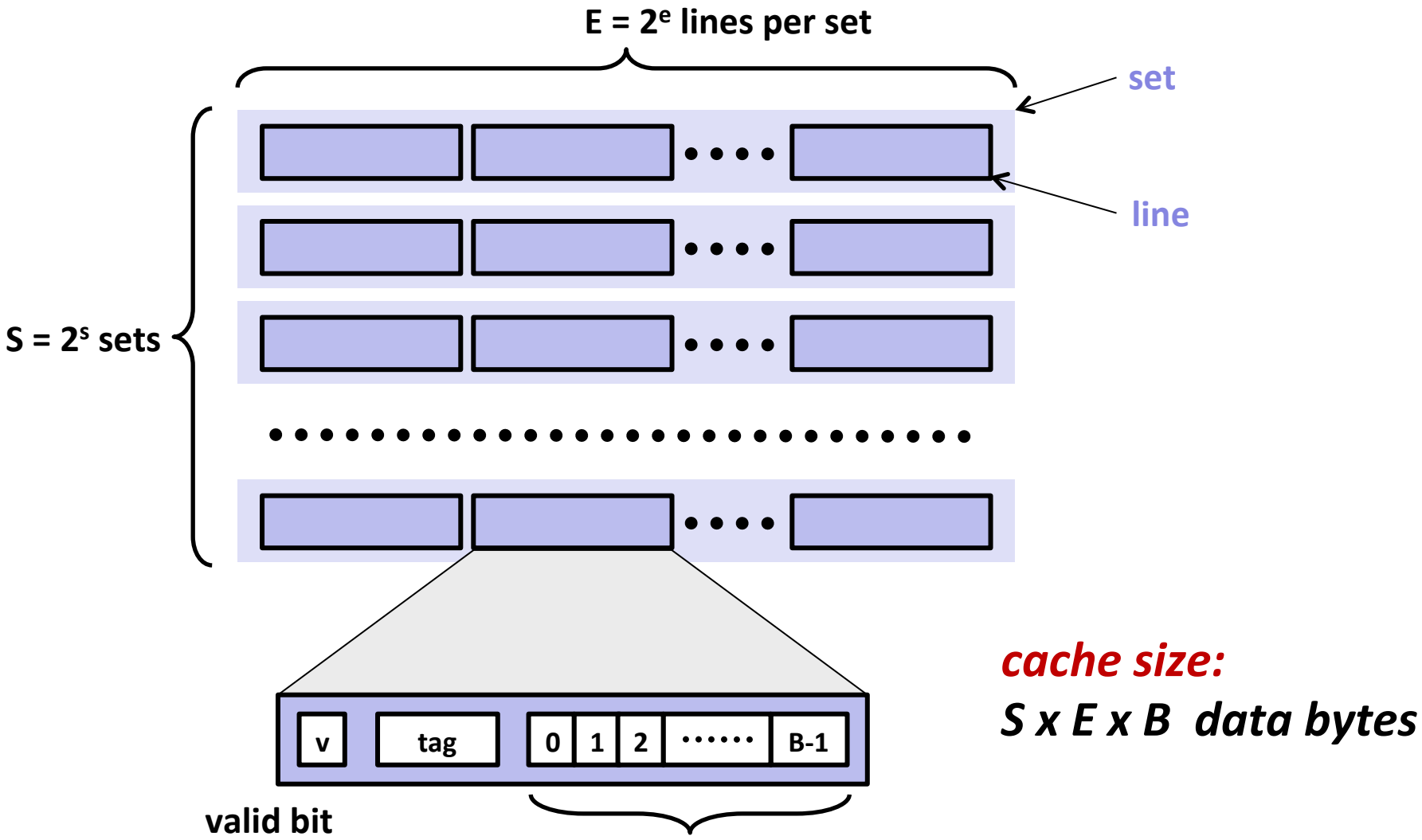
Section 7: Memory and Caches

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

General Cache Organization (S, E, B)

