CACHING AND PAGE REPLACEMENT

Zhang Shiqi, Zeng Xinxun, Huang Bo

OVERVIEW

- **Caching Strategy**
  - Direct-mapped cache
  - Set-associative cache
  - Full-associative cache
- **Paging Replacement**
  - FIFO
  - MIN
  - LRU
  - Clock
  - Second-chance list

WHAT IS CACHE

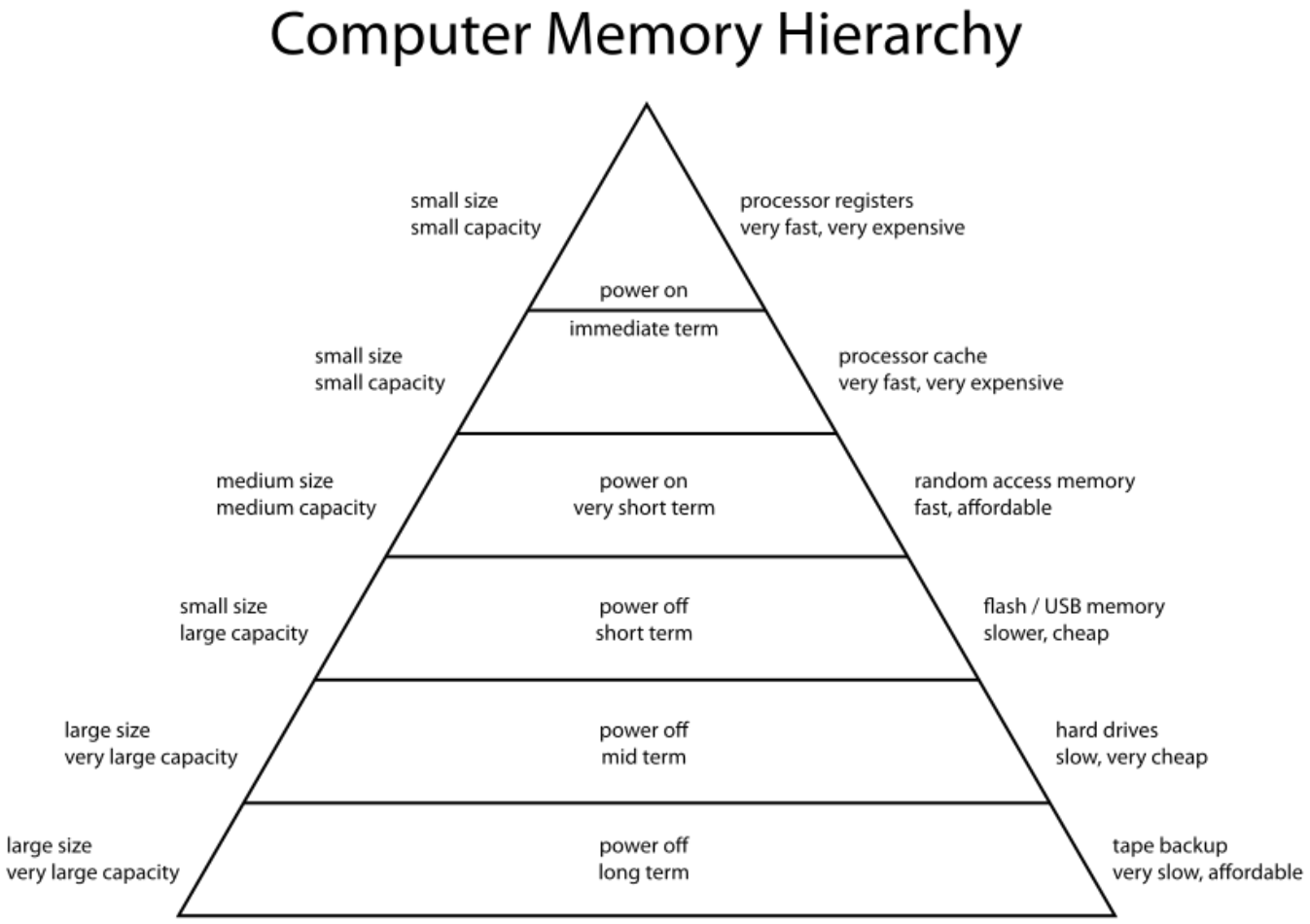
A repository for copies that can be accessed more quickly than the original.

Make frequent case fast and infrequent case less dominant.

Why is caching useful?

