

# CS305

# Computer Architecture

## Cache Design: A Beginning

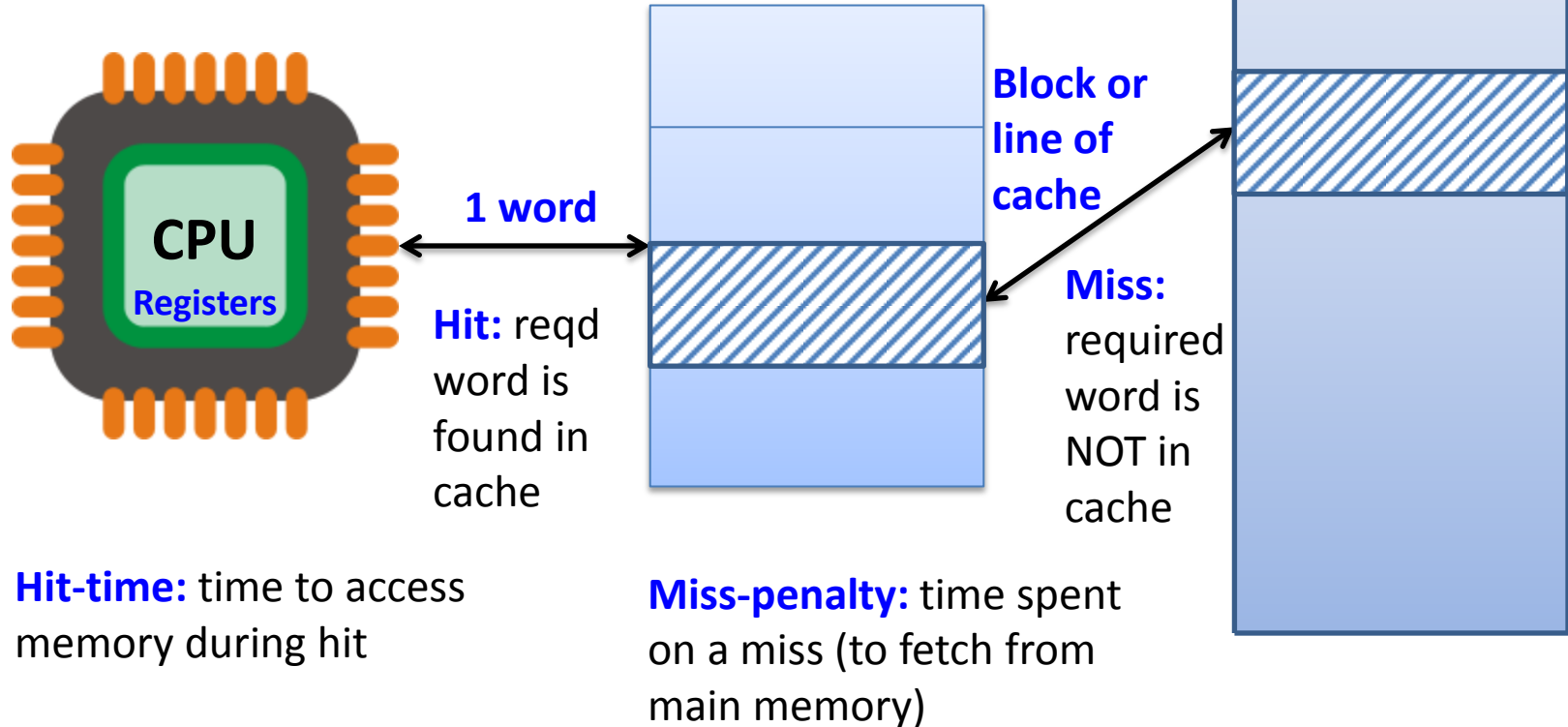
Bhaskaran Raman  
Room 406, KR Building  
Department of CSE, IIT Bombay

