

Advanced Computer Architecture

Lecture 13

Introduction to Cache Memory



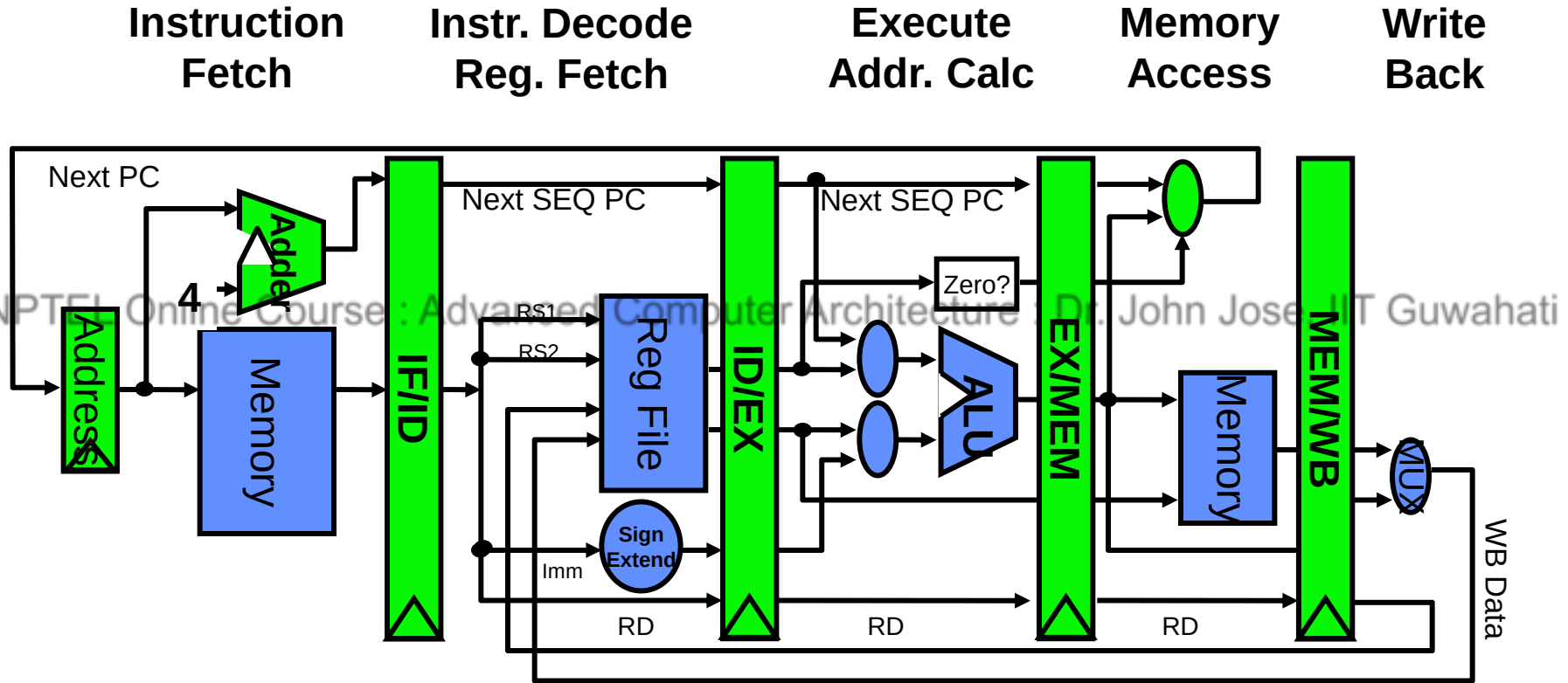
NPTEL Online Course : Advanced Computer Architecture : Dr. John Jose, IIT Guwahati

Dr. John Jose

Assistant Professor

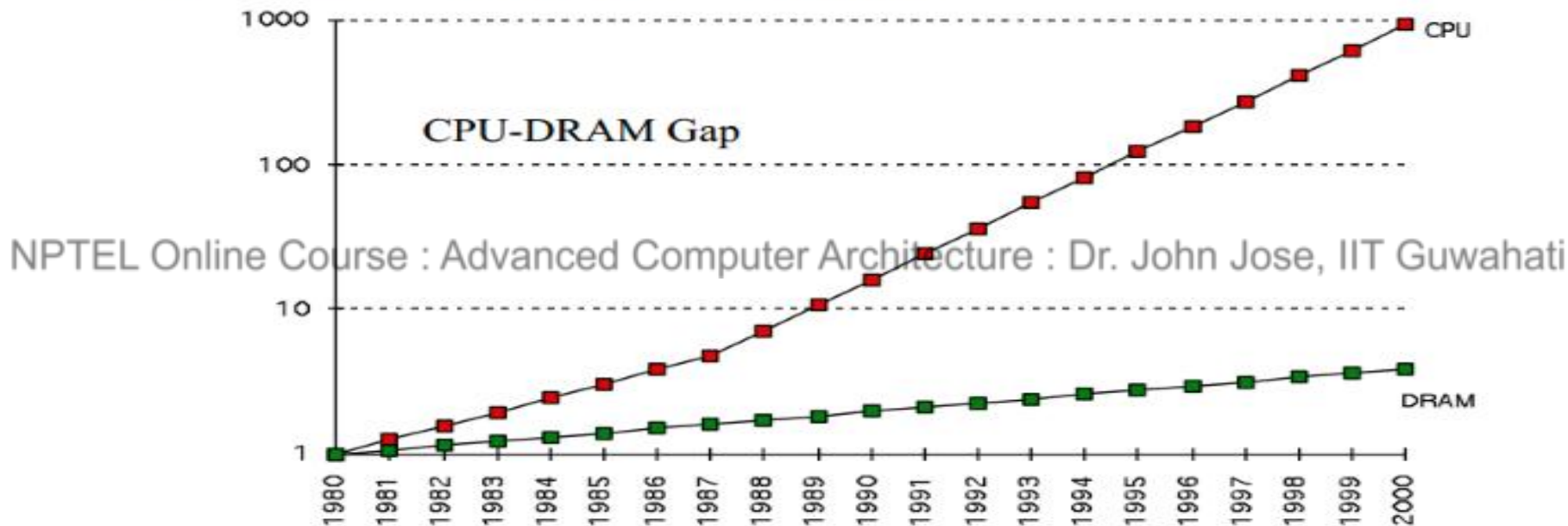
**Department of Computer Science & Engineering
Indian Institute of Technology Guwahati, Assam.**

Pipelined RISC Data path



Processor Memory Performance Gap

■ Processor vs Memory Performance



1980: no cache in microprocessor;

1995 2-level cache

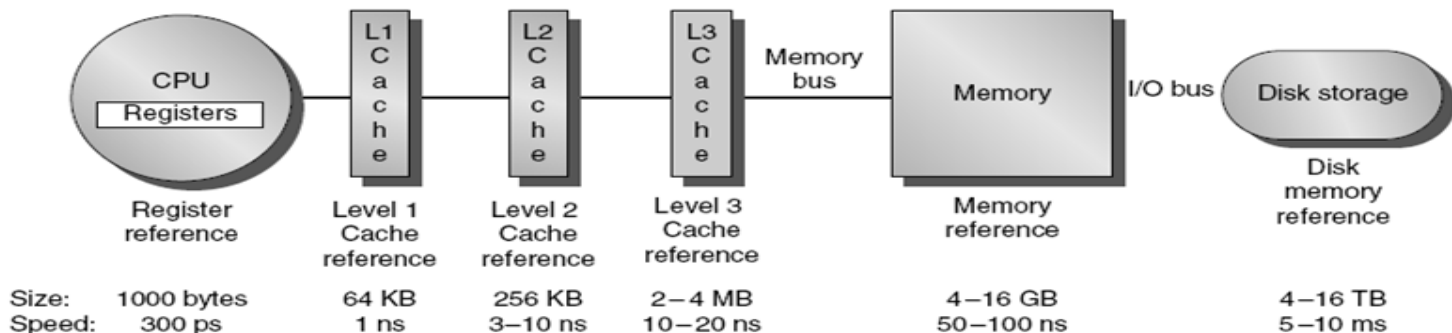
NPTEL Online Course : Advanced Computer Architecture : Dr. John Jose, IIT Guwahati