- **Temporal locality**
- **Spatial Locality**

CACHING



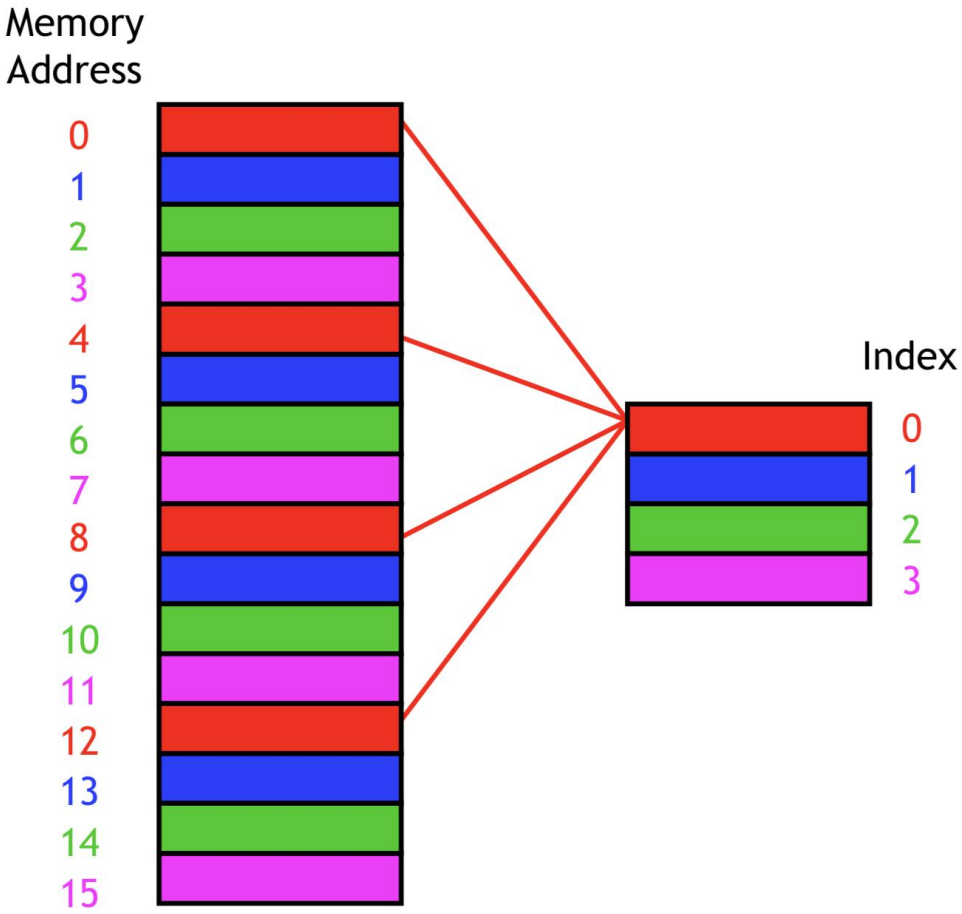
Pictures from wiki

DIRECT MAPPED CACHE

A *direct-mapped* cache is the simplest approach:

- **each main memory address maps to exactly one cache block.**

How to compute this mapping?