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

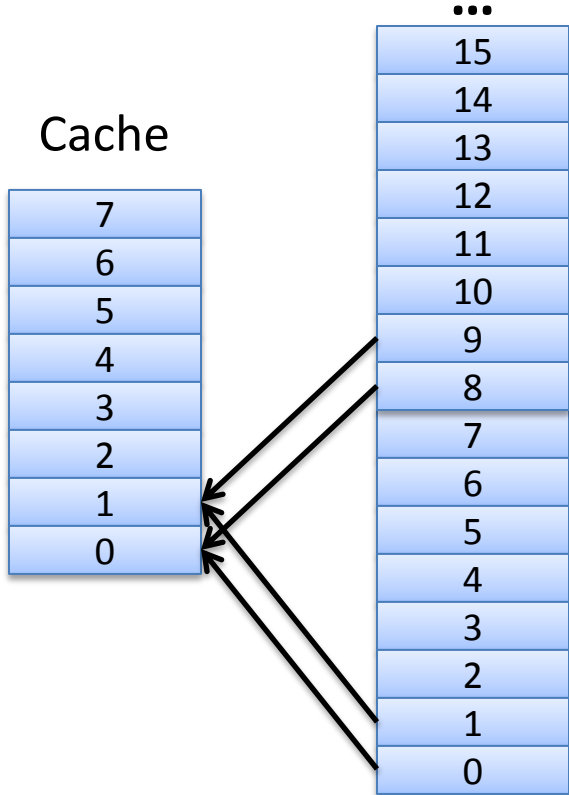
# Terminology

**Hit-ratio/hit-rate:**  $\#hits/\#accesses$

**Miss-ratio/miss-rate:**  $\#misses/\#accesses$  Cache



# Simplest Possible Cache Design: Single Word per Block, Direct-Mapped



Single word per block; block size = 1 word

Given memory word,  
block # in cache =  
(memory block #) MOD (# blocks in cache)

Only one cache location for a given  
memory word → **Direct Mapped**

If # blocks in cache is power of 2,  
MOD == last few bits of memory word addr

# Direct Mapped Cache: A Numeric Example

Memory =  $2^6$  words, Cache =  $2^3$  words

Number of bits for memory address = 8

Number of possible states of a cache block =  $2^3 + 1$  (can be invalid)



$2^3$  possible words in memory can map to the same block in cache

Block # in cache

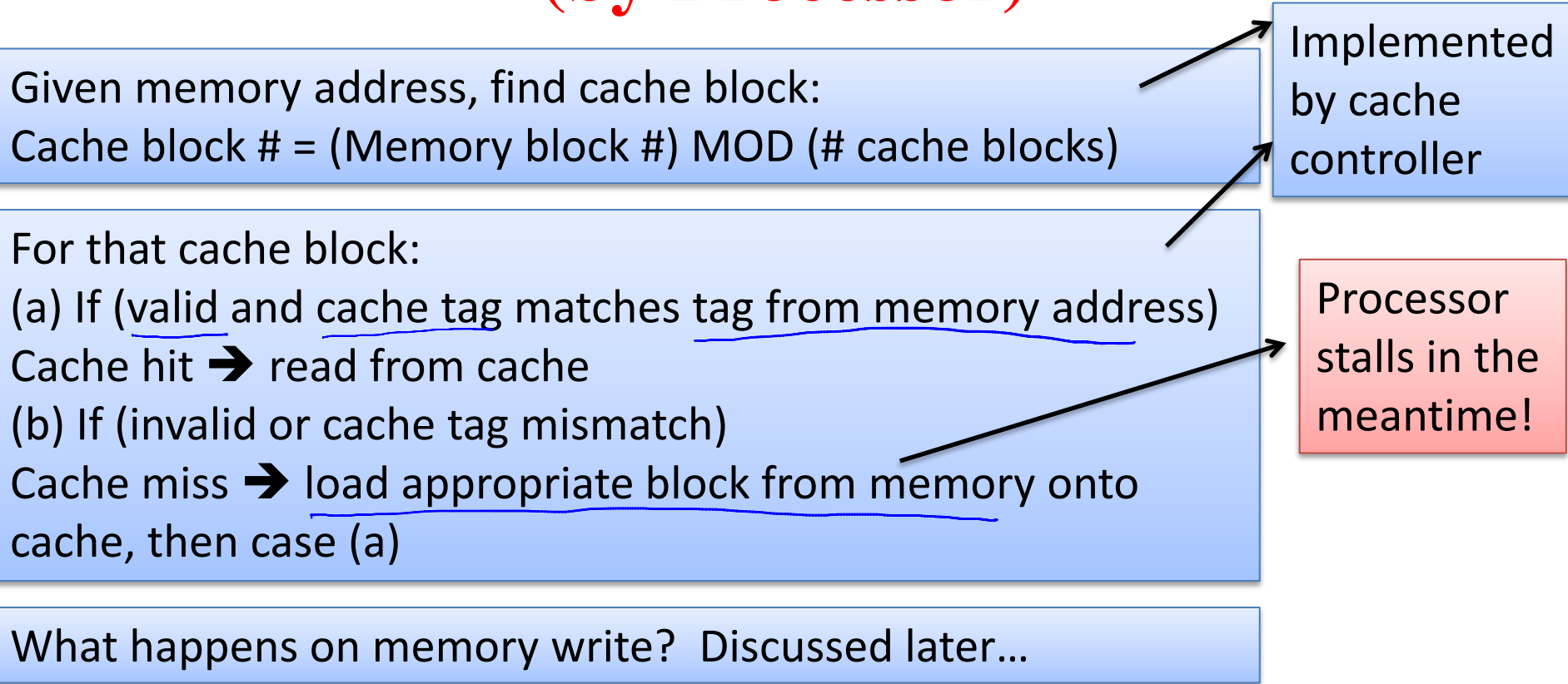
Each cache block, in addition to data (or instruction) has:  
1 valid bit  
3 bits of tag (which memory block is in cache currently)

# What Happens on a Memory Read? (by Processor)

Given memory address, find cache block:

Cache block # = (Memory block #) MOD (# cache blocks)

Implemented  
by cache  
controller



For that cache block:

(a) If (valid and cache tag matches tag from memory address)

Cache hit → read from cache

(b) If (invalid or cache tag mismatch)

Cache miss → load appropriate block from memory onto cache, then case (a)

Processor  
stalls in the  
meantime!

