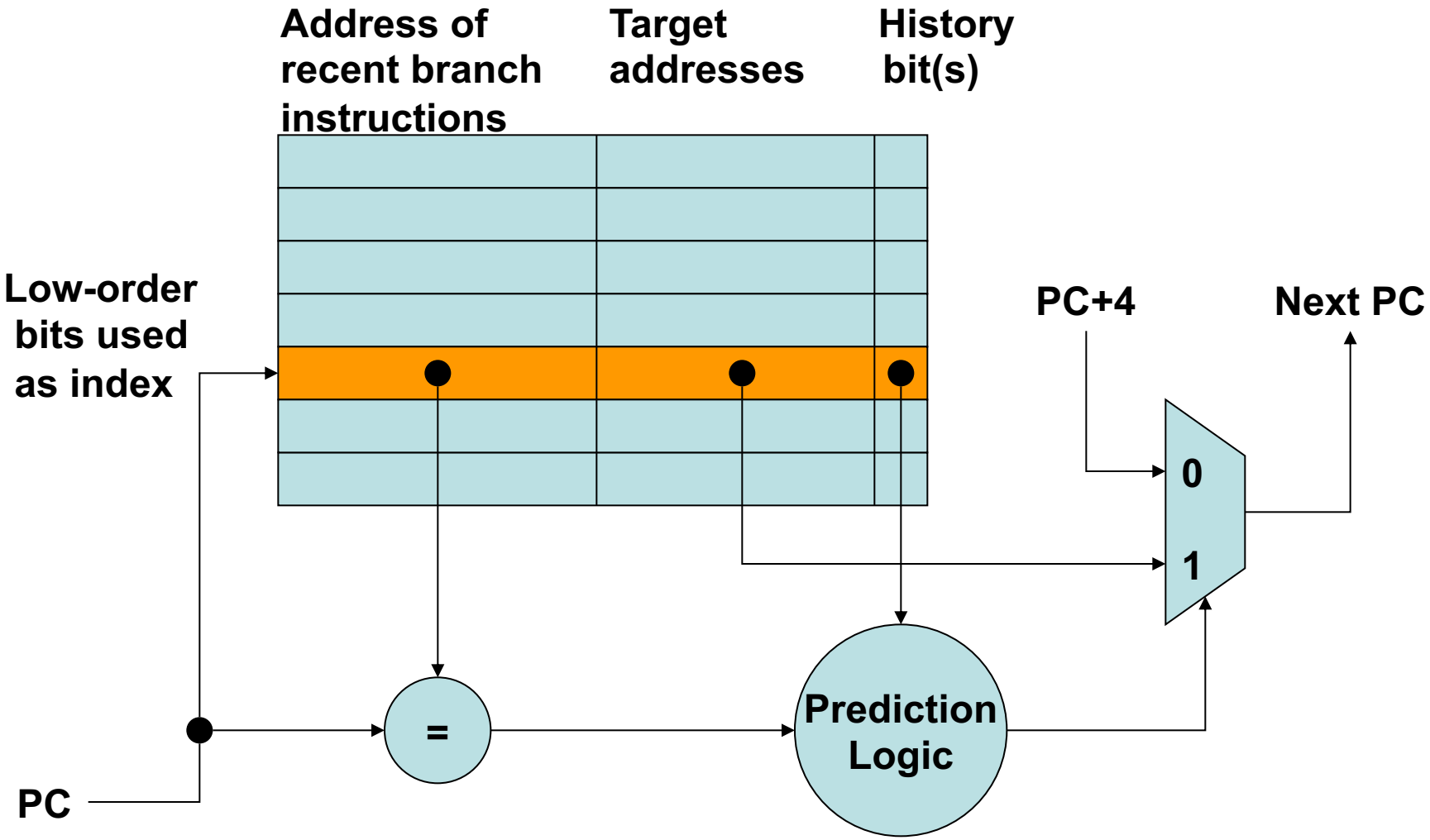
Fetch

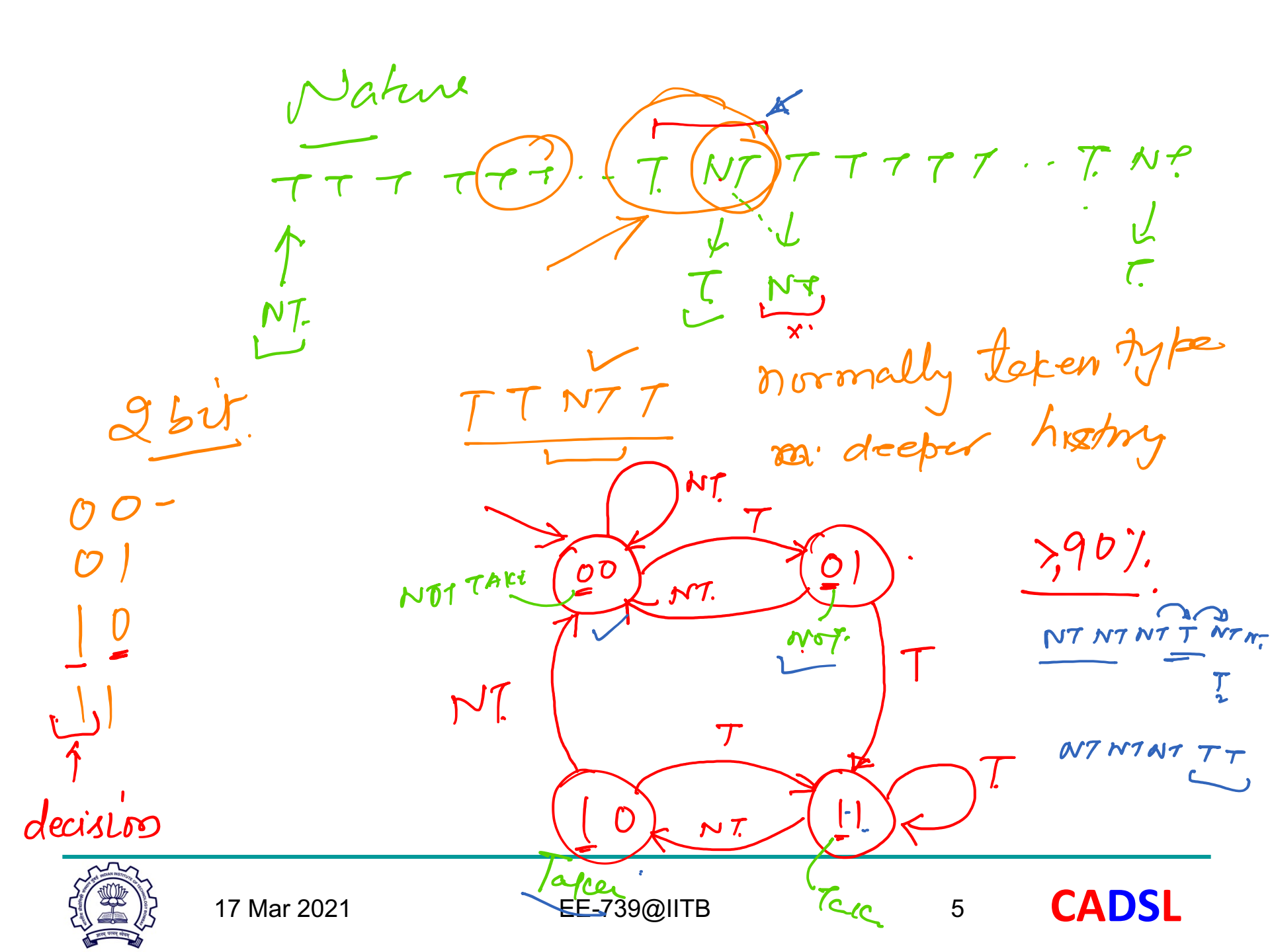
Branch Prediction

- 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