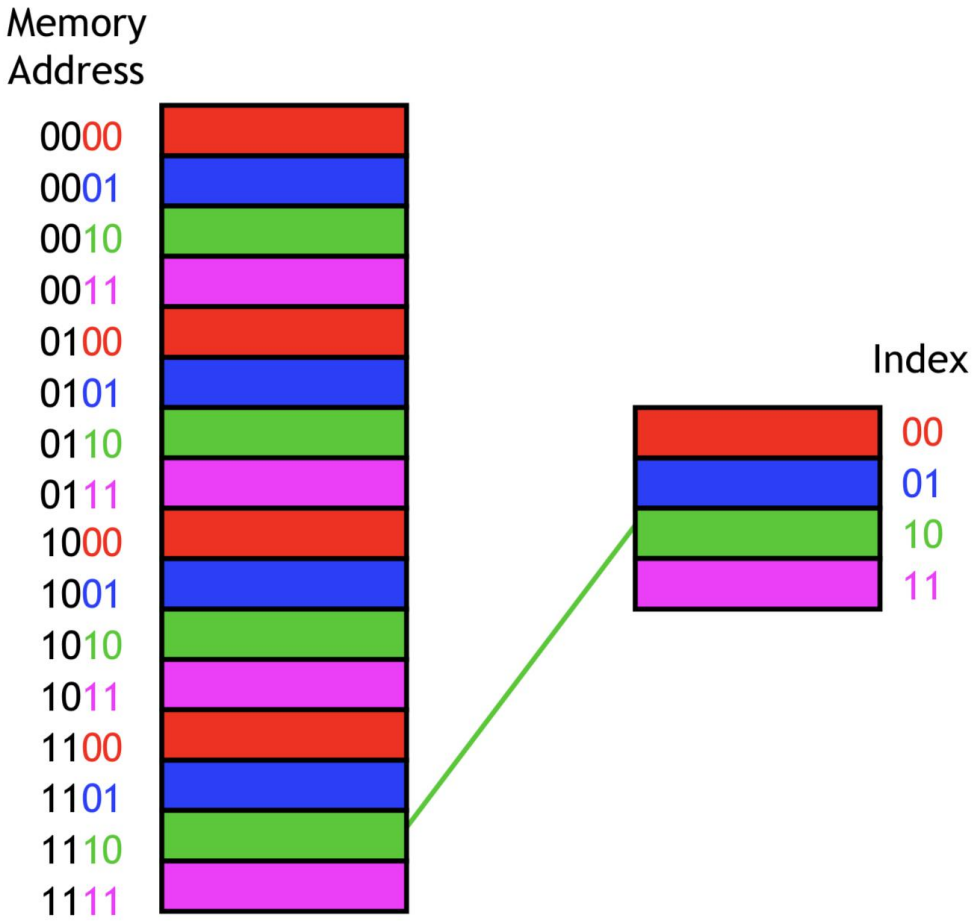


https://courses.cs.washington.edu/courses/cse378/09wi/lectures/lec15.pdf

MAPPING

Use *k least-significant bits* to mapping

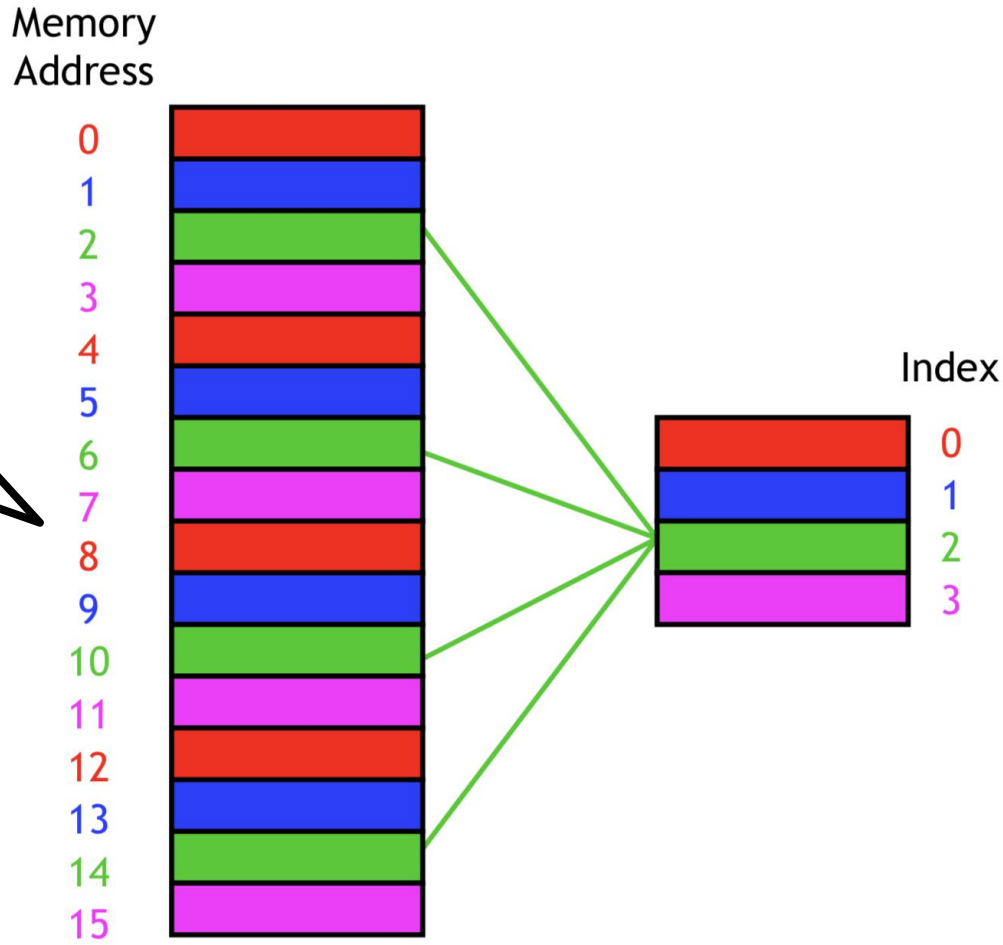
- **k: determined by *cache size***



https://courses.cs.washington.edu/courses/cse378/09wi/lectures/lec15.pdf

QUESTION

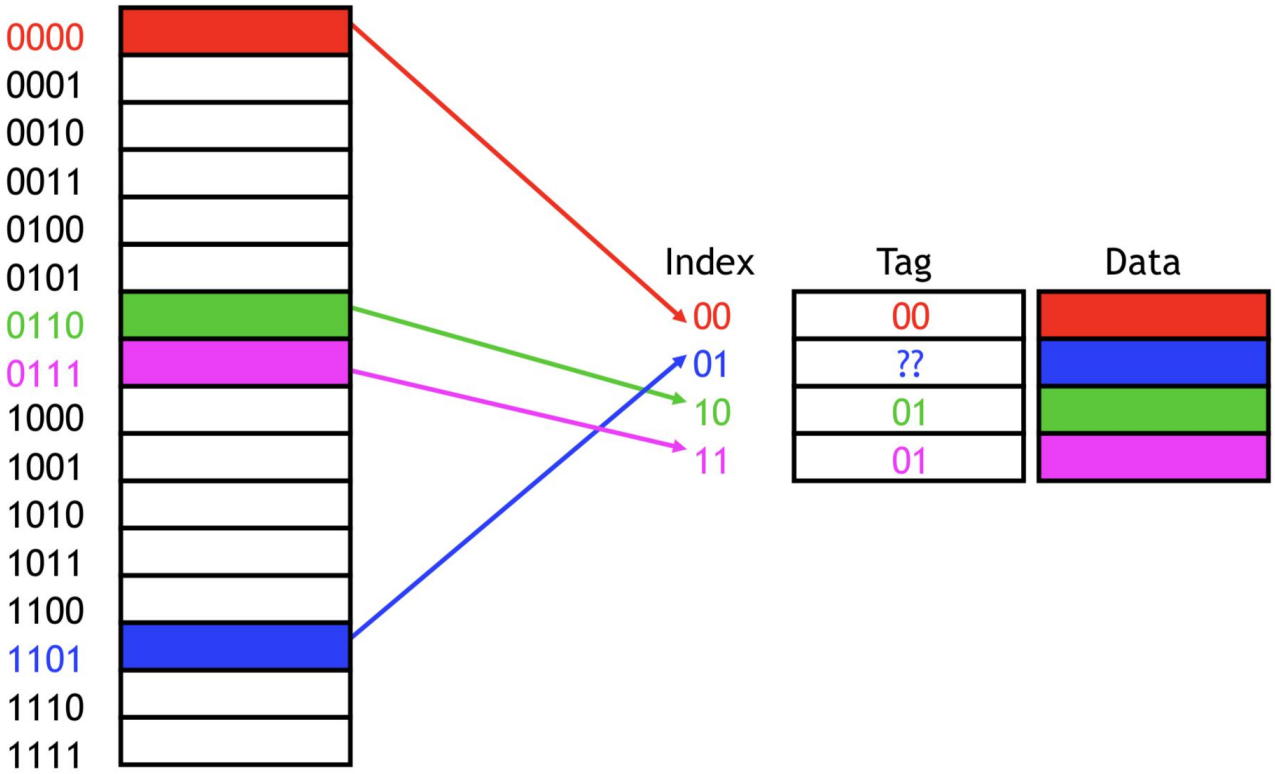
How to DISTINGUISH the correct data?