Role of memory

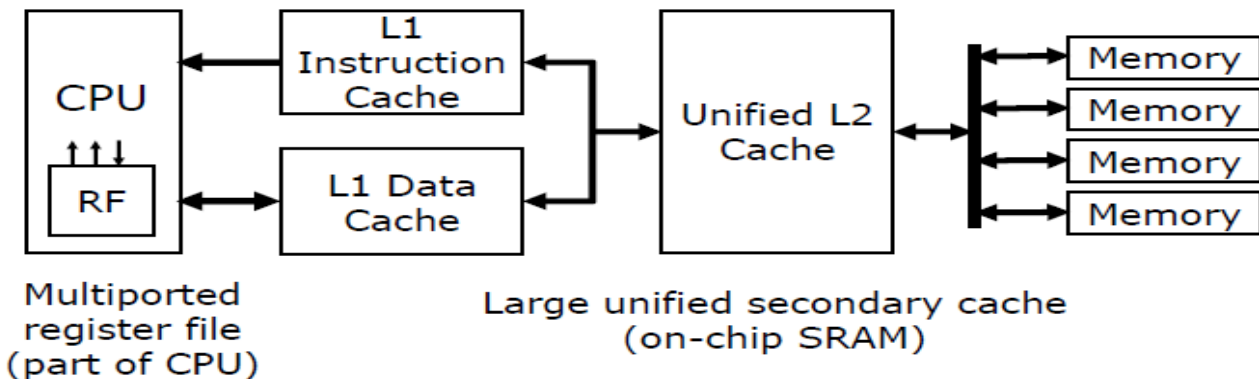
- ❖ Programmers want **unlimited** amount of **fast memory**
- ❖ Create the **illusion** of a **very large** and **fast** memory
- ❖ Implement the memory of a computer as a hierarchy
- ❖ Multiple levels of memory with different speeds and sizes
- ❖ Entire addressable memory space available in largest, slowest memory
- ❖ Keep the smaller and faster memories close to the processor and the slower, large memory below that.

Memory Hierarchy



Split instruction & data
primary caches
(on-chip SRAM)

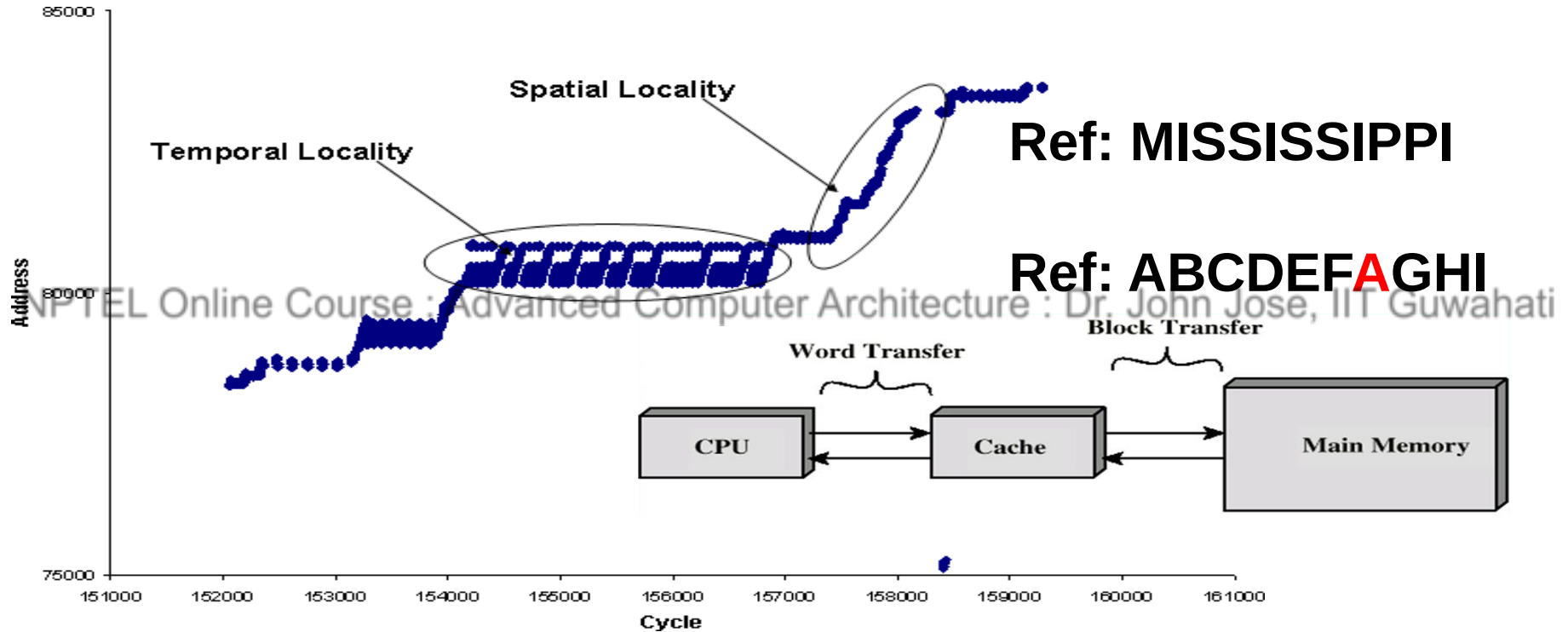
Multiple interleaved
memory banks
(DRAM)



Cache Memory - Introduction

- ❖ Cache is a small, fast buffer between processor and memory
- ❖ Old values will be removed from cache to make space for new values
- ❖ **Principle of Locality** : Programs access a relatively small portion of their address space at any instant of time
- ❖ **Temporal Locality** : If an item is referenced, it will tend to be referenced again soon
- ❖ **Spatial Locality** : If an item is referenced, items whose addresses are close by will tend to be referenced soon

Access Patterns



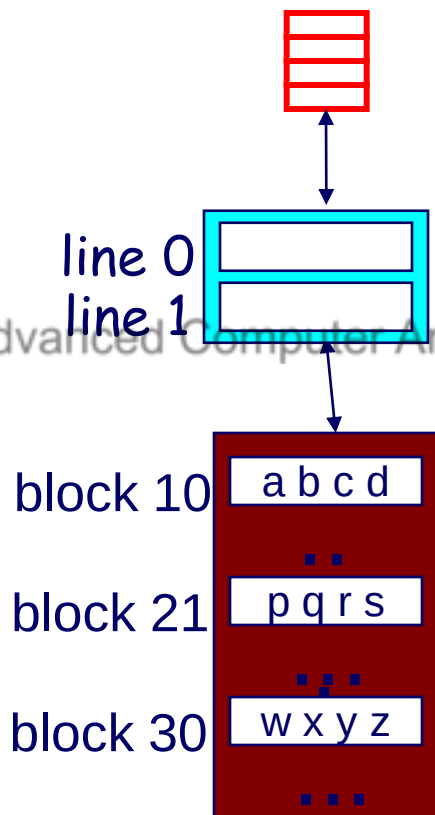
Cache Fundamentals

- ❖ **Block/Line** : Minimum unit of information that can be either present or not present in a cache level
- ❖ **Hit** : An access where the data requested by the processor is present in the cache
- ❖ **Miss** : An access where the data requested by the processor is not present in the cache
- ❖ **Hit Time** : Time to access the cache memory block and return the data to the processor.
- ❖ **Hit Rate / Miss Rate**: Fraction of memory access found (**not found**) in the cache
- ❖ **Miss Penalty** : Time to replace a block in the cache with the corresponding block from the next level.

