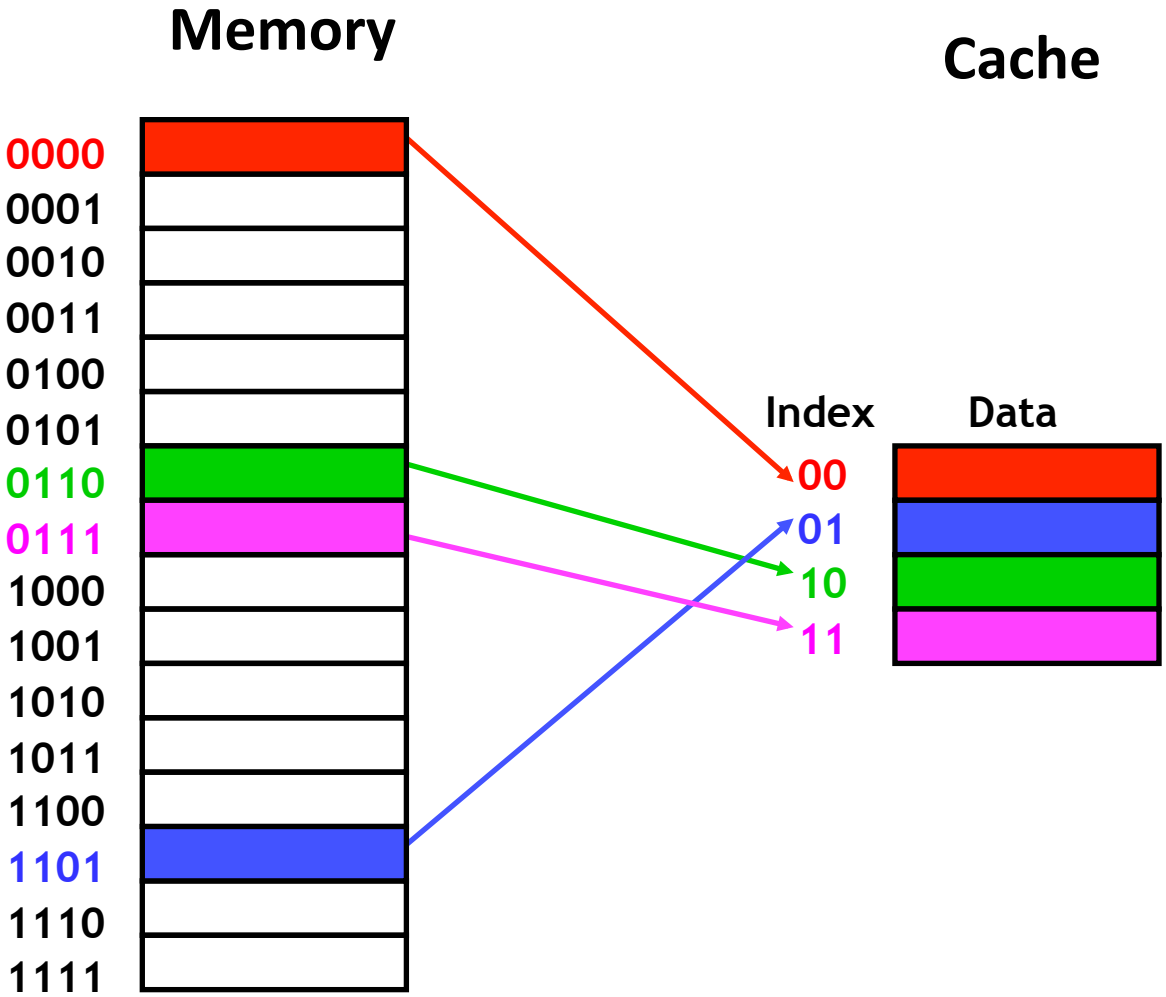
Section 7: Memory and Caches

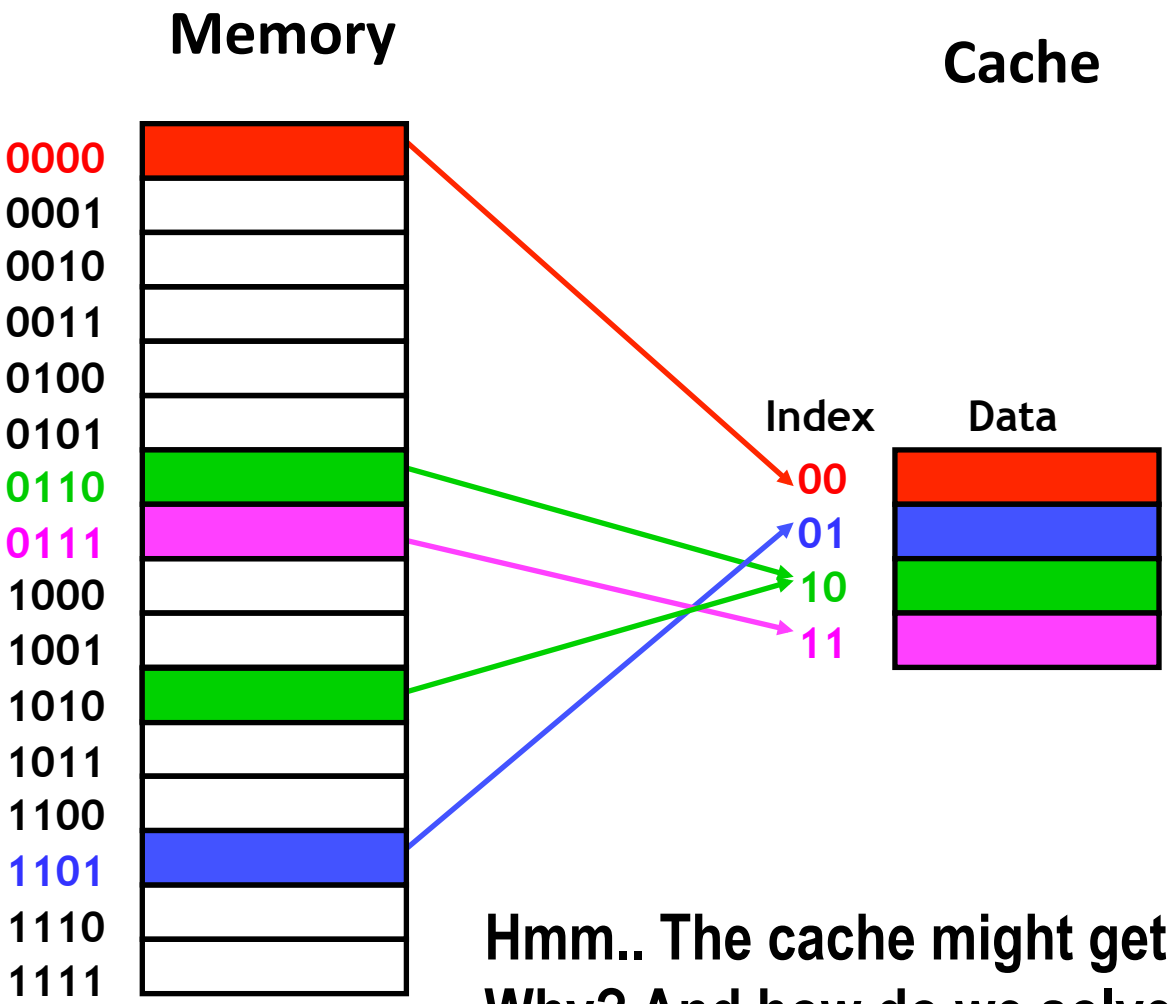
- ~~Cache basics~~
- ~~Principle of locality~~
- ~~Memory hierarchies~~
- Cache organization
- Program optimizations that consider caches

Where should we put data in the cache?