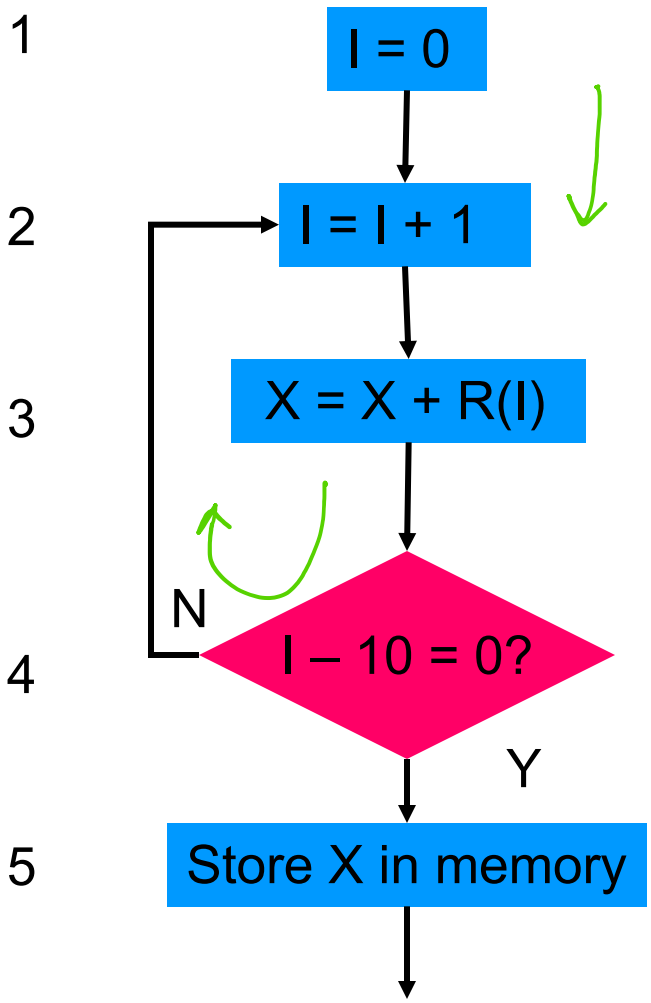


Branch Prediction




Nature
T T T T T T . . T NT T T T T T . . T NT
NT
T
NT
2 bit
TTNTT
normally taken type
ex: deeper history
00
01
10
11
decision
NT
T
NOT TAKE
NT
NOT
T
NT
T
T
NT
T
>90%
NT NT NT T NT
NT NT NT TT
Taken
Take