CPU – Cache Interaction

❖ The transfer unit between the CPU register file and the cache is a 4-byte word

❖ The transfer unit between the cache and main memory is a 4-word block (16B)



❖ The tiny, very fast CPU register file has room for four 4-byte words

❖ The small fast **L1 cache** has room for two 4-word blocks

❖ The big slow main memory has room for many 4-word blocks

General Organization of a Cache

❖ Cache is an array of sets

❖ Each set contains one or more lines

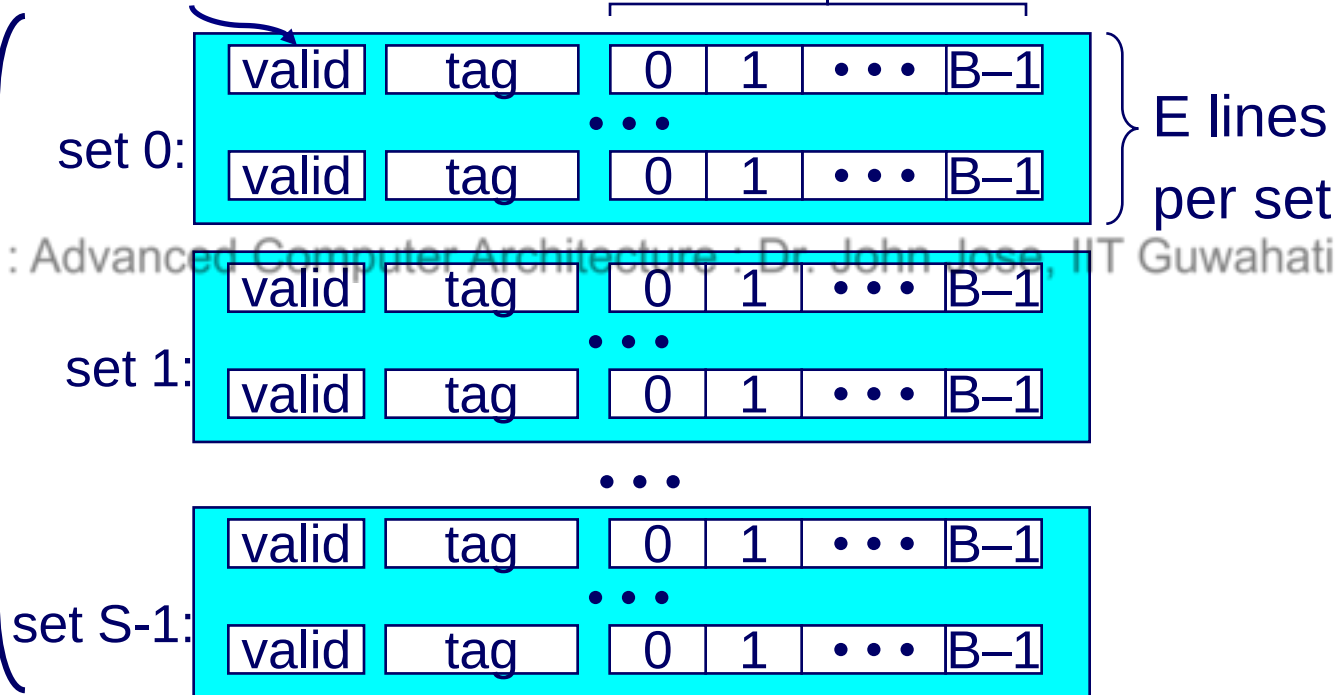
❖ Each line holds a block of data

$$S = 2^s \text{ sets}$$

1 valid bit
per line

t tag bits
per line

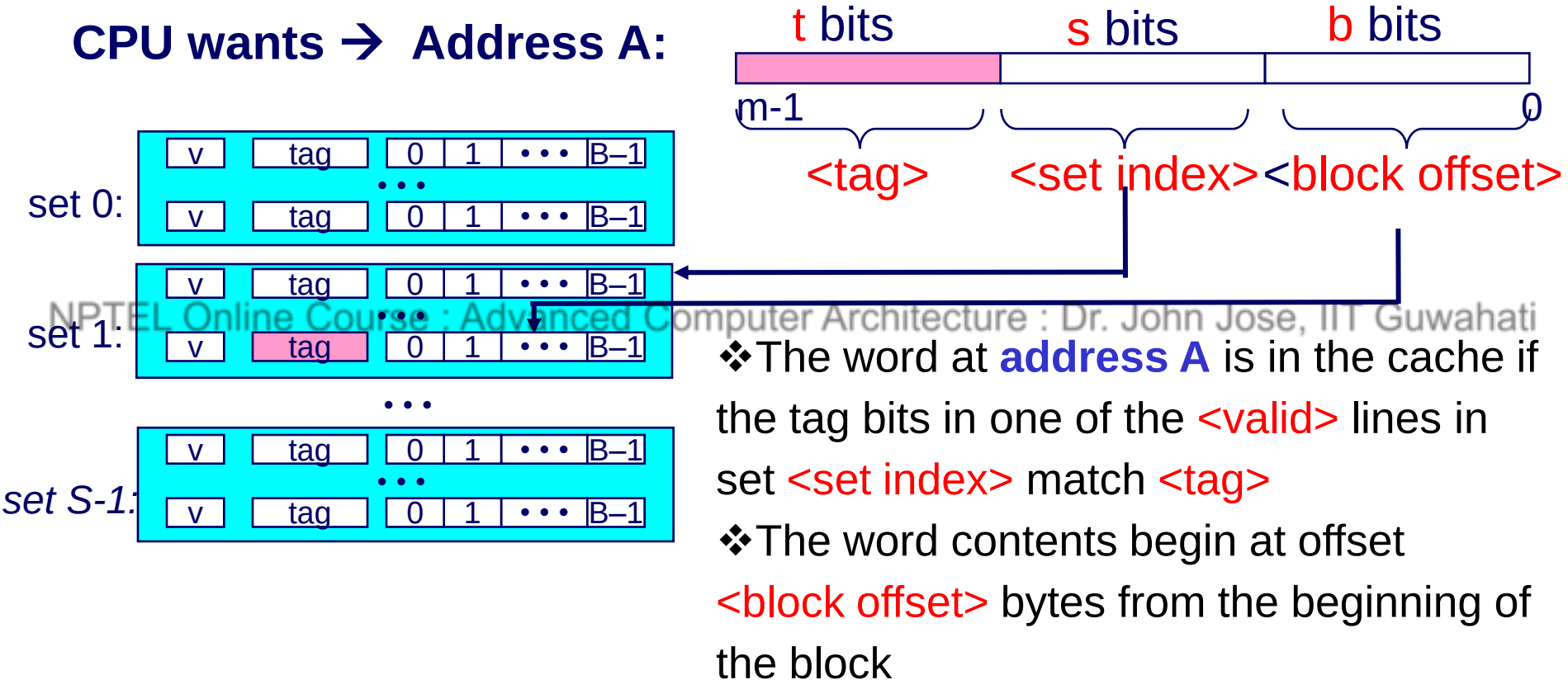
$B = 2^b$ bytes
per cache block



Cache size: $C = B \times E \times S$ data bytes

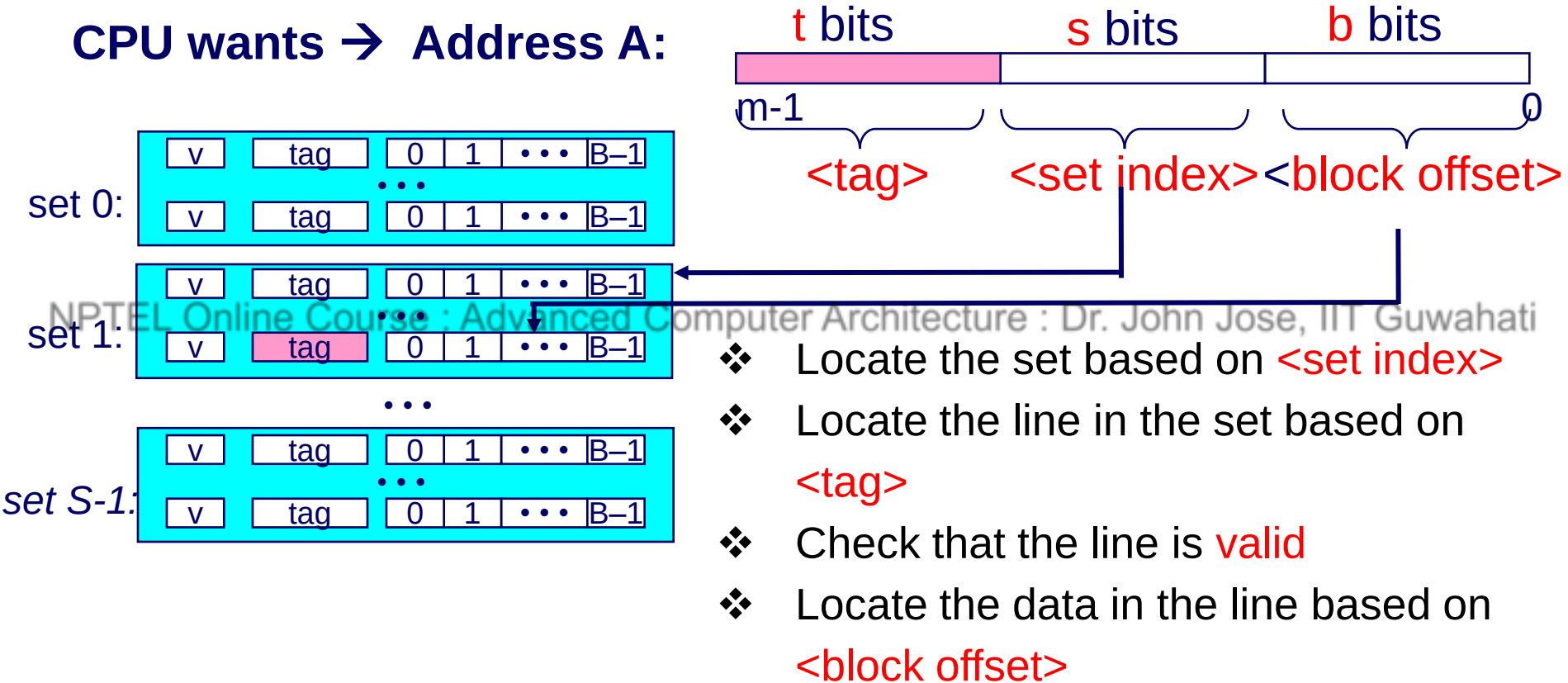
Addressing Caches

CPU wants → Address A:



Addressing Caches

CPU wants → Address A:



Index and Offset Calculations

A cache has 512 KB capacity, 4B word, 64B block size and 8-way set associative. The system is using 32 bit address. Given the address 0X ABC89984, which set of cache will be searched and specify which word of the selected cache block will be forwarded if it is a hit in cache?

NPTEL Online Course: Advanced Computer Architecture Dr. John Jose, IIT Guwahati

$$\# \text{ sets} = \text{CS}/(\text{BS} \times \text{A}) = 2^{19}/(2^6 \times 2^3) = 2^{10} = 1024 \text{ sets}$$

1 word = 4B , Hence 64 byte block has 16 words