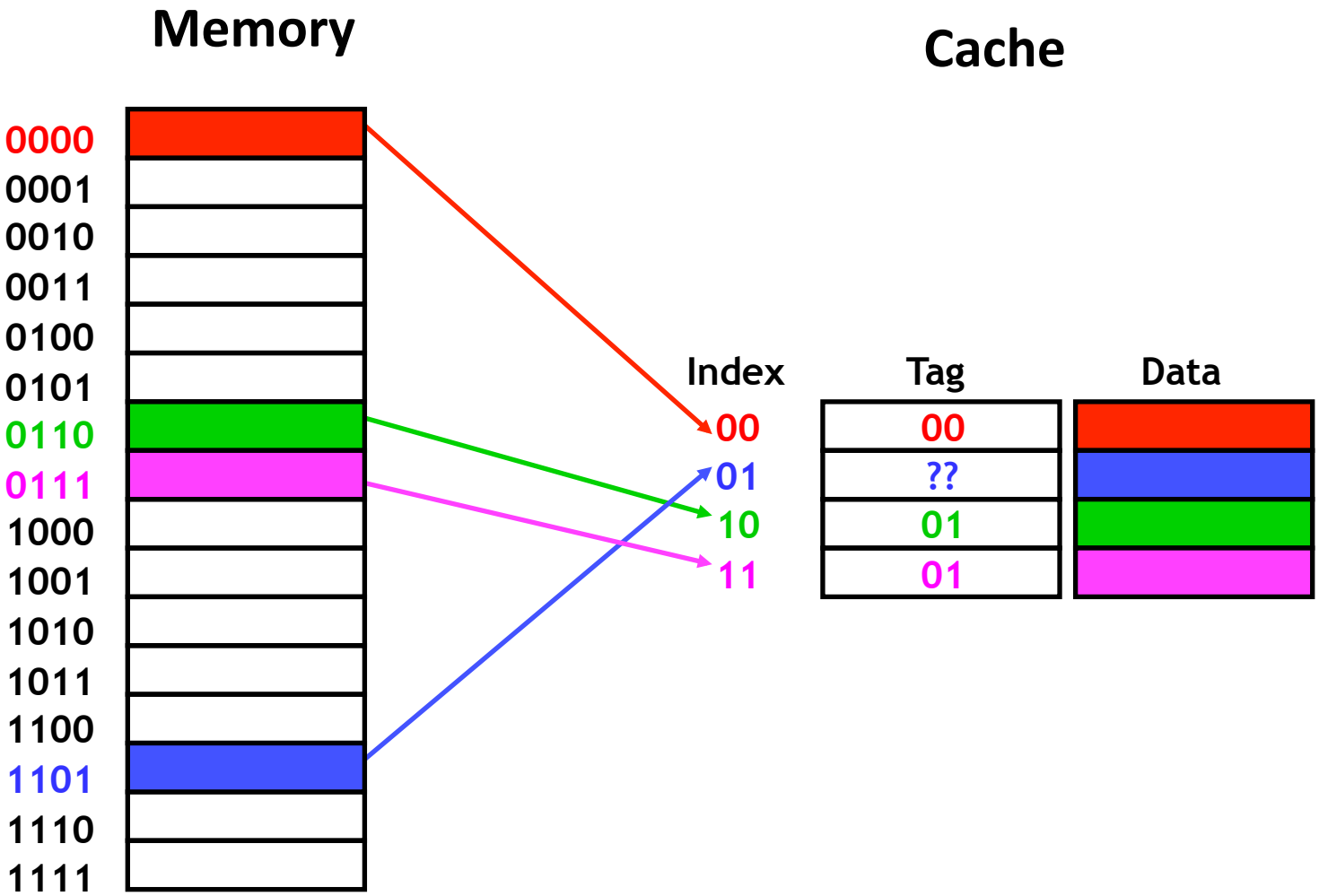
- How can we compute this mapping?

Where should we put data in the cache?