What happens on memory write? Discussed later...

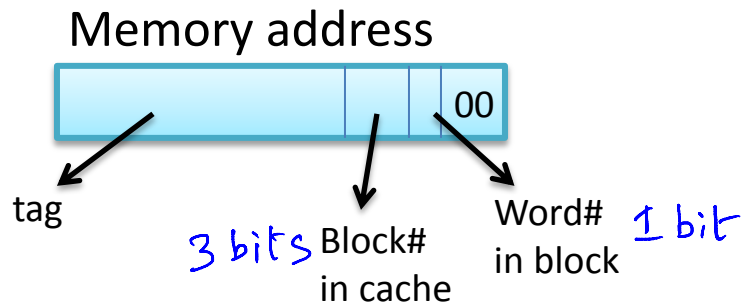
# Direct Mapped Cache with Multiple Words Per Block

Cache

7
6
5
4
3
2
1
0

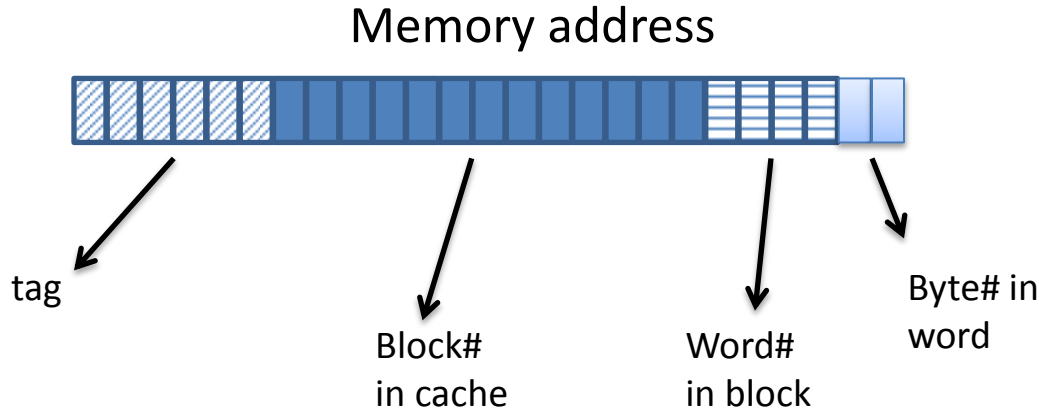
To take advantage of spatial locality: block size  $> 1$ , say block size = 2 words

Recall: block is unit of transfer between cache and memory  $\rightarrow$  on read of any word in a block, entire block is loaded onto cache



# Multiple Words/Block: A Numeric Example

Main memory = 32 MB, Cache = 512 KB,  
Block size = 16 words  
Draw fields of memory address



$$\frac{512 \text{ KB}}{16 \text{ words}} = \frac{2^{19} \text{ bytes}}{2^6 \text{ bytes}} = 2^{13}$$

$$\# \text{ bits in cache} = 512 \text{ KB} + (2^{13} \times (6+1))$$

# Cache in Action: An Example

Cache

7
6
5
4
3
2
1
0

Block size = 2 words

Cache size = 16 words

Main memory word access sequence:

0, 1, 20, 19, 17, 0

Walk through cache state...

20, 21 in cache  
19 in cache  
0, 1 in cache x  
17, 16 in cache x  
0, 1 in cache

0: Miss

1: Hit

(1000) 20: Miss  
(1001) 19: Miss  
(1001) 17: Miss  
0: Miss



# Handling Writes:

## What Happens on Write Hit?

**Write-back:** write (block) to memory “later”  
Need “dirty” bit to know if block needs to be written back, at the time of replacement

**Write-through:** write (word) to memory now  
Can be slow!  
Can have a write-buffer to mask delay  
Sequence of writes close to one another →  
delay will surface

# Write Hit: Some Remarks on Performance

In write-back, even write hit can take two cycles!  
One to read the tag, one to do the actual write  
Can use **store-buffer** to mask latency

**Note:** can have write-buffer for “write-back” too,  
to reduce miss penalty

**Beware:** on read-miss, be sure to check store-  
buffer and write-buffer as well!

# Handling Writes:

## What Happens on Write Miss?

**Write-allocate:** fetch block from memory

**Write-no-allocate:** do not fetch block from memory

# Handling Writes: The Four Design Choices

	Write-back	Write-through
Allocate		<i>Possible, makes less sense</i>
No-Allocate	<i>Possible, makes less sense</i>	Write-around

# Summary

- Direct mapped: many-to-one mapping
- Memory address =  
**tag::block#incache::word#inblock::00**
- Cache has: block (data), tag, valid bit
- Write policy: four design possibilities