https://courses.cs.washington.edu/courses/cse378/09wi/lectures/lec15.pdf

TAG FIELD

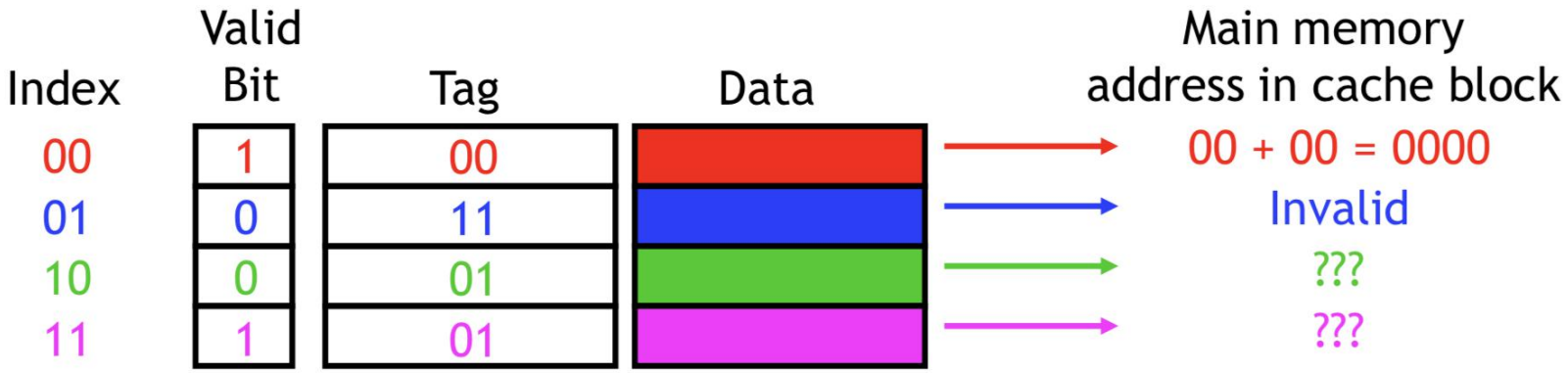
Use register to distinguish between different memory locations



https://courses.cs.washington.edu/courses/cse378/09wi/lectures/lec15.pdf

VALID BIT

- **When the system is initialized, all the valid bits are set to 0**
- **When data is loaded into a particular cache block, the corresponding valid bit is set to 1.**



QUESTION

For a 32-bit system, 9 least significant bits are used for cache index and other bits are used for tag field. Each cache data contains one byte.

What is the total size of cache?