Tag = 16

Index = 10

Offset = 6 (4+2)

0x ABC89984 = 1001 1001 1000 0100 → Set 614, word 1

0x 485669AC = 0110 1001 1010 1100 → Set 422, word 11

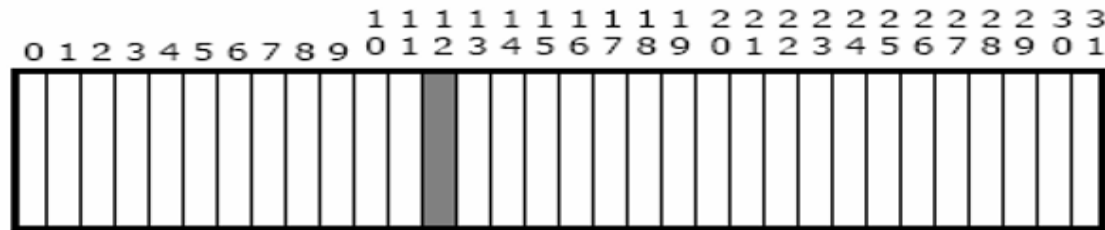
Four cache memory design choices

- ❖ Where can a block be placed in the cache?
 - **Block Placement**
- ❖ How is a block found if it is in the upper level?
 - **Block Identification**
- ❖ Which block should be replaced on a miss?
 - **Block Replacement**
- ❖ What happens on a write?
 - **Write Strategy**

Block Placement

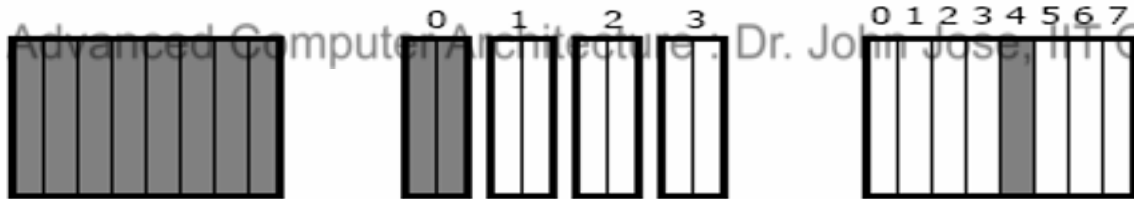
Block Number

Memory



Set Number

Cache



Fully
Associative

(2-way) Set
Associative

Direct
Mapped

block 12
can be placed

anywhere

anywhere in
set 0
($12 \bmod 4$)

only into
block 4
($12 \bmod 8$)

Cache Mapping / Block Placement

❖ Direct mapped

- ❖ Block can be placed in only one location
- ❖ $(\text{Block Number}) \bmod (\text{Number of blocks in cache})$

❖ Set associative

NPTEL Online Course : Advanced Computer Architecture : Dr. John Jose, IIT Guwahati

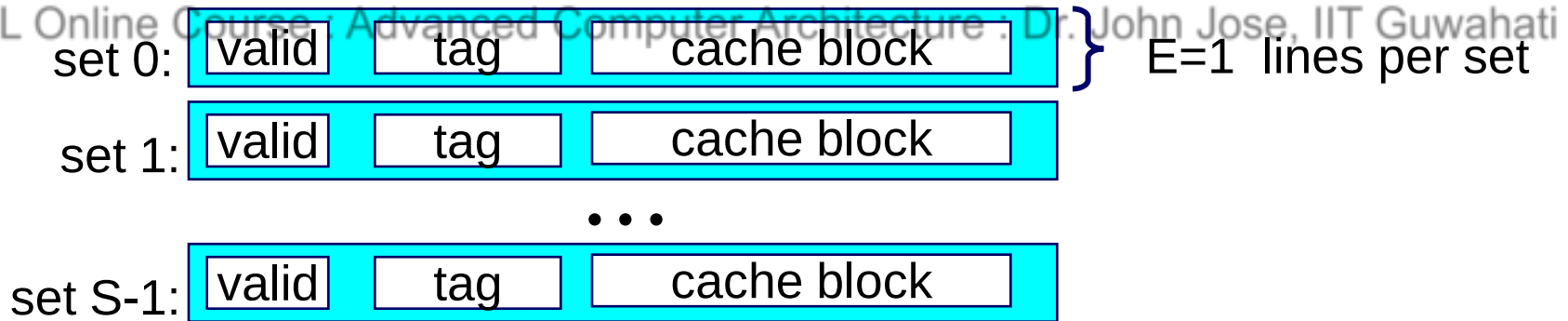
- ❖ Block can be placed in one among a list of locations
- ❖ $(\text{Block Number}) \bmod (\text{Number of sets})$

❖ Fully associative

- ❖ Block can be placed anywhere

Direct-Mapped Cache

- ❖ Simplest kind of cache, easy to build
- ❖ Only 1 tag compare required per access
- ❖ Characterized by exactly one line per set.