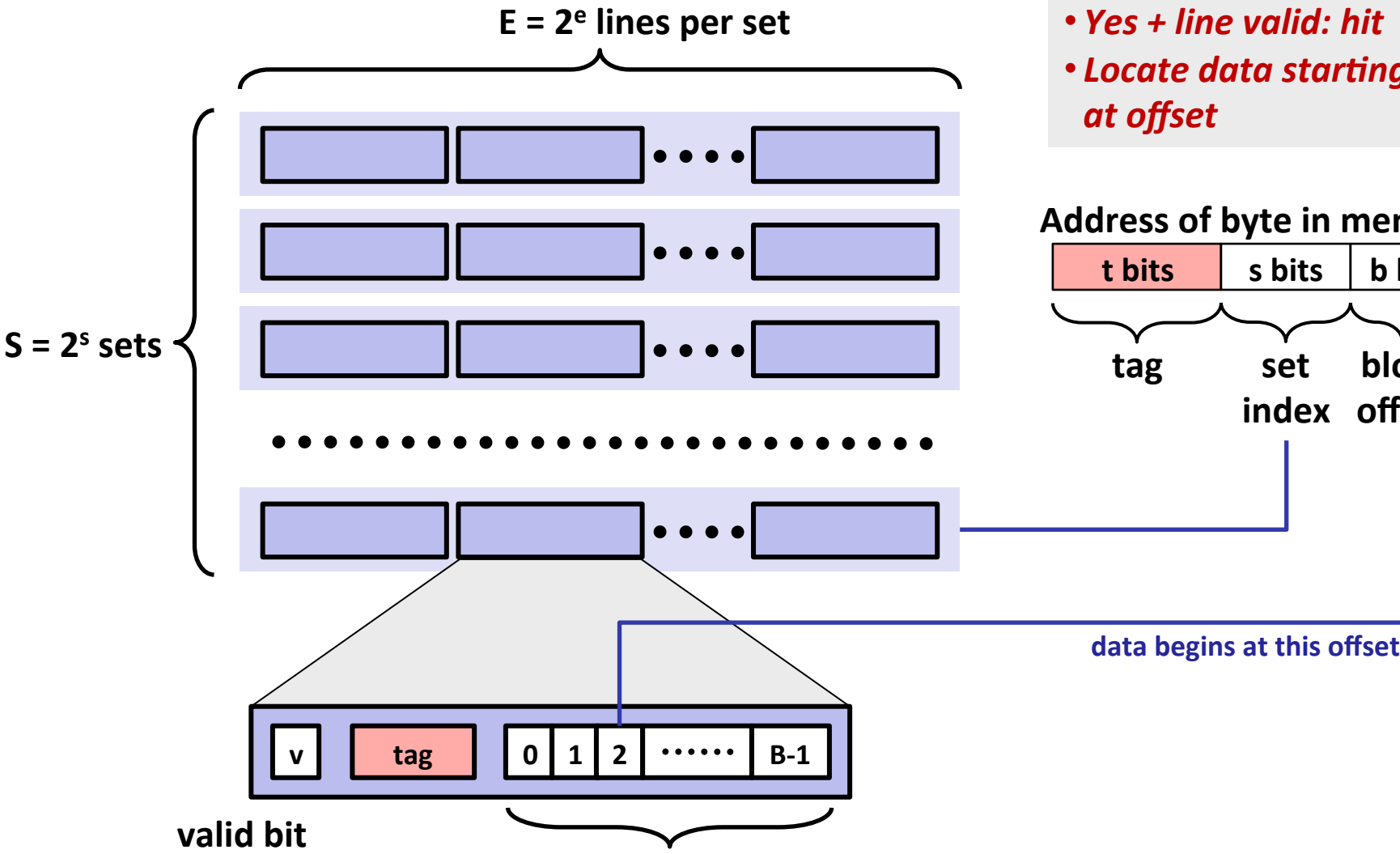


cache size:

S x E x B data bytes

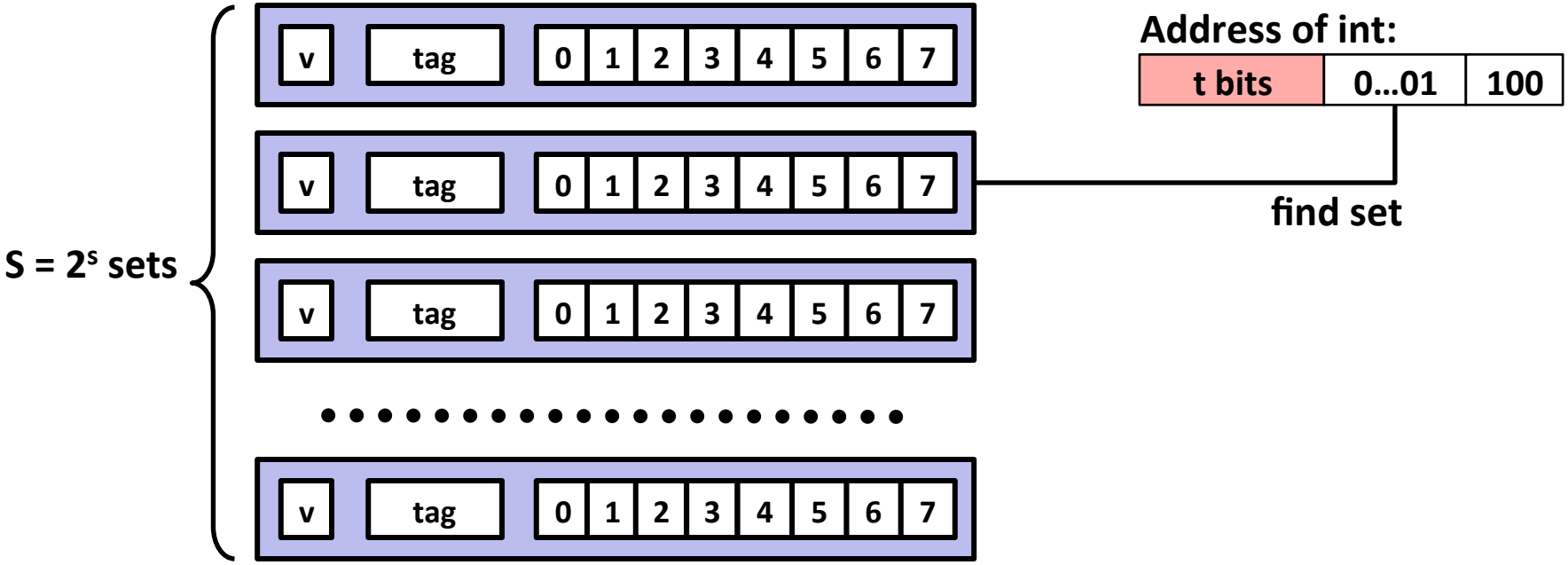
Cache Read

- *Locate set*
- *Check if any line in set has matching tag*
- *Yes + line valid: hit*
- *Locate data starting at offset*