QUESTION

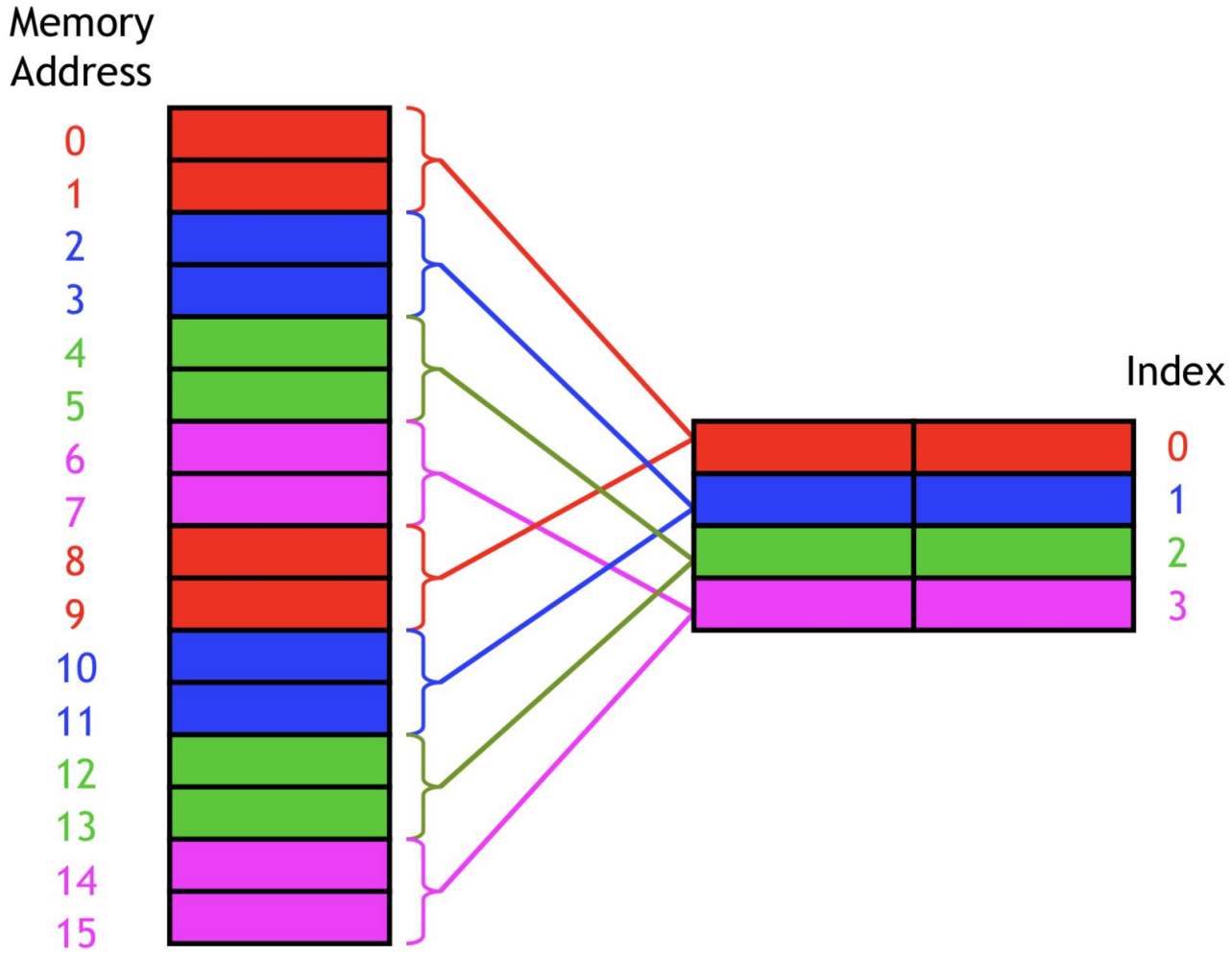
Currently, each cache block only contains one byte from one memory location. What is the problem?

We don’t consider Spatial Locality

How to solve it?