Branch Prediction for a Loop



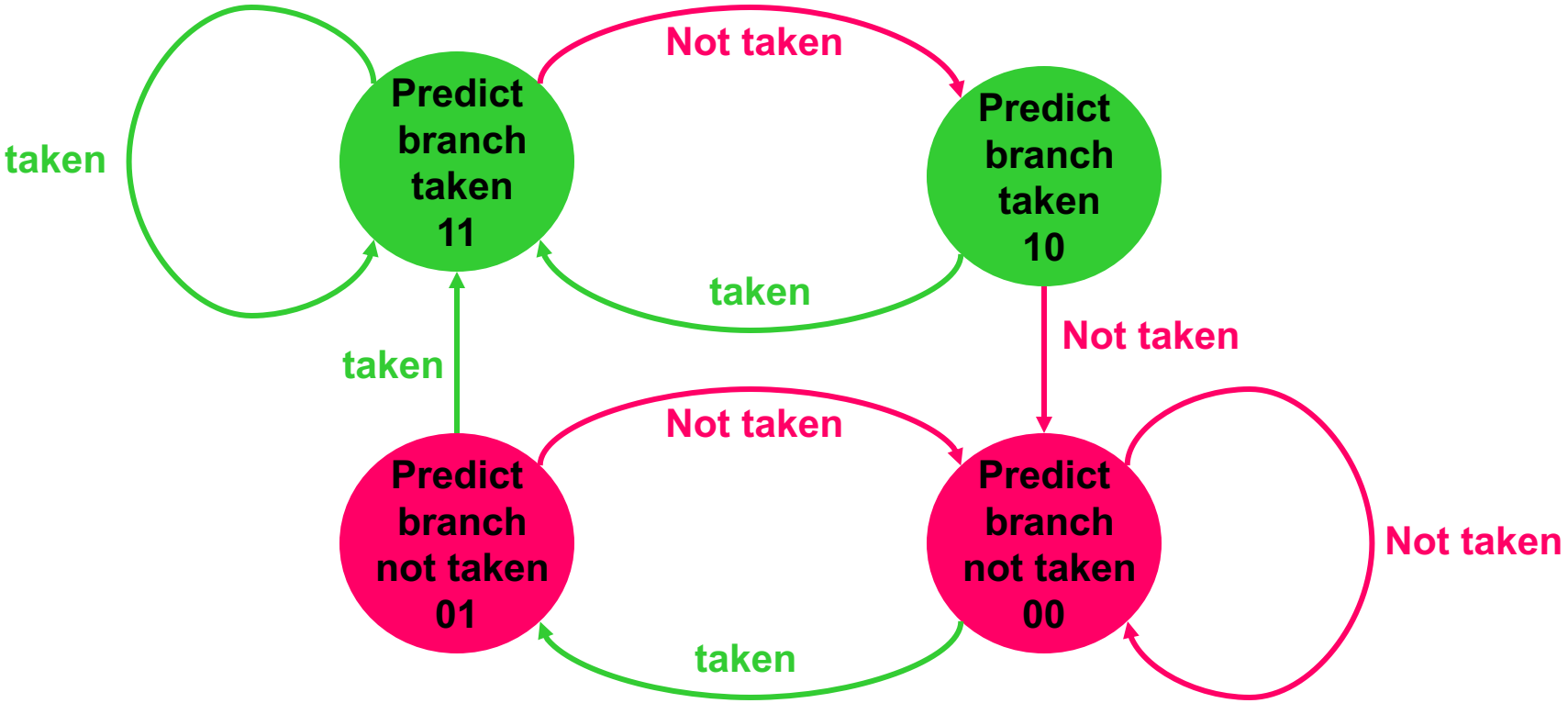
Execution of Instruction 4

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

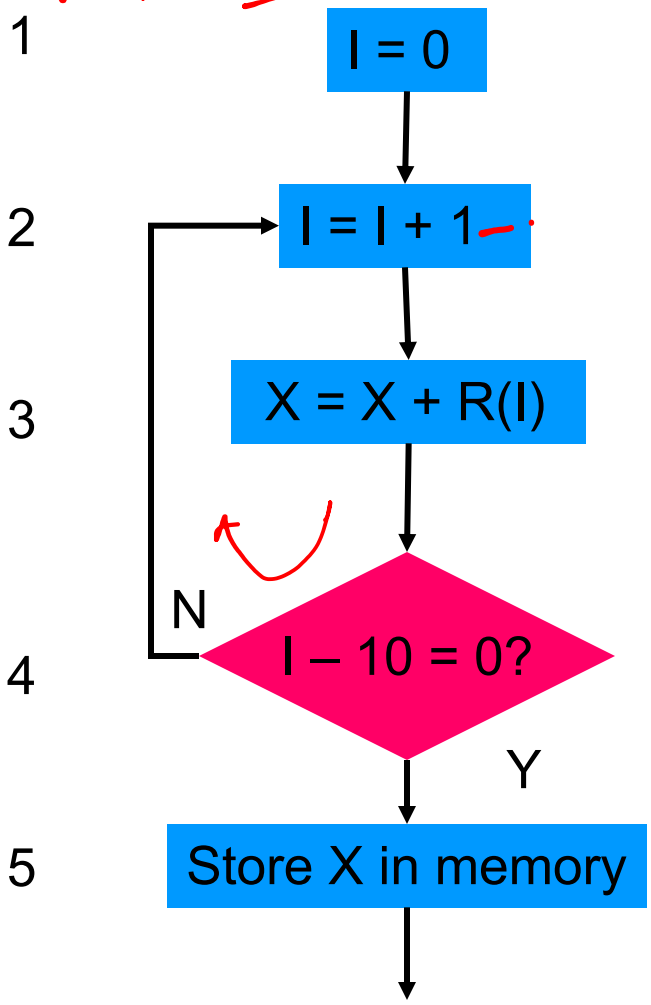
h.bit = 0 *branch not taken*, h.bit = 1 *branch taken*.