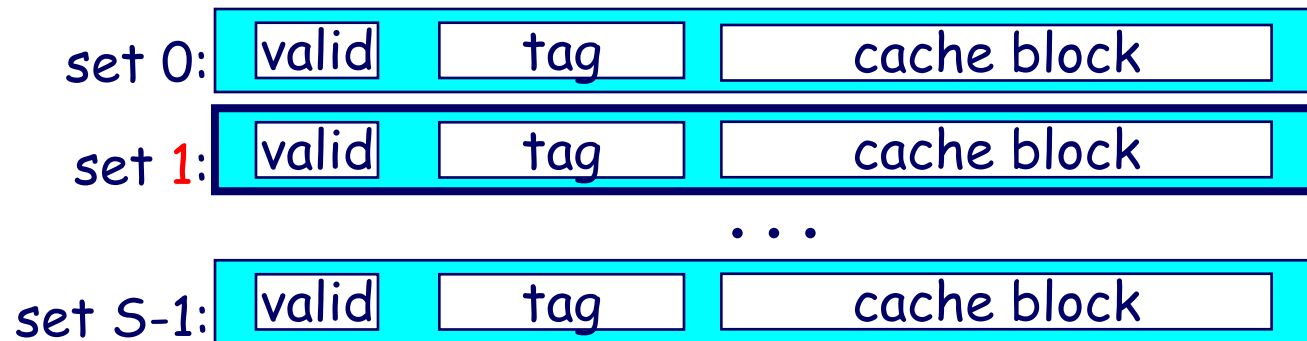
Cache size: $C = B \times S$ data bytes

Accessing Direct-Mapped Caches

- ❖ Set selection is done by the set index bits

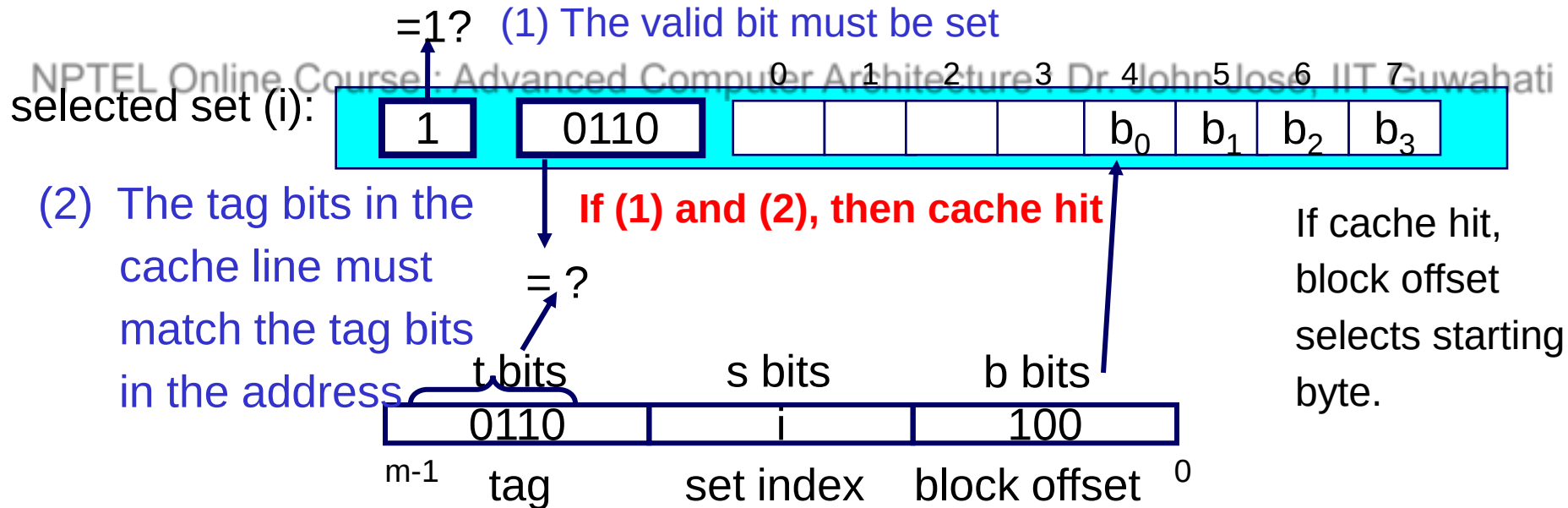


NPTEL Online Course : Advanced Computer Architecture : Dr. John Jose, IIT Guwahati

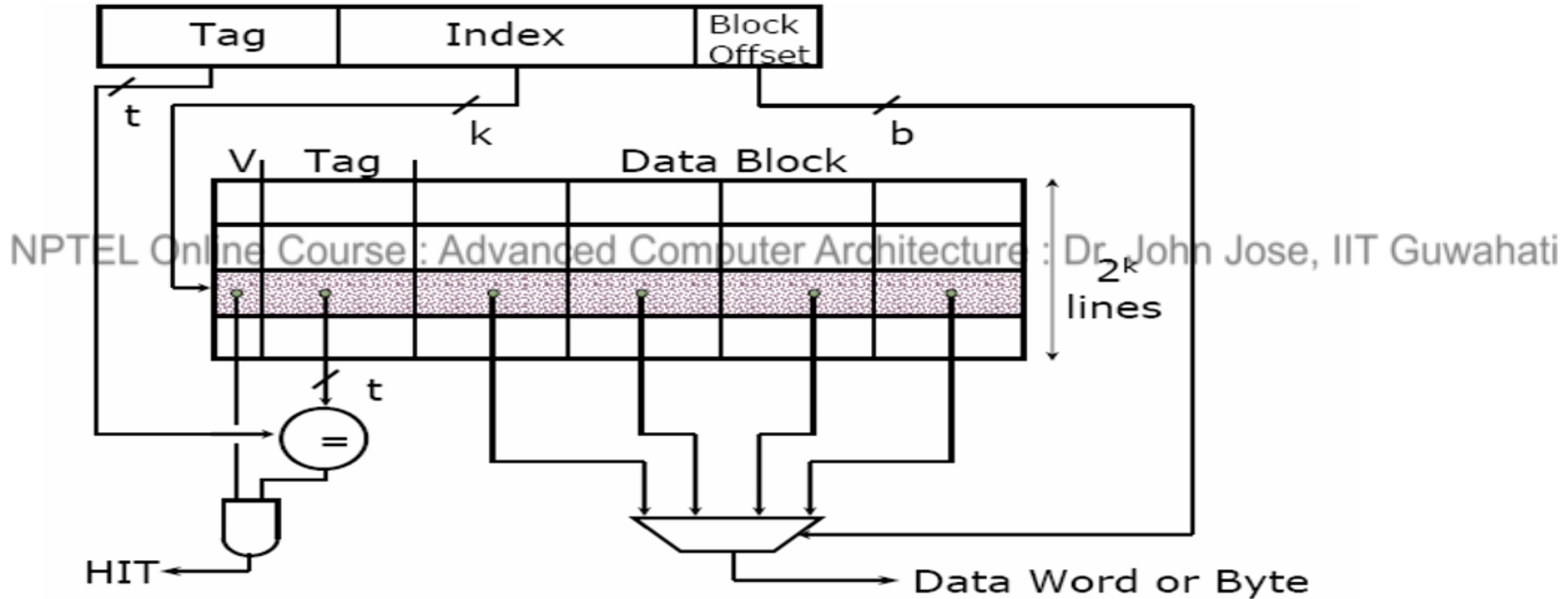


Accessing Direct-Mapped Caches

- ❖ **Block matching**: Find a valid block in the selected set with a matching tag
- ❖ **Word selection**: Then extract the word