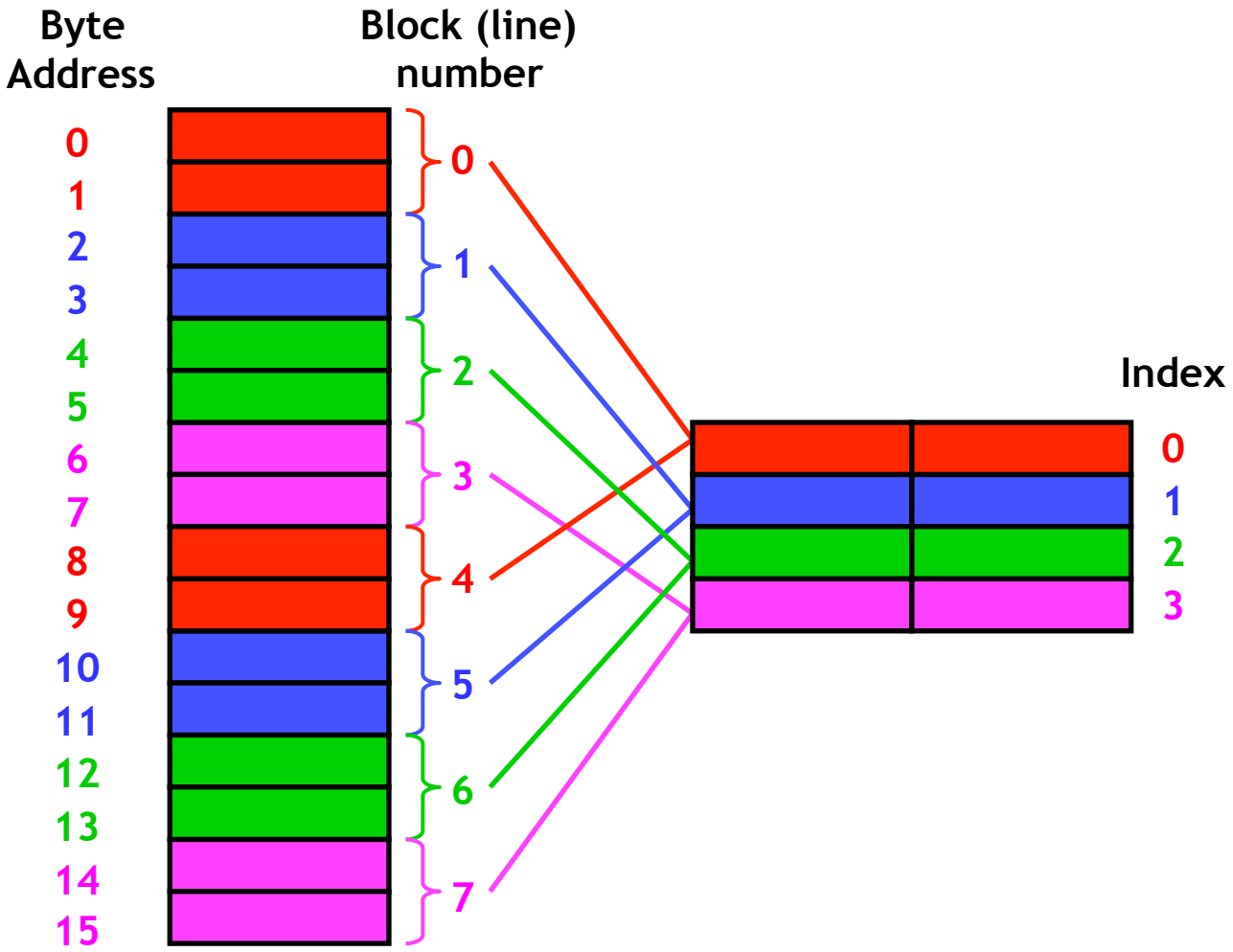


Hmm.. The cache might get confused later!
Why? And how do we solve that?

Use tags!


Memory
Cache
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
Index
00
01
10
11
Tag
Data
00
??
01
01

What's a cache block? (or *cache line)*



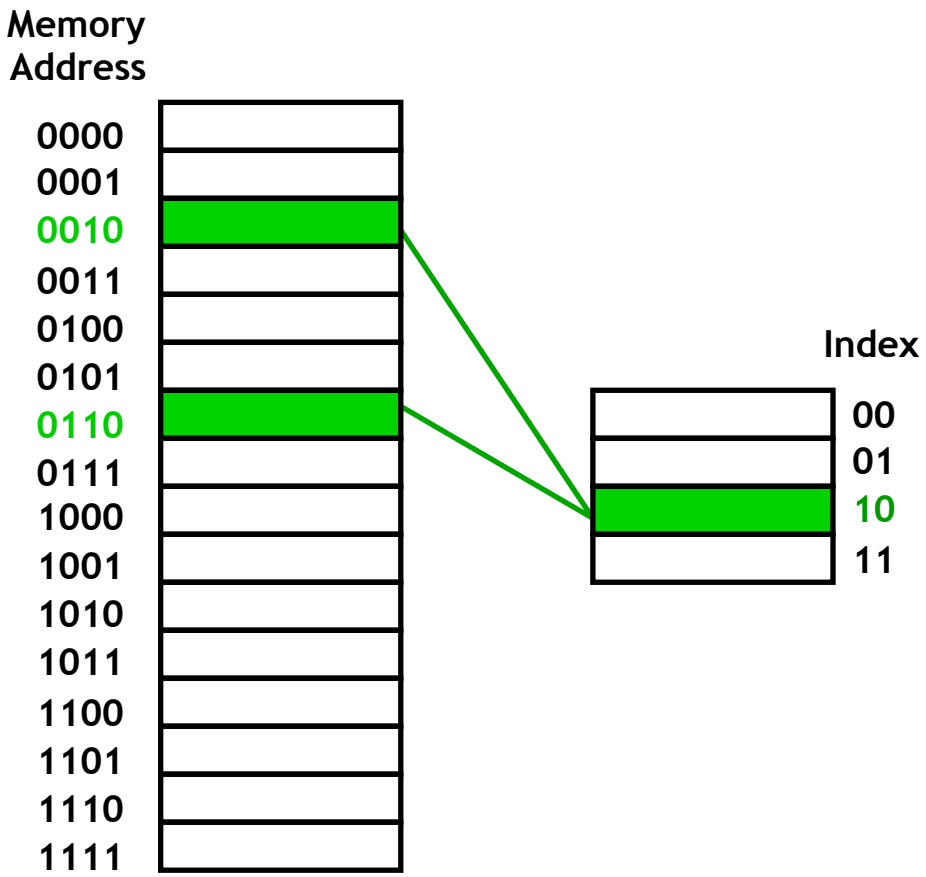
A puzzle.