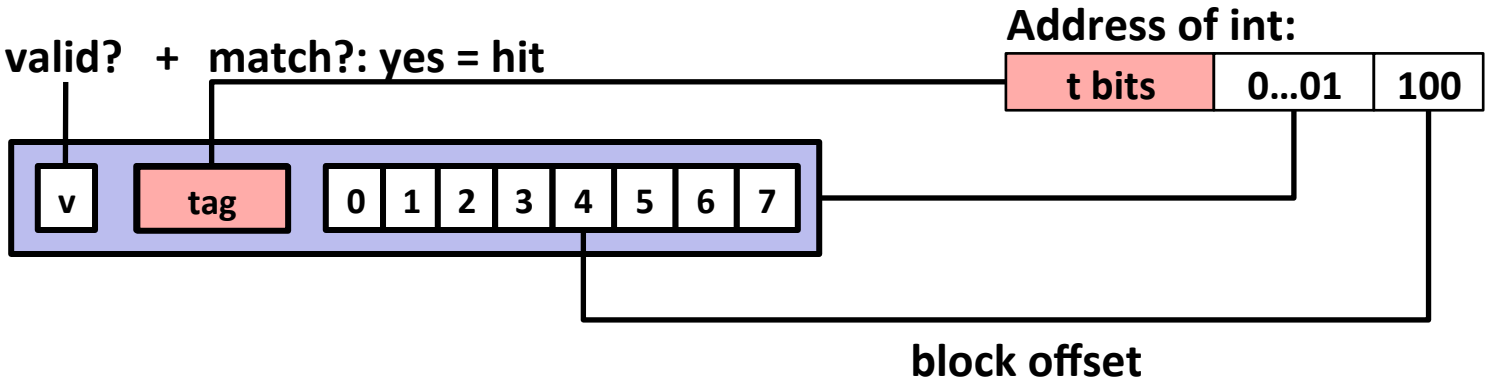
Example: Direct-Mapped Cache (E = 1)

Direct-mapped: One line per set
Assume: cache block size 8 bytes



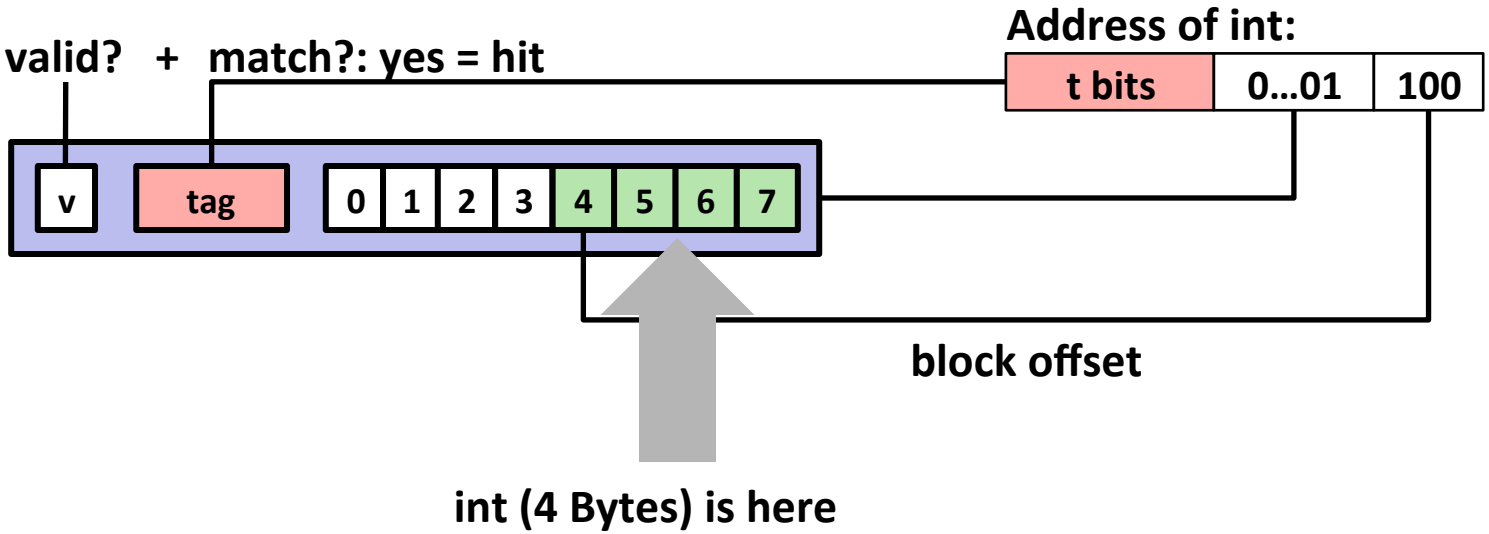
Example: Direct-Mapped Cache (E = 1)