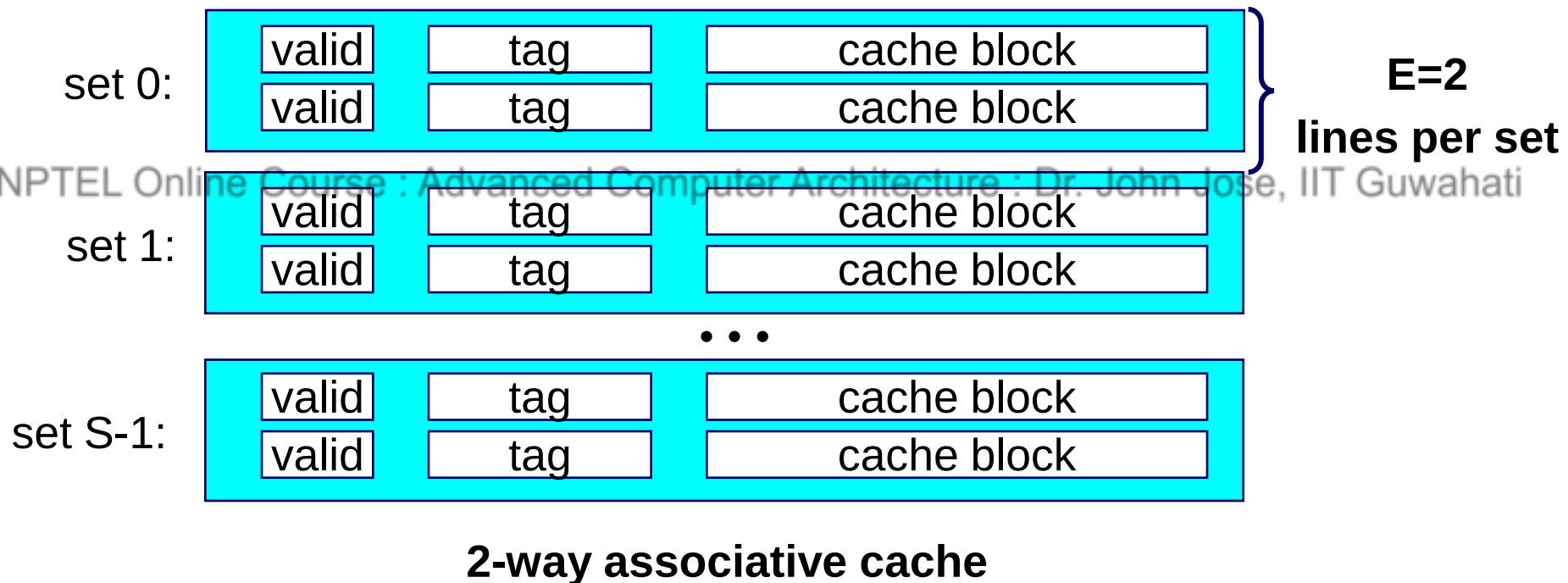


Block Identification – Direct mapped



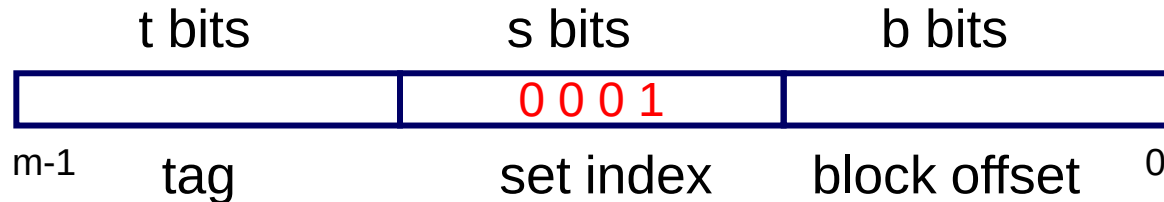
Set Associative Cache

- ❖ Characterized by more than one line per set

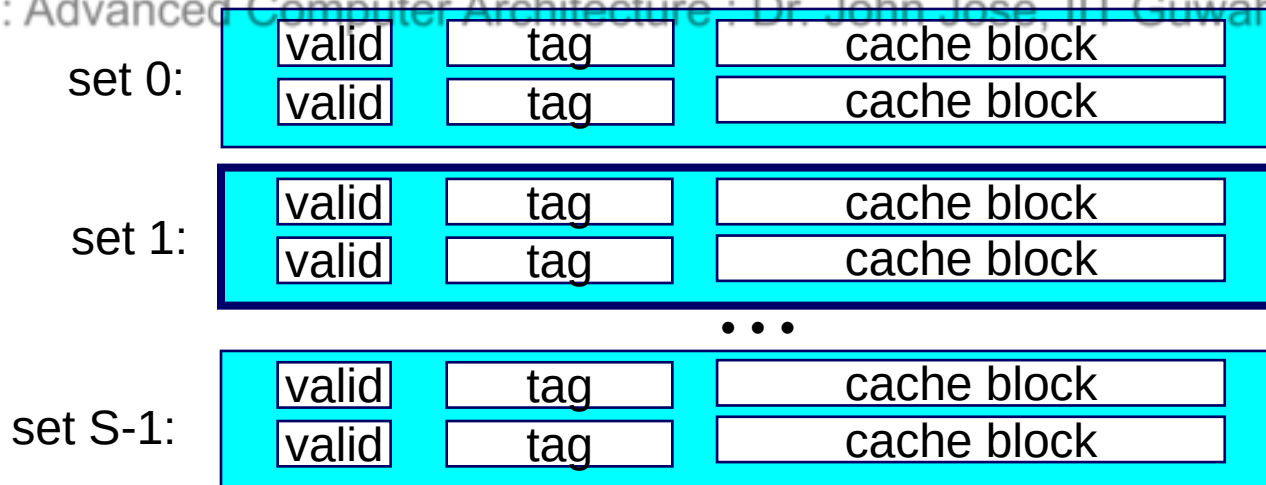


Accessing Set Associative Caches

- ❖ Set selection is identical to direct-mapped cache

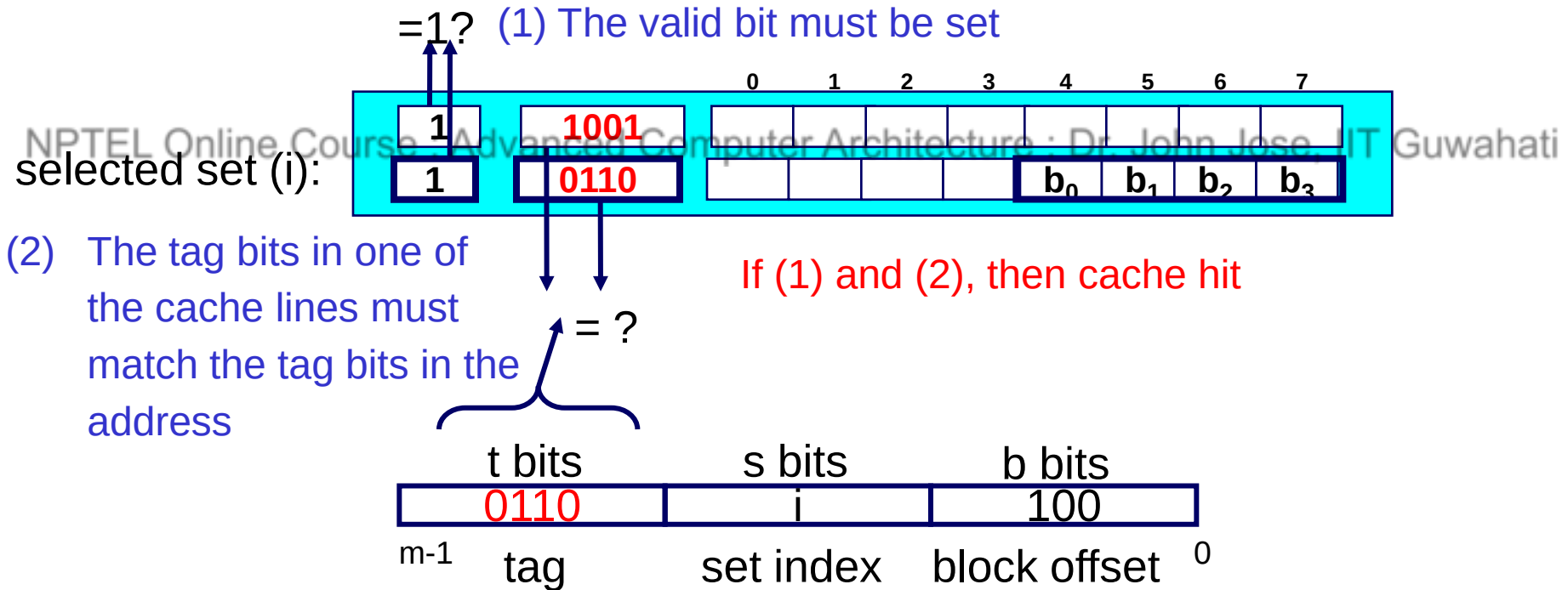


NPTEL Online Course : Advanced Computer Architecture : Dr. John Jose, IIT Guwahati