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

1 bit 85%.
2 bit 90%. } → 90–92%. accuracy

3 bit } → 93%.
4 bit } → 94%.

70% → 85% → 90%.

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

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

Large memory
increasing size of
program & data
↓
SRAM
very expensive

Fetch - Decode - Exec. - Mem - WB
↓
200
200 i

(2) DRAM.
(slow)

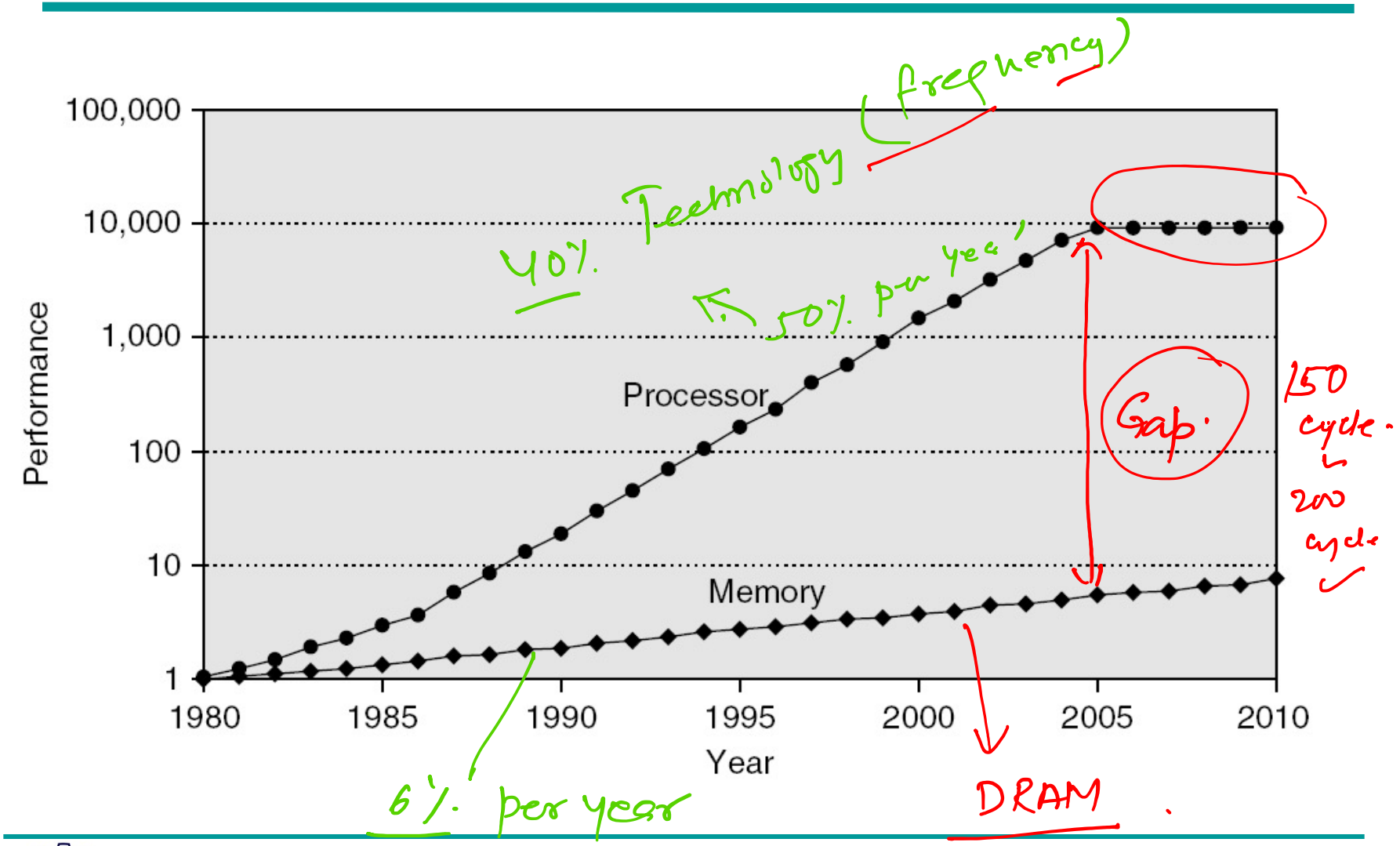
Memory System

Single cycle memory access. → 200 cycles

Big Picture

- Memory
  - Just an "ocean of bits"
  - Many technologies are available
- Key issues
  - Technology (how bits are stored)
  - Placement (where bits are stored)
  - Identification (finding the right bits)
  - Replacement (finding space for new bits)
  - Write policy (propagating changes to bits)
- Must answer these regardless of memory type

Memory Performance Gap



Types of Memory

Type	Size	Speed	Cost/bit
Register	< 1KB	< 1ns	$$$$
