CS305
Computer Architecture

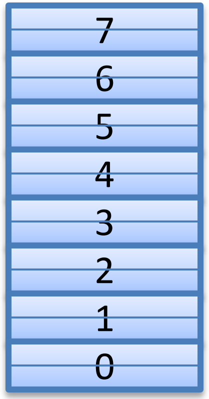
Associative Caches

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

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

Recall Direct Mapped Cache Example

Cache



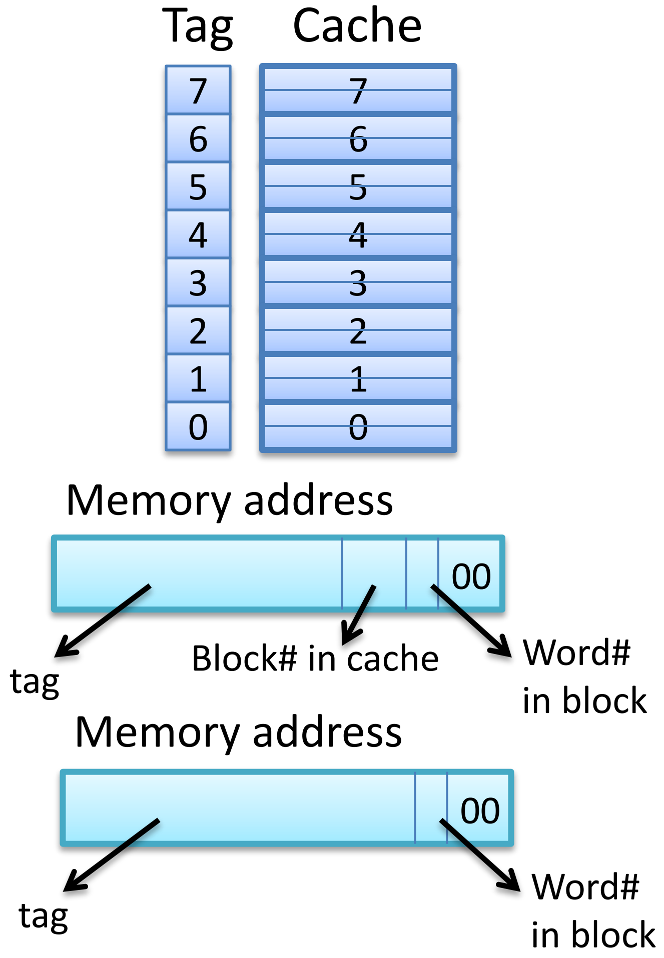
Block size = 2 words
Cache size = 16 words
Main memory word access sequence:
0, 1, 20, 19, 17, 0

17 evicts 0&1 even though there is plenty of empty space in the cache! **CONFLICT** miss
This causes 0 to miss next.

Suppose 17 & 0 accessed in a loop: hit-ratio=0 !

Main culprit:
Many-to-one mapping of Direct Mapped cache

Associative Cache: The Idea



A block of main memory can be in ANY cache block: **fully associative** cache

Main memory block # → Cache block #
Now a many-to-many mapping!

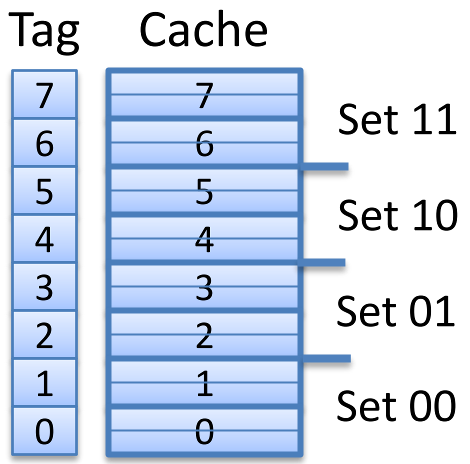
Price to pay: search for block in cache, given memory address ➔ parallel comparator

(+) Reduced conflict misses
(–) Comparator cost
(–) Increased hit time!

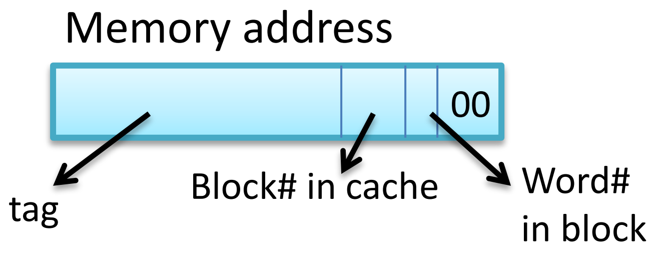
Set Associative Cache: The Idea

Direct Mapped	A block of main memory can be in exactly **ONE** cache block
Fully Associative	A block of main memory can be in **ANY** cache block
Set Associative	A block of main memory can be in **ANY of a SET** of cache blocks