Enlarge block size that can contains data from continuous memory location

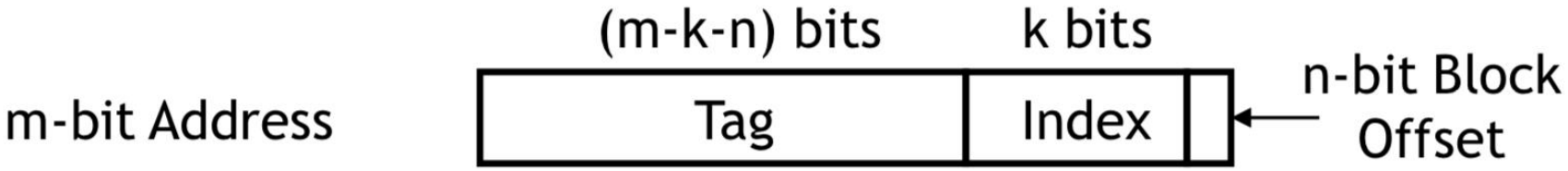
ENLARGE BLOCK SIZE



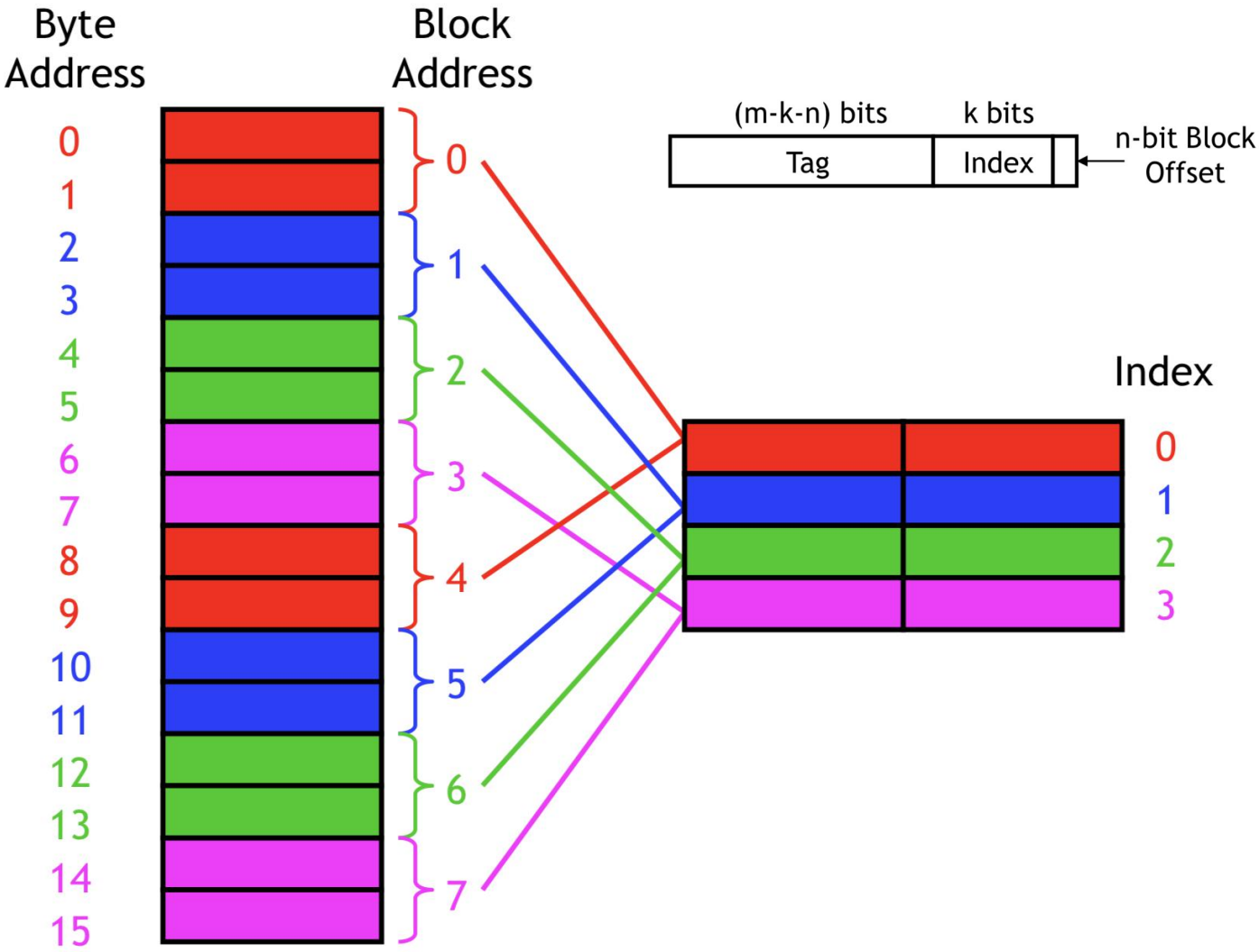
https://courses.cs.washington.edu/courses/cse378/09wi/lectures/lec15.pdf

DIRECT MAPPED CACHE

- **For a m-bit system**
  - Each cache line contain 2^n bytes.
  - Using k bits to identify each cache line.
  - Tag field is the most (m-k-n) bits



DIRECT MAPPED CACHE



https://courses.cs.washington.edu/courses/cse378/09wi/lectures/lec15.pdf

DISADVANTAGE

- All address with the same least significant k bits will mapped to same place
- Eg. 40004**03X**, 50328**03X**
- May cause a lot of conflict

SET ASSOCIATIVE CACHE

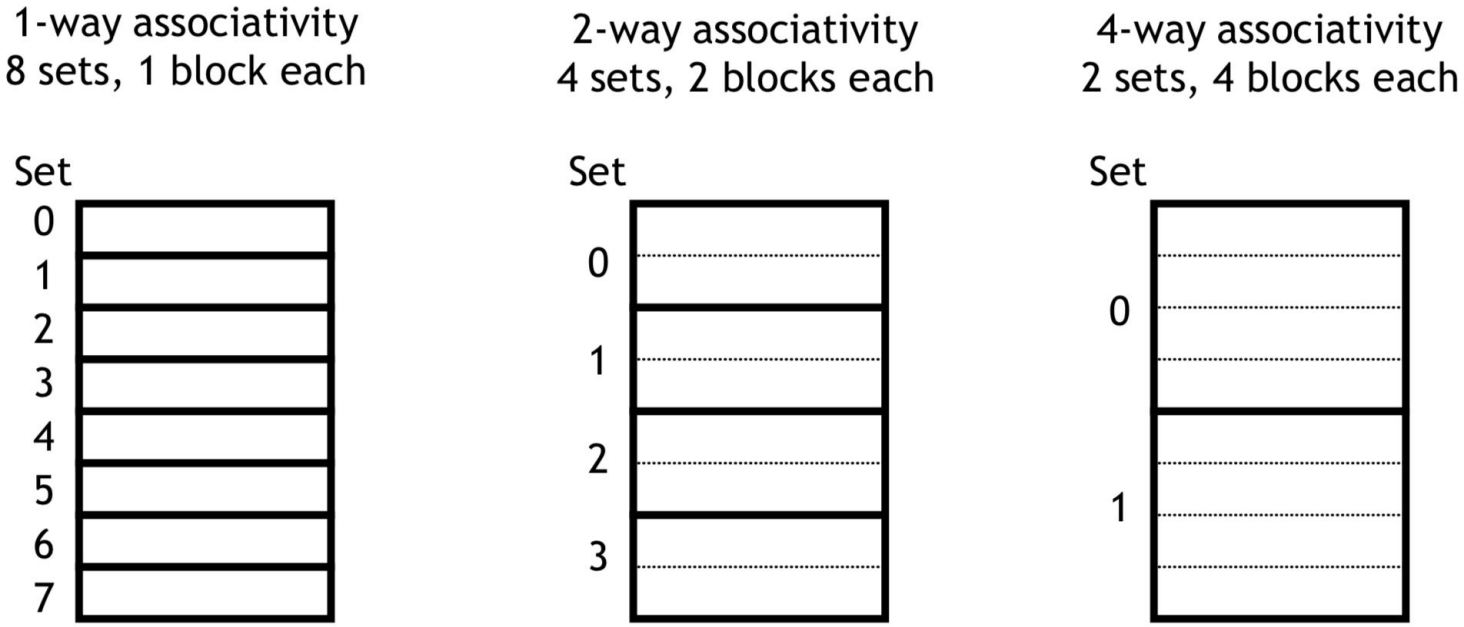
How to avoid conflict?