- What can you infer from this:

- Cache starts *empty*
- Access (addr, hit/miss) stream
- (10, miss), (11, hit), (12, miss)

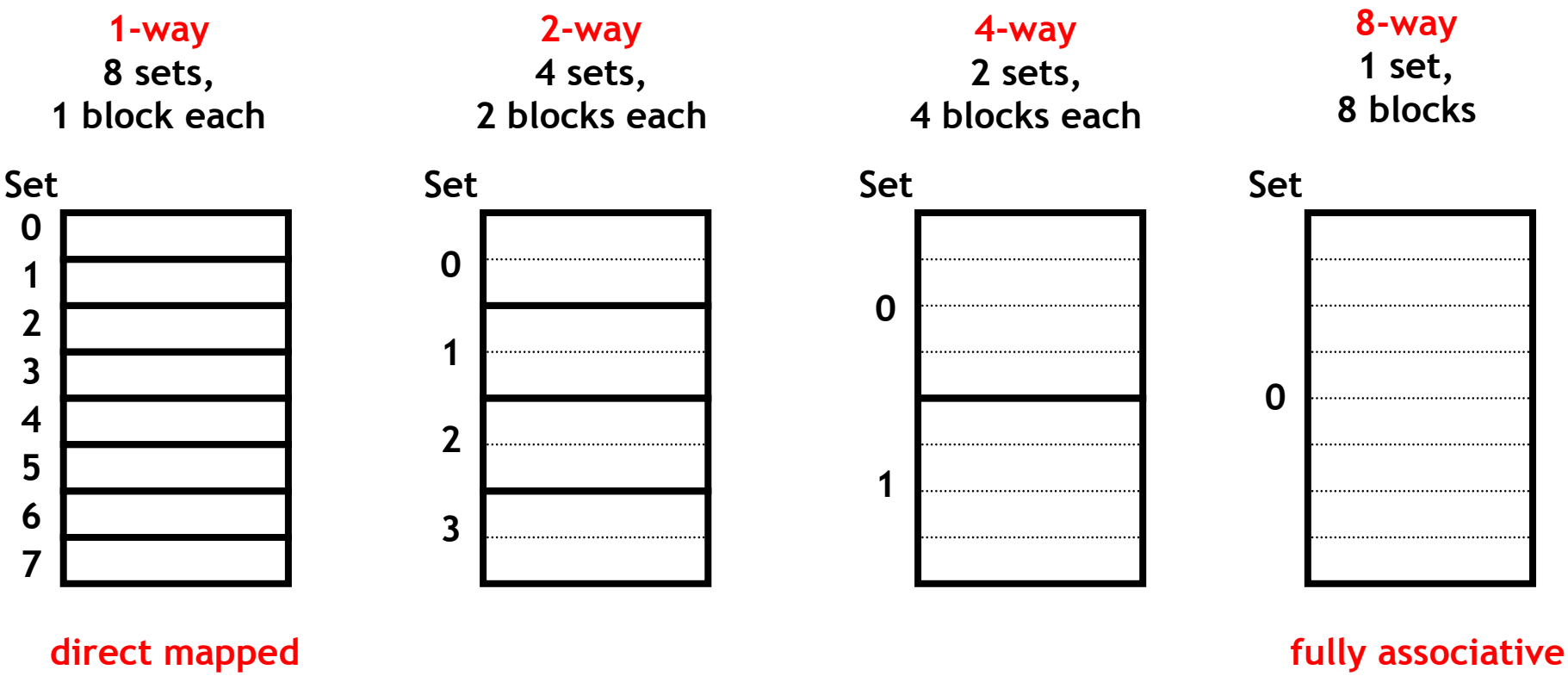
Problems with direct mapped caches?