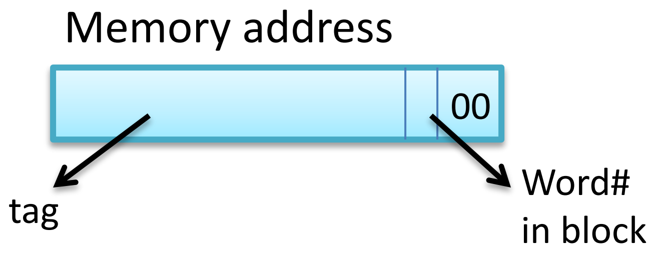
How a Set Associative Cache Works



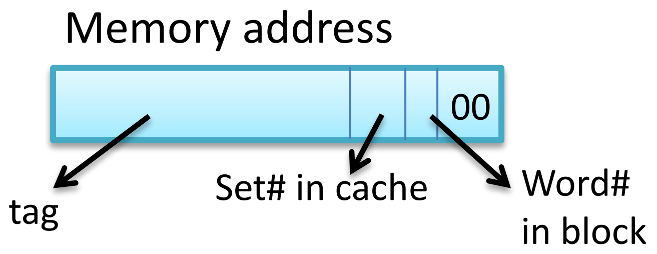
Direct Mapped



Fully Associative



Set Associative



Given memory address, find set#

#parallel comparators = #blocks in set

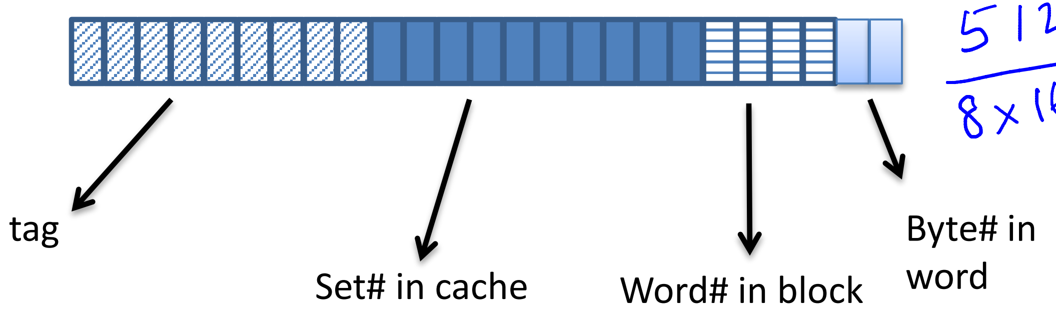
The Design Space Continuum

Direct Mapped	Set associative cache with set size = 1 block Called 1-way set associative
Fully Associative	Set associative cache with set size = K K = (# blocks in cache), K-way set associative
Set Associative	Set associative cache with set size S: 1 < S < K S-way set associative

Set Associative Cache: A Numeric Example

Main memory = 32MB, Cache size = 512 KB, Block size = 16 words
Show main memory address fields for 8-way set associative cache

Memory address

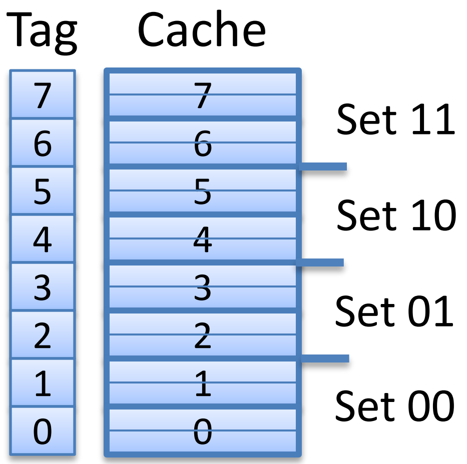


$$\frac{512\,KB}{8 \times 16 \times 4 \text{ bytes}} = 2^{10} \text{ sets}$$

Number of comparators required = 8

Number of bits to be compared in each comparator = tag size = 9 bits

Replacement Policy



What to do when set is full? Which cache block to replace?

Not relevant for direct mapped cache: no choice

Some possibilities:

a) Random
b) First-In-First-Out (FIFO)
c) Least frequently used (LFU)
d) Least recently used (LRU)

LRU found most effective in practice: difficult to implement in hardware
What is implemented: **clock algorithm**, an approximation of LRU, works well

More comprehensive treatment of replacement algorithms: in OS course

Summary

- Associativity: helps reduce cache conflict misses
- Fully associative: too expensive (cost, hit time)
- Set associative: get benefits of fully associative and direct mapped
- Block replacement policy: LRU
- Next: facets of cache performance