Direct-mapped: One line per set
Assume: cache block size 8 bytes



Example: Direct-Mapped Cache (E = 1)

Direct-mapped: One line per set
Assume: cache block size 8 bytes

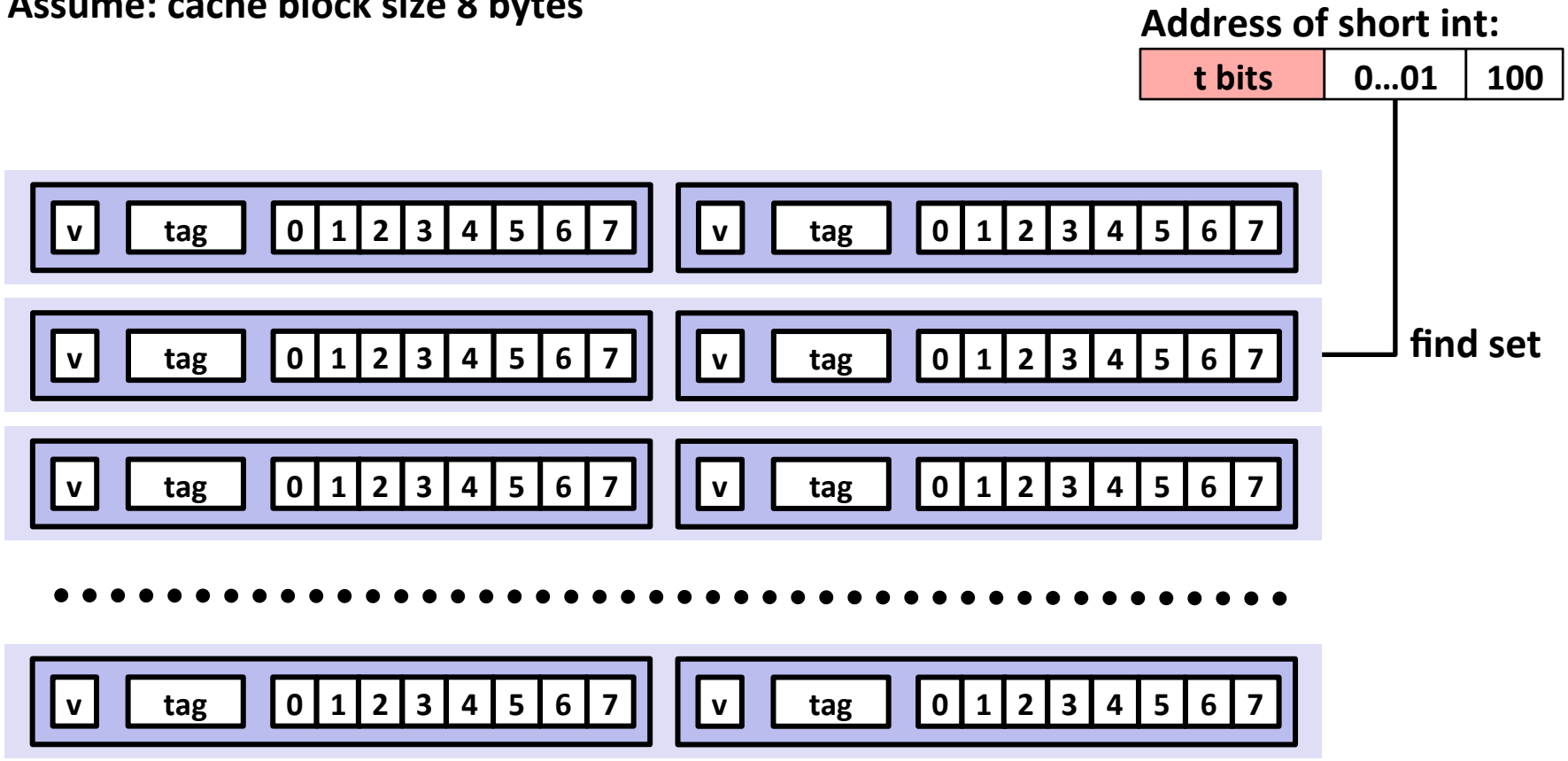


No match: old line is evicted and replaced

E-way Set-Associative Cache (Here: E = 2)