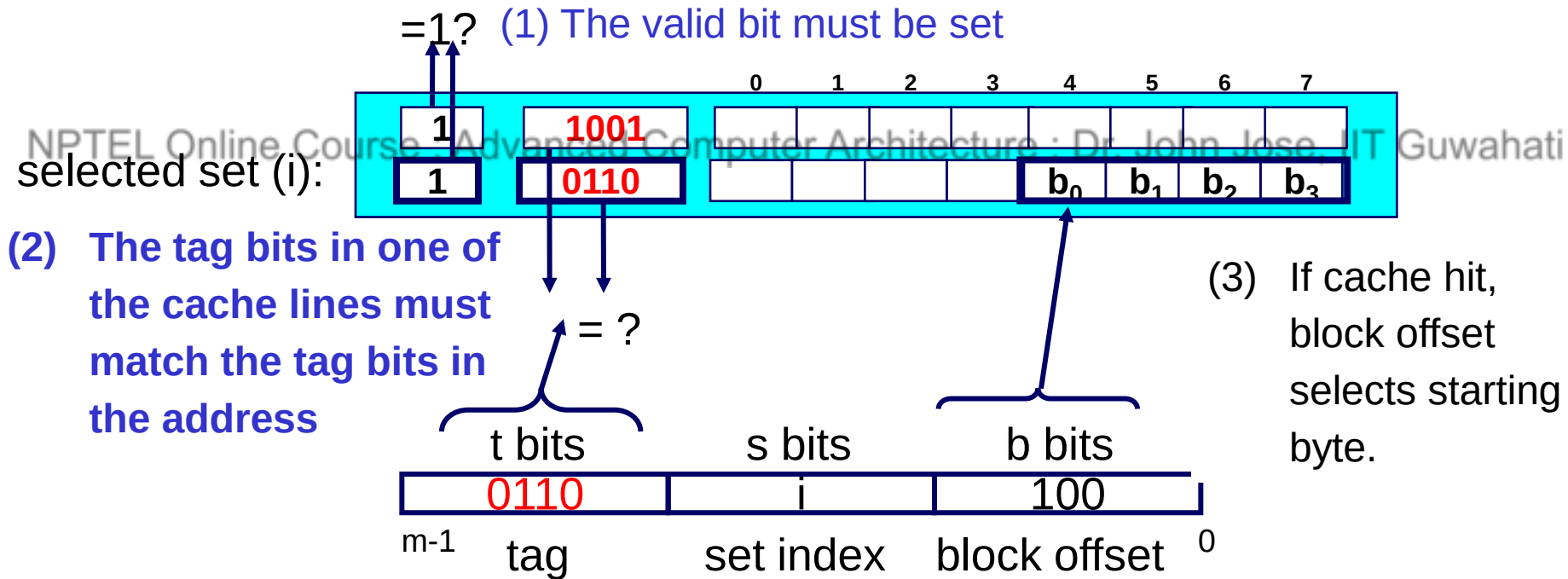
Accessing Set Associative Caches

- ❖ Block matching is done by comparing the tag in each valid line in the selected set.