On-chip SRAM	8KB-6MB	< 10ns	$$$
Off-chip SRAM	1Mb – 16Mb	< 20ns	$$
DRAM	64MB – 1TB	< 100ns	$
Disk	40GB – 1PB	< 20ms	~0

Memory System

- Programmers want unlimited amounts of memory with low latency
- Fast memory technology is more expensive per bit than slower memory
- Solution: organize memory system into a hierarchy
  - Entire addressable memory space available in largest, slowest memory
  - Incrementally smaller and faster memories, each containing a subset of the memory below it, proceed in steps up toward the processor
- Temporal and spatial locality insures that nearly all references can be found in smaller memories
  - Gives the allusion of a large, fast memory being presented to the processor

for(i=0; i<100; i++)

a[i] = b[i] + c[i]

temporal locality

e = b + c

d = a + e

p = q + r

for(i=0; i<100)

instructions

spatial locality

B[i] B[0] B[1]

B[2] B[3]

for(i=0; i<100; i++)

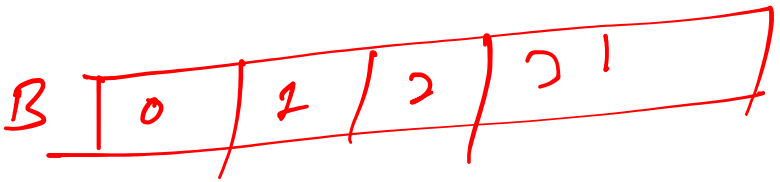
{

a[i] = b[i] + c[i]