E = 2: Two lines per set

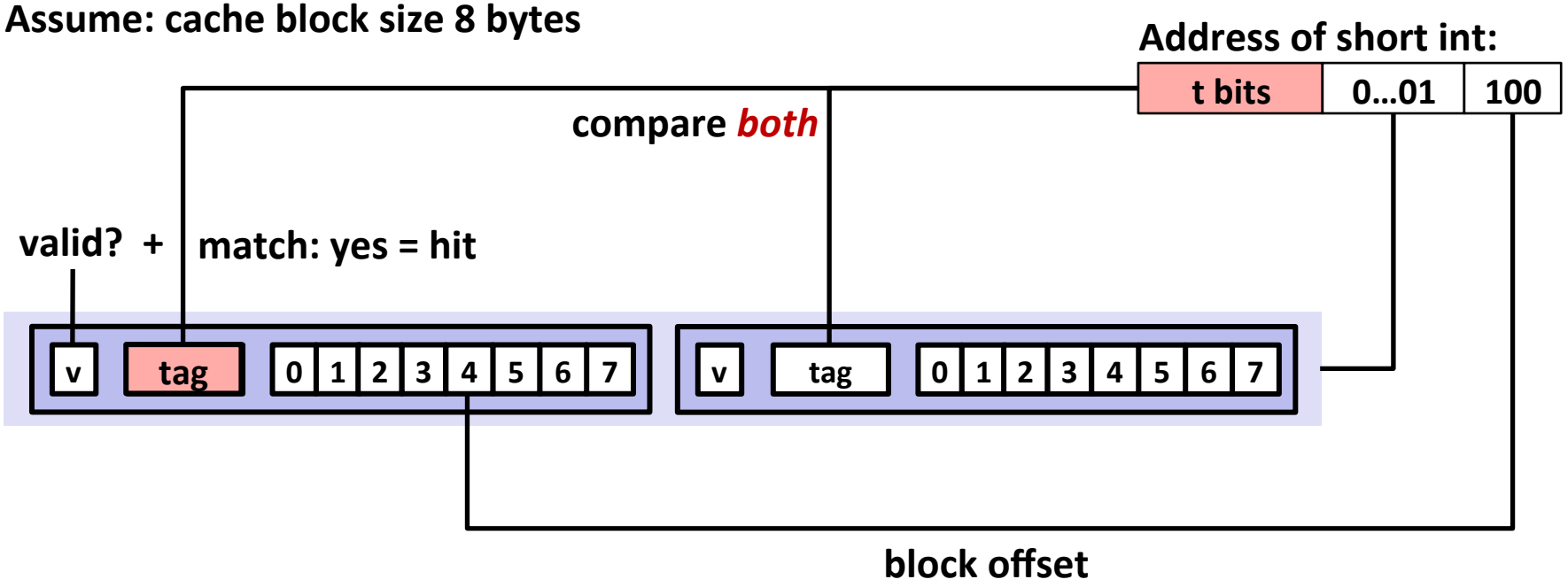
Assume: cache block size 8 bytes



E-way Set-Associative Cache (Here: E = 2)