We store many of them!

- **We have N-direct mapped caches(N-way)**

SET ASSOCIATIVE CACHE

- **The cache is divided into groups of blocks, called *sets*.**
- **If each set has n blocks, the cache is an n-way associative cache**



https://courses.cs.washington.edu/courses/cse378/09wi/lectures/lec15.pdf

SET ASSOCIATIVE CACHE

Pros

- **Improve the spatial utilizations**
- **Reduce miss rate**

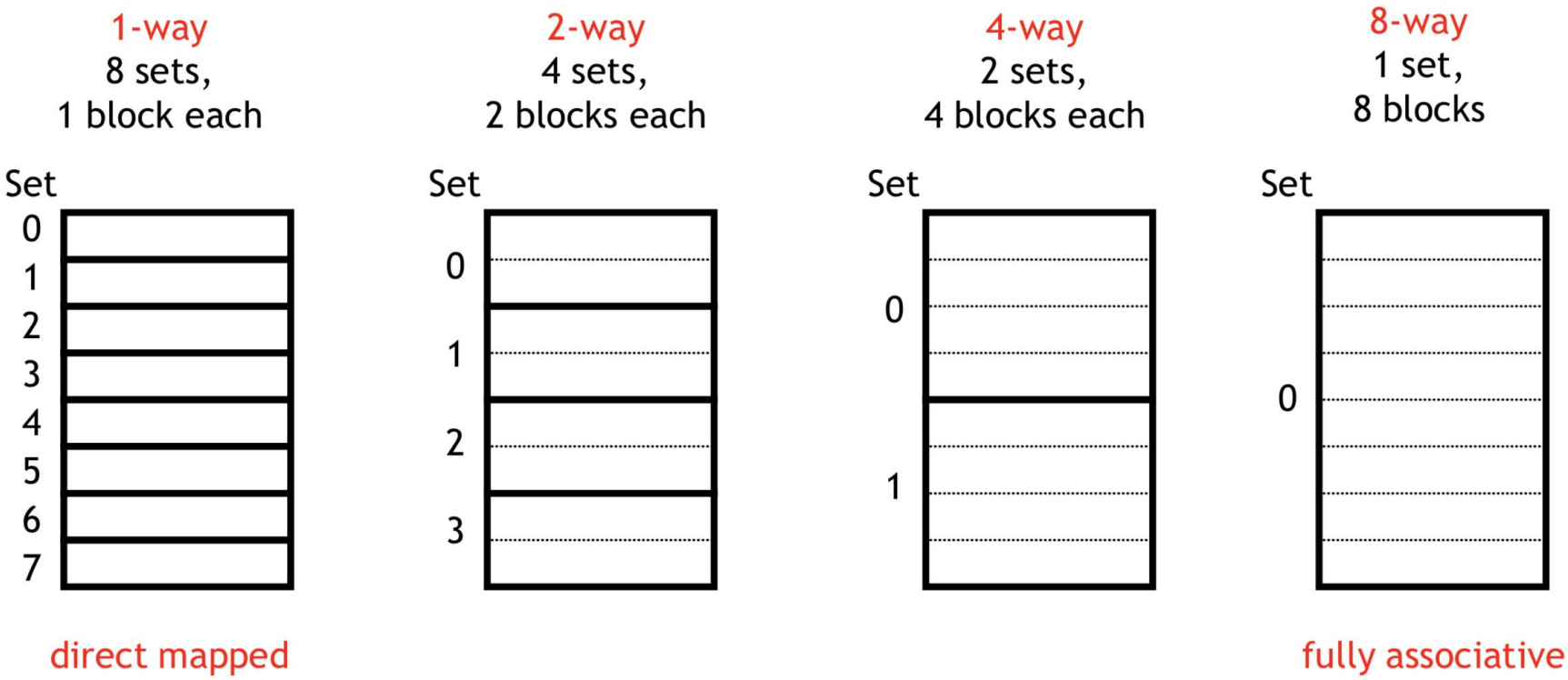
Cons

- **Traverse every way in a set, lower speed**
- **More complicate hardware support**

FULLY ASSOCIATIVE CACHE

- We don’t use cache index. We can use cache tag to distinguish them.
- Only one set in set associative cache

MORE GENERAL VIEW



https://courses.cs.washington.edu/courses/cse378/09wi/lectures/lec15.pdf

QUESTION

How to do replacement in a set?