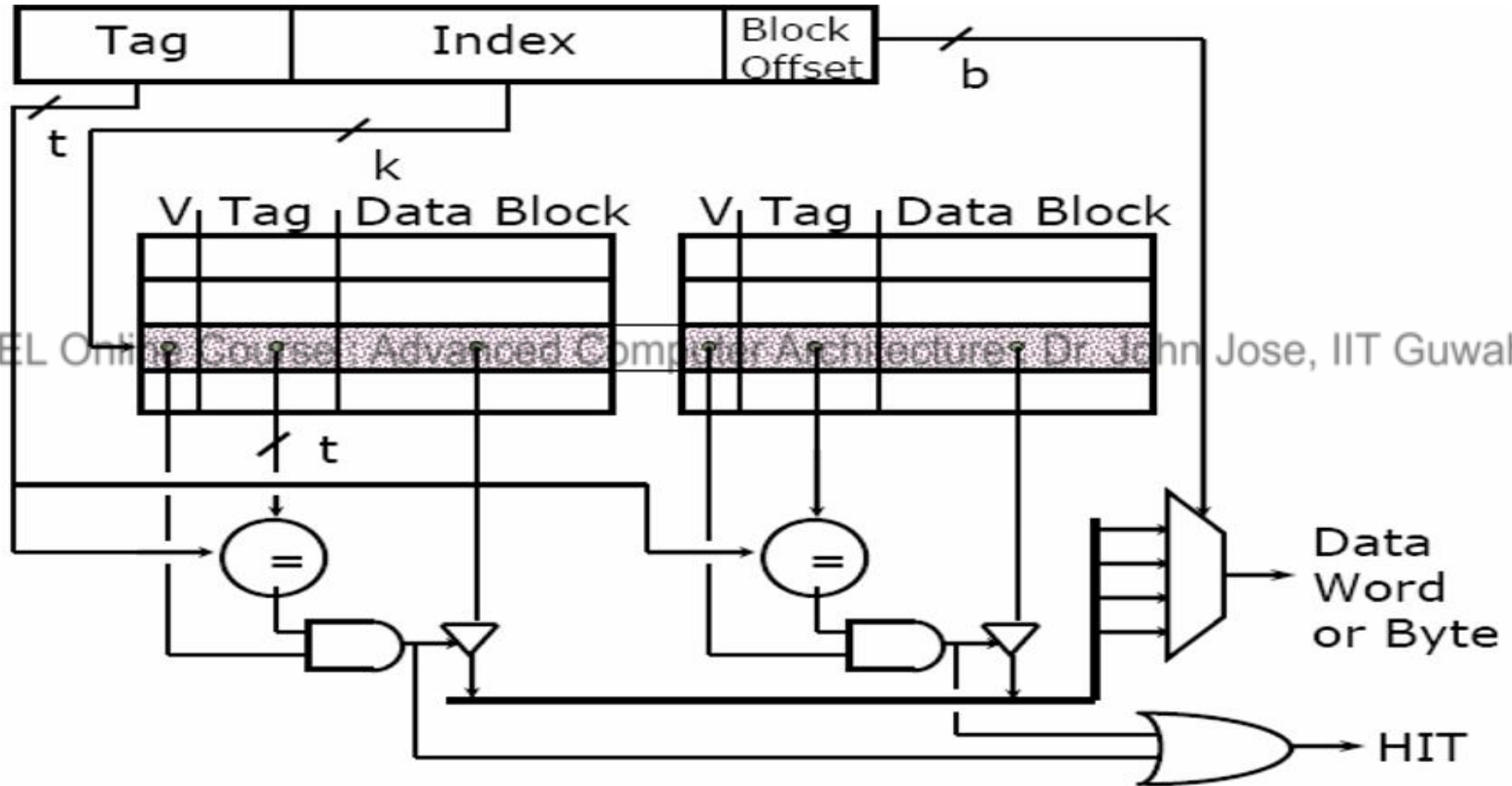


Accessing Set Associative Caches

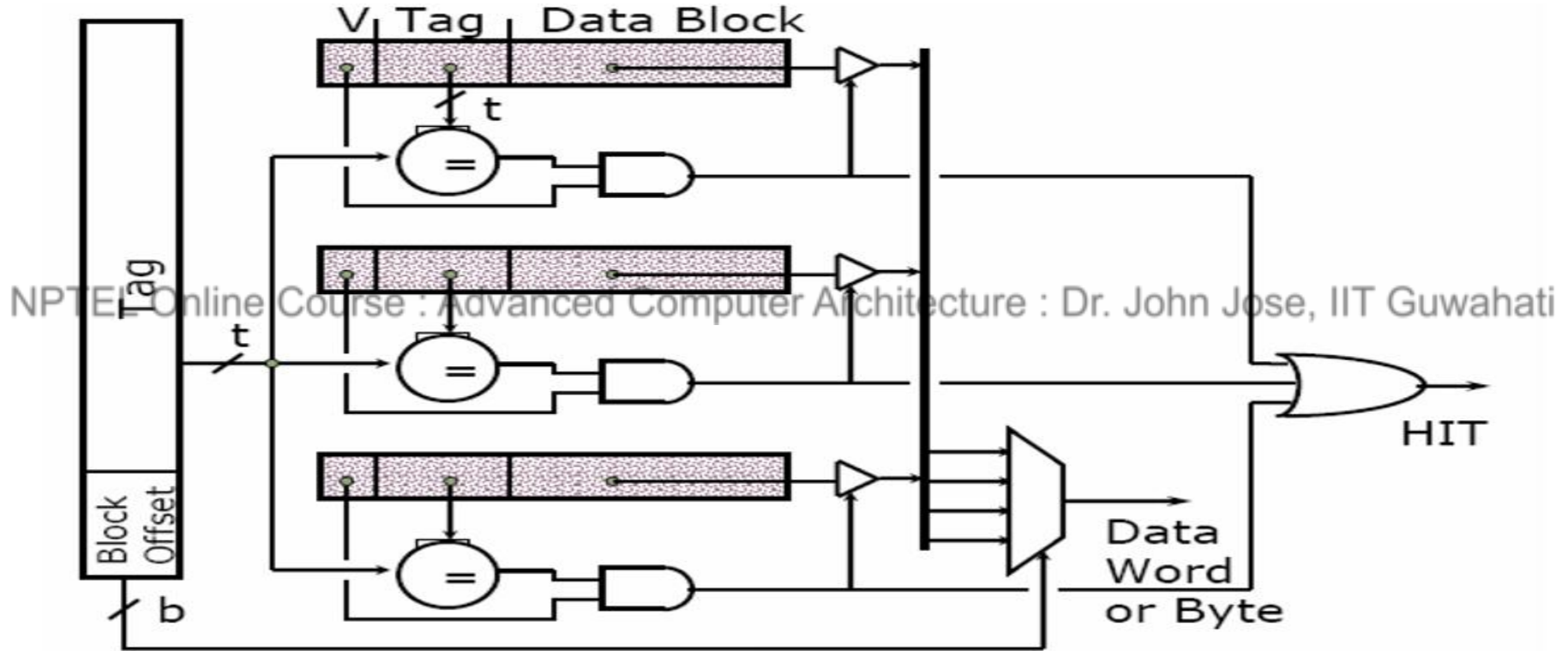
- ❖ Word selection is done same as direct mapped cache but chosen only on the line has produced a hit.



Block Identification – Set Associative



Block Identification – Fully Associative

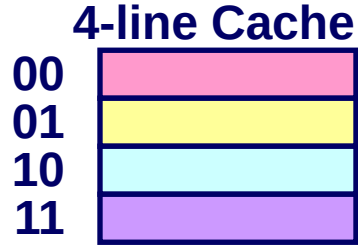


Cache Indexing



- ❖ Decoders are used for indexing
- ❖ Indexing time depends on decoder size ($s: 2^s$)
- ❖ Smaller number of sets, less indexing time.

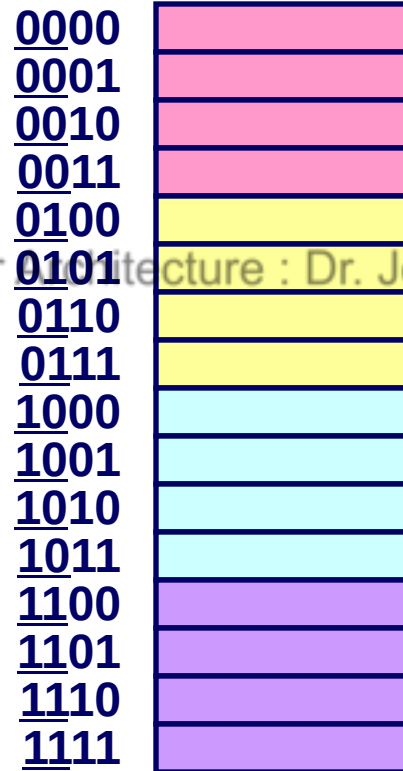
Why Use Middle Bits as Index?



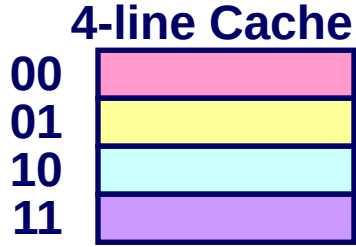
High-Order Bit Indexing

- ❖ Adjacent memory lines would map to same cache entry
- ❖ Poor use of spatial locality

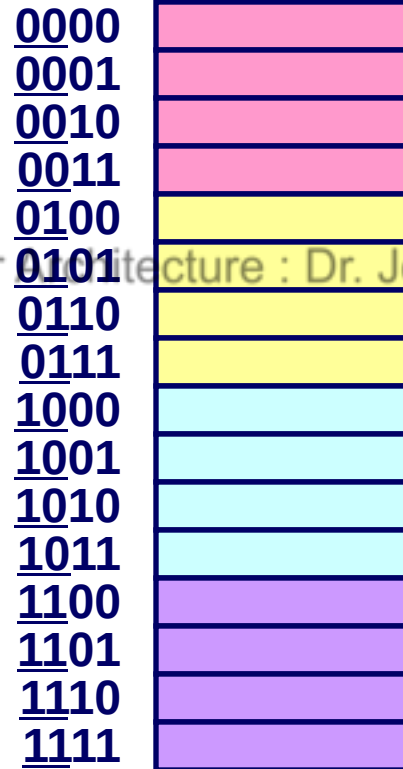
High-Order Bit Indexing



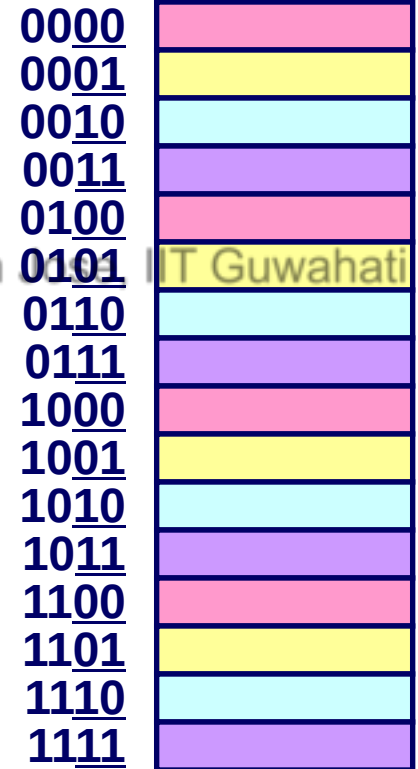
Why Use Middle Bits as Index?



High-Order
Bit Indexing



Middle-Order
Bit Indexing



Middle-Order Bit Indexing

- ❖ Consecutive memory lines map to different cache lines
- ❖ Better use of spacial locality without replacement

**THANK
YOU!**

FOR

YOUR

PRESENCE

NPTEL Online Course : Advanced Computer Architecture : Dr. John Jose, IIT Guwahati

johnjose@iitg.ernet.in
<http://www.iitg.ac.in/johnjose/>