- What happens if a program uses addresses 2, 6, 2, 6, 2, …?

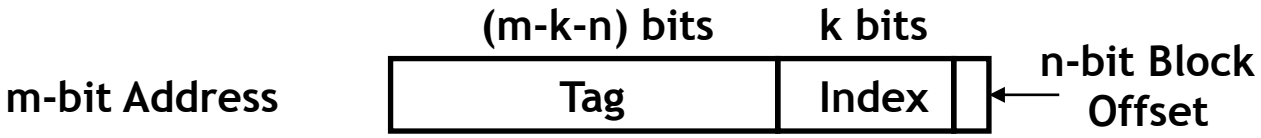


Associativity

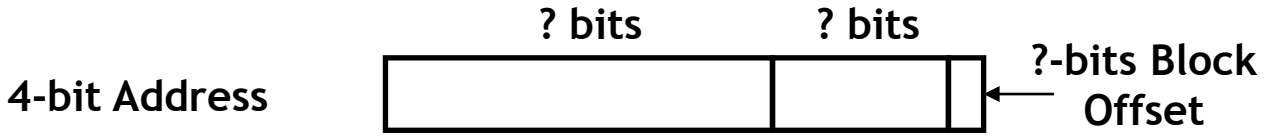
- What if we could store data in *any* place in the cache?
- But that might slow down caches… so we do something in between.



But now how do I know where data goes?