Can we use FIFO, LRU?

PAGE REPLACEMENT

Why we need page replacement?

To support demand paging

Why we need demand paging?

- Increase multiprogramming degree
- virtual memory

PAGE REPLACEMENT

Random

FIFO

Min

LRU

Clock

Second-Chance List

FIFO

Using a fixed-size queue to maintain the page.

When there comes a new query:

1) If it exists, return the result. (Hit)
2) If it doesn't exist. (Miss)
    1. if the queue is full, pop
    2. push back

MIN

We replace the one won't be used for longest time.

Min gives us the minimum number of faults.

When there comes a new query:

1) If it exists, return the result. (Hit)
2) If it doesn't exist. (Miss)
    1. if the ? is full, find the one
    2. replace it

MIN

Q. Can you prove that Min algorithm gives the minimum number of faults?

LRU

Replace the least recent used one.

When there comes a new query:

1) If it exists(Hit)
    1. update the data structure
    2. return the result.
2) If it doesn't exist. (Miss)
    1. if the ? is full, find the one
    2. replace it
    3. update the data structure

LRU

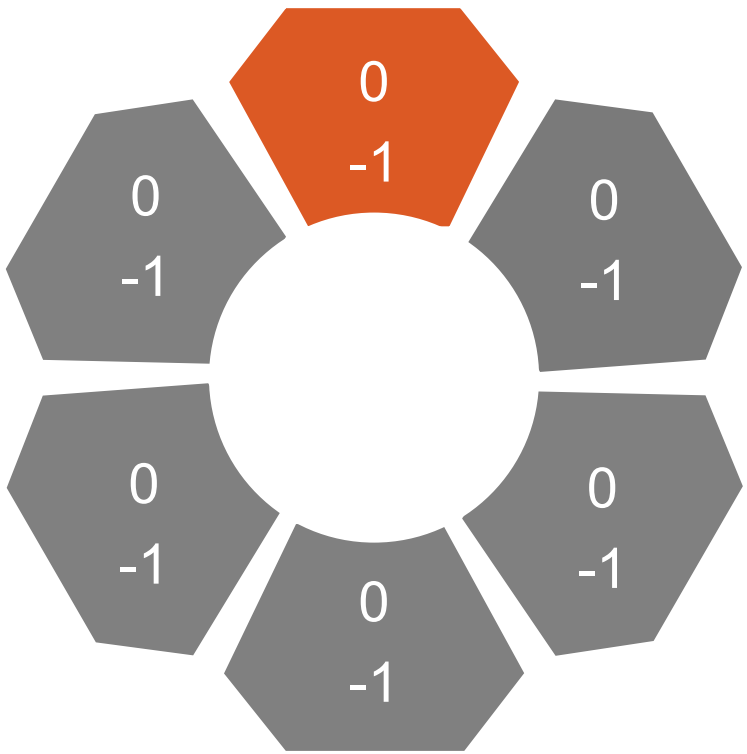
Q: How to implement?

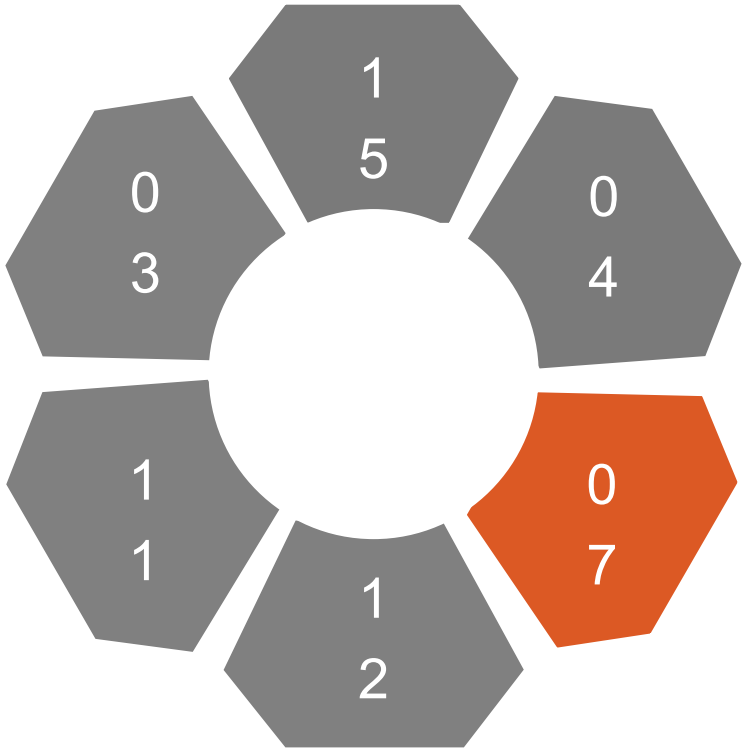
- Counter
- Timestamp
- Double-Linked-List

Q: What's the problem?

Additional operation for every hit, 10 times slower.

CLOCK





SECOND-CHANCE