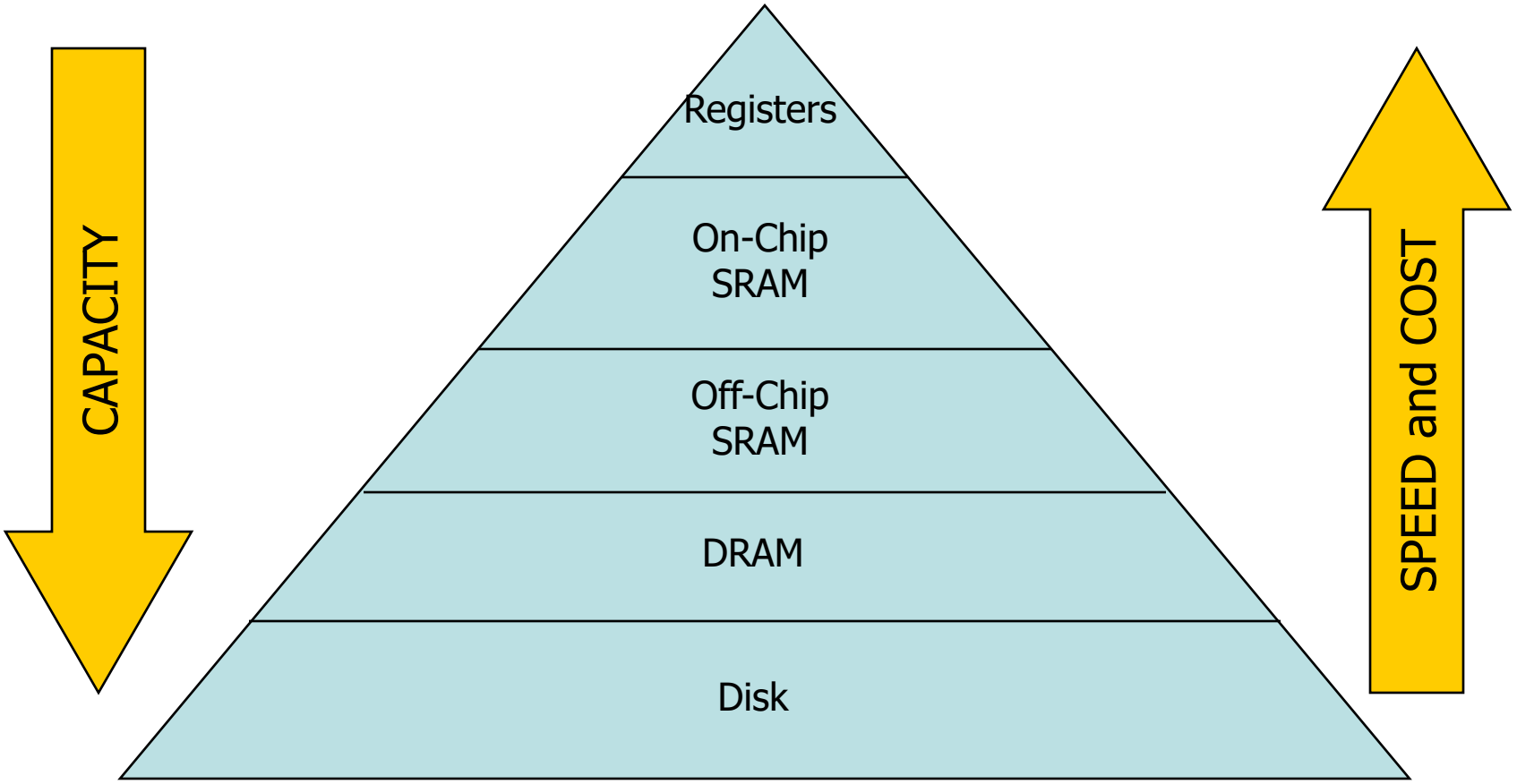
d[i] = a[i] + e[i]

p[i] = q[i] * r[i]