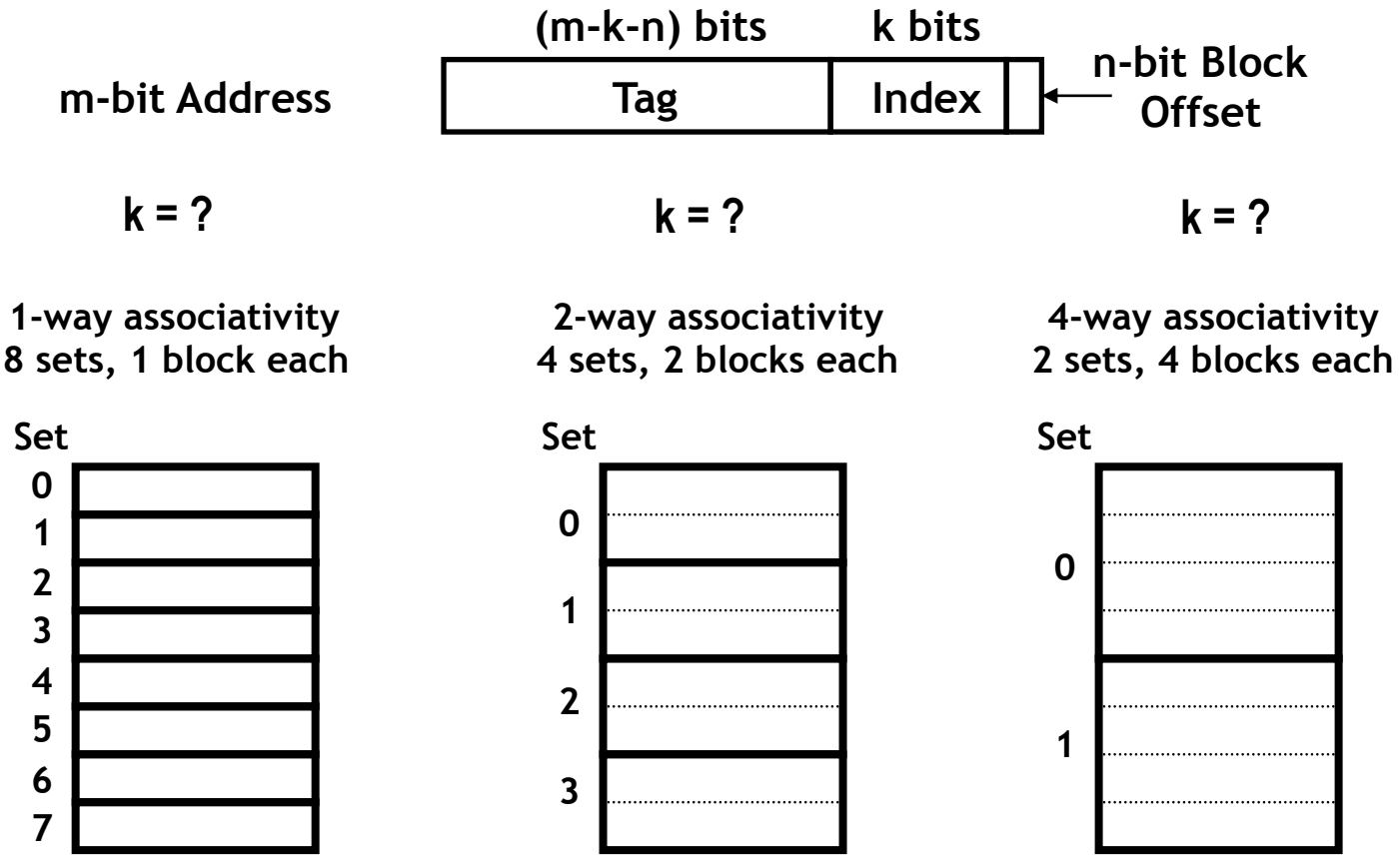


Our example used a 2^2-block cache with 2^1 bytes per block. Where would 13 (1101) be stored?



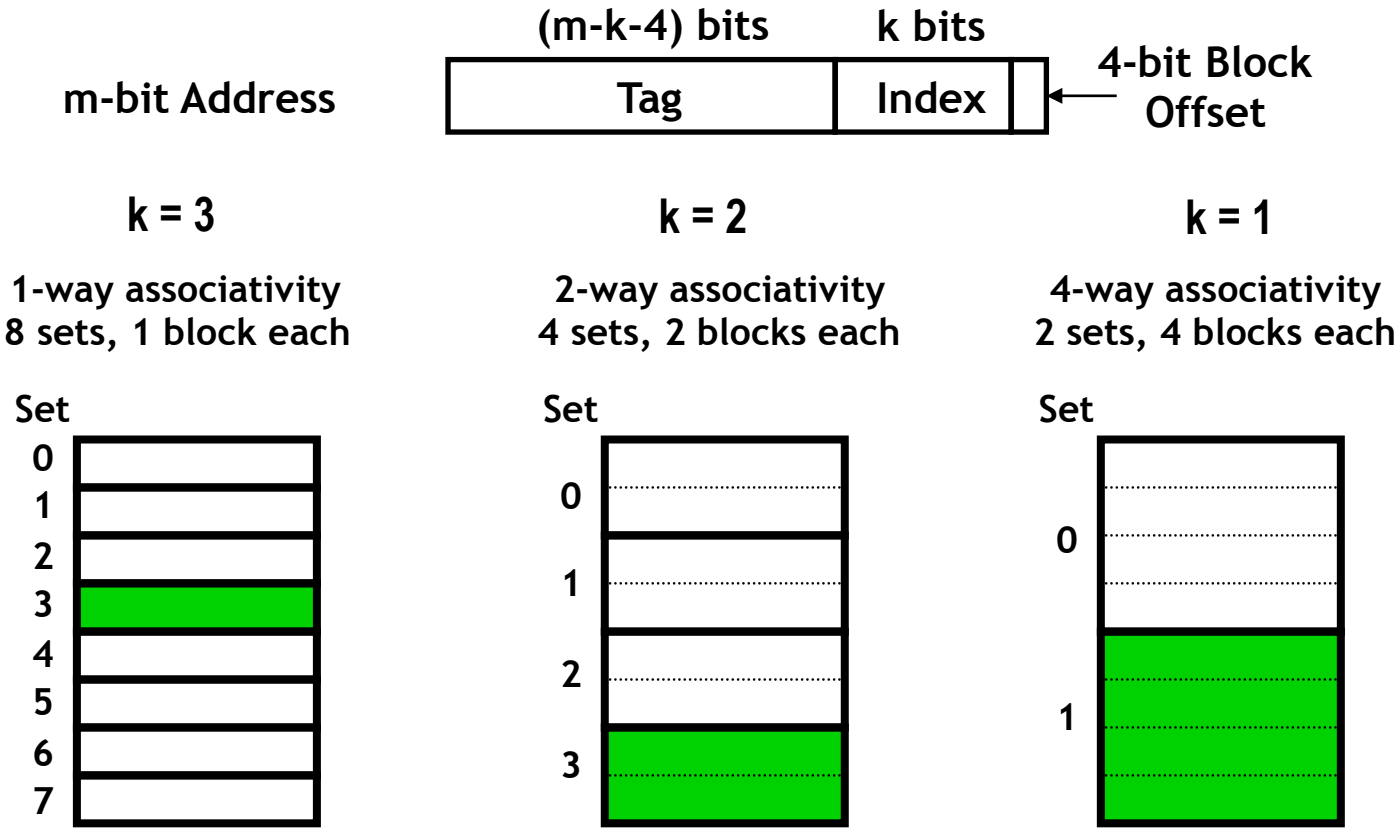
Example placement in set-associative caches

- Where would data from address 0x1833 be placed?
  - Block size is 16 bytes.
- 0x1833in binary is 00...0110000 011 0011.



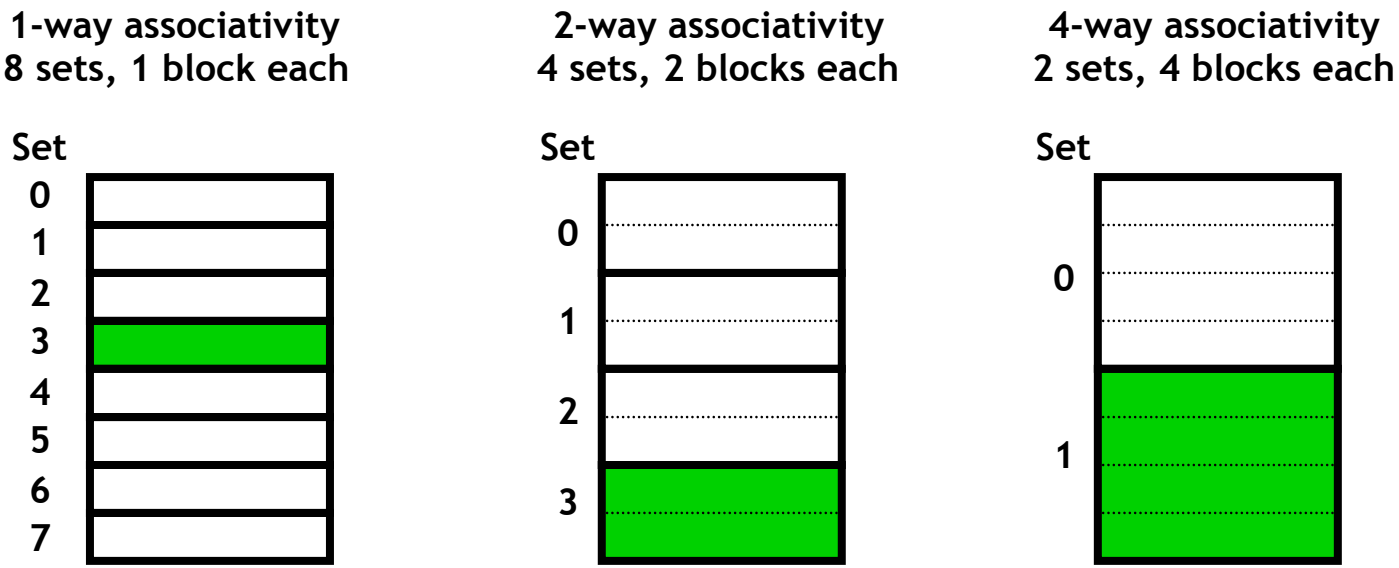
Example placement in set-associative caches

- Where would data from address 0x1833 be placed?
  - Block size is 16 bytes.
- 0x1833in binary is 00...0110000 011 0011.



Block replacement

- Any empty block in the correct set may be used for storing data.
- If there are no empty blocks, which one should we replace?
- Replace something, of course, but what?
  - Caches typically use something close to least-recently-used



Another puzzle.

- What can you infer from this:

- Cache starts *empty*
- Access (addr, hit/miss) stream
- (10, miss); (12, miss); (10, miss)