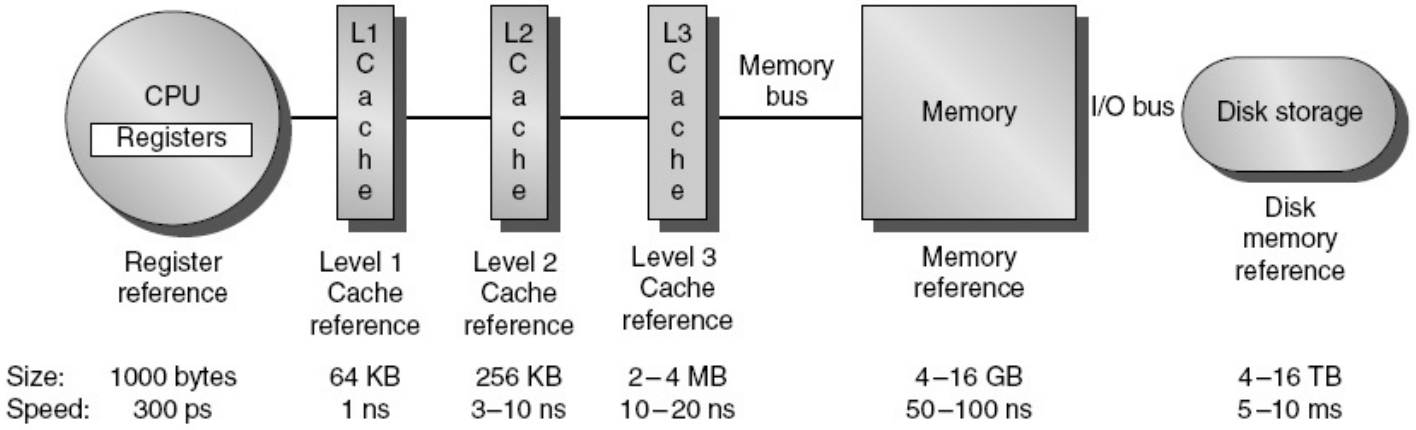
}



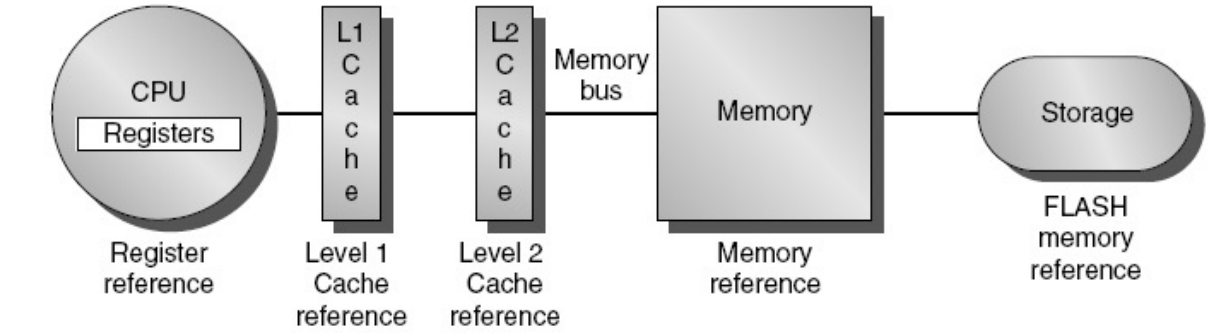
Memory Hierarchy



Memory Hierarchy



(a) Memory hierarchy for server



(b) Memory hierarchy for a personal mobile device

Why Memory Hierarchy?

- Need lots of bandwidth

$$BW = \frac{1.0inst}{cycle} \times \left[\frac{1Ifetch}{inst} \times \frac{4B}{Ifetch} + \frac{0.2Dref}{inst} \times \frac{4B}{Dref} \right] \times \frac{1Gcycles}{\text{sec}}$$
$$= \frac{4.8GB}{\text{sec}}$$

- 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

$$bw = \frac{data}{sec} = \frac{data}{fetch} \times \frac{fetch}{inst} \times \frac{inst}{cycle} \times \frac{cycle}{sec}$$

3 GHz

inst $(4B + 0.3 \times 4B) \times 1 \times 1 \times 3 \times 10^9$

data only for I/O / $ store

≈ 30%.

$= 4 \times 1.3 \times 3 \times 10^9 = 12 \times 1.3 \times 10^9$

$= 15.6 \times 10^9$

= 15.6 GBps

memory Communication.

Memory Hierarchy Design

- Memory hierarchy design becomes more crucial with recent multi-core processors:
  - Aggregate peak bandwidth grows with # cores:
    - Intel Core i7 can generate two references per core per clock
    - Four cores and 3.2 GHz clock
      - 25.6 billion 64-bit data references/second +
      - 12.8 billion 128-bit instruction references
      - = 409.6 GB/s!
    - DRAM bandwidth is only 6% of this (25 GB/s)
    - Requires:
      - Multi-port, pipelined caches
      - Two levels of cache per core
      - Shared third-level cache on chip

Why Locality?

- Analogy:
  - Library (Disk)
  - Bookshelf (Main memory)
  - Stack of books on desk (off-chip cache)
  - Opened book on desk (on-chip cache)
- Likelihood of:
  - Referring to same book or chapter again?
    - Probability decays over time
    - Book moves to bottom of stack, then bookshelf, then library
  - Referring to chapter n+1 if looking at chapter n?

Thank You