E = 2: Two lines per set

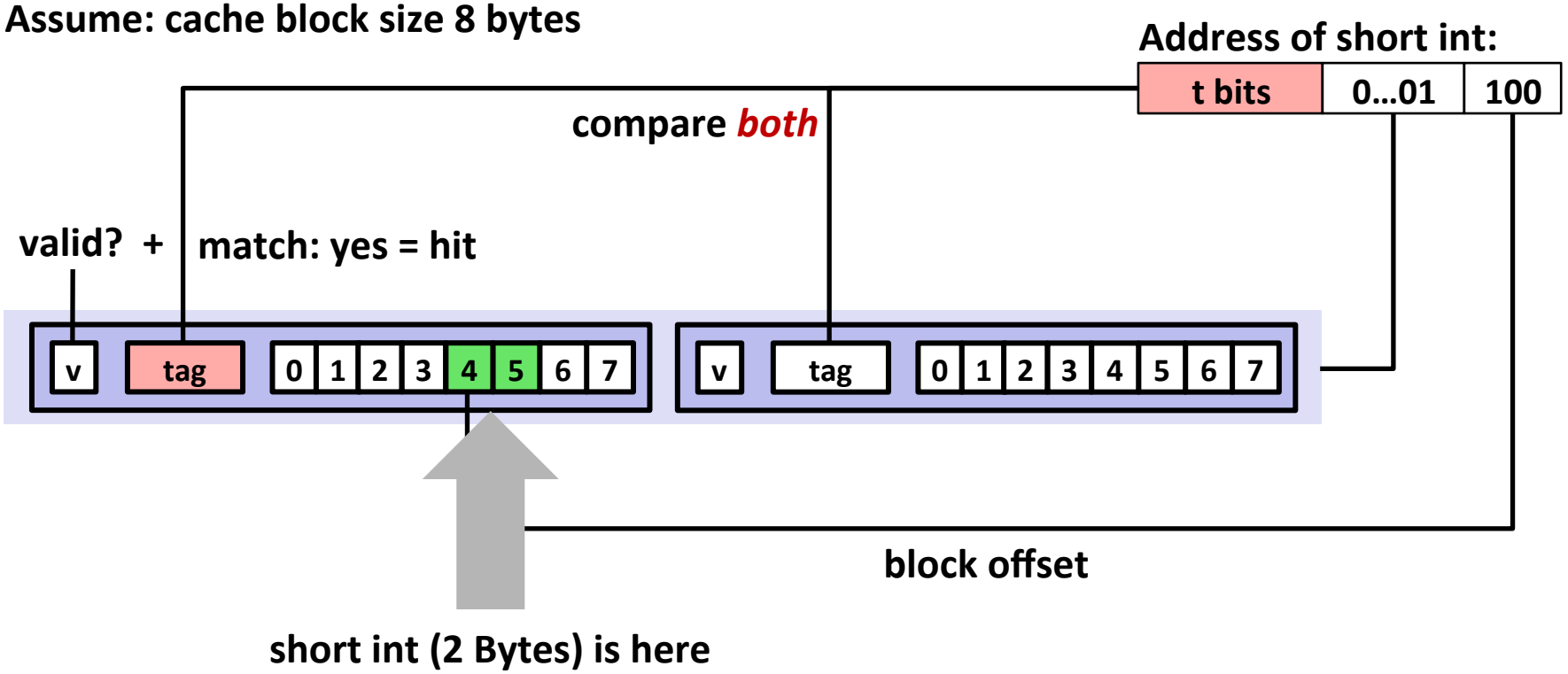
Assume: cache block size 8 bytes



E-way Set-Associative Cache (Here: E = 2)

E = 2: Two lines per set

Assume: cache block size 8 bytes



No match:

- One line in set is selected for eviction and replacement
- Replacement policies: random, least recently used (LRU), …

Types of Cache Misses

- **Cold (compulsory) miss**
  - Occurs on first access to a block
- **Conflict miss**
  - Most hardware caches limit blocks to a small subset (sometimes just one) of the available cache slots
    - if one (e.g., block i must be placed in slot (i mod size)), direct-mapped
    - if more than one, n-way set-associative (where n is a power of 2)
  - Conflict misses occur when the cache is large enough, but multiple data objects all map to the same slot
    - e.g., referencing blocks 0, 8, 0, 8, ... would miss every time
- **Capacity miss**
  - Occurs when the set of active cache blocks (the working set) is larger than the cache (just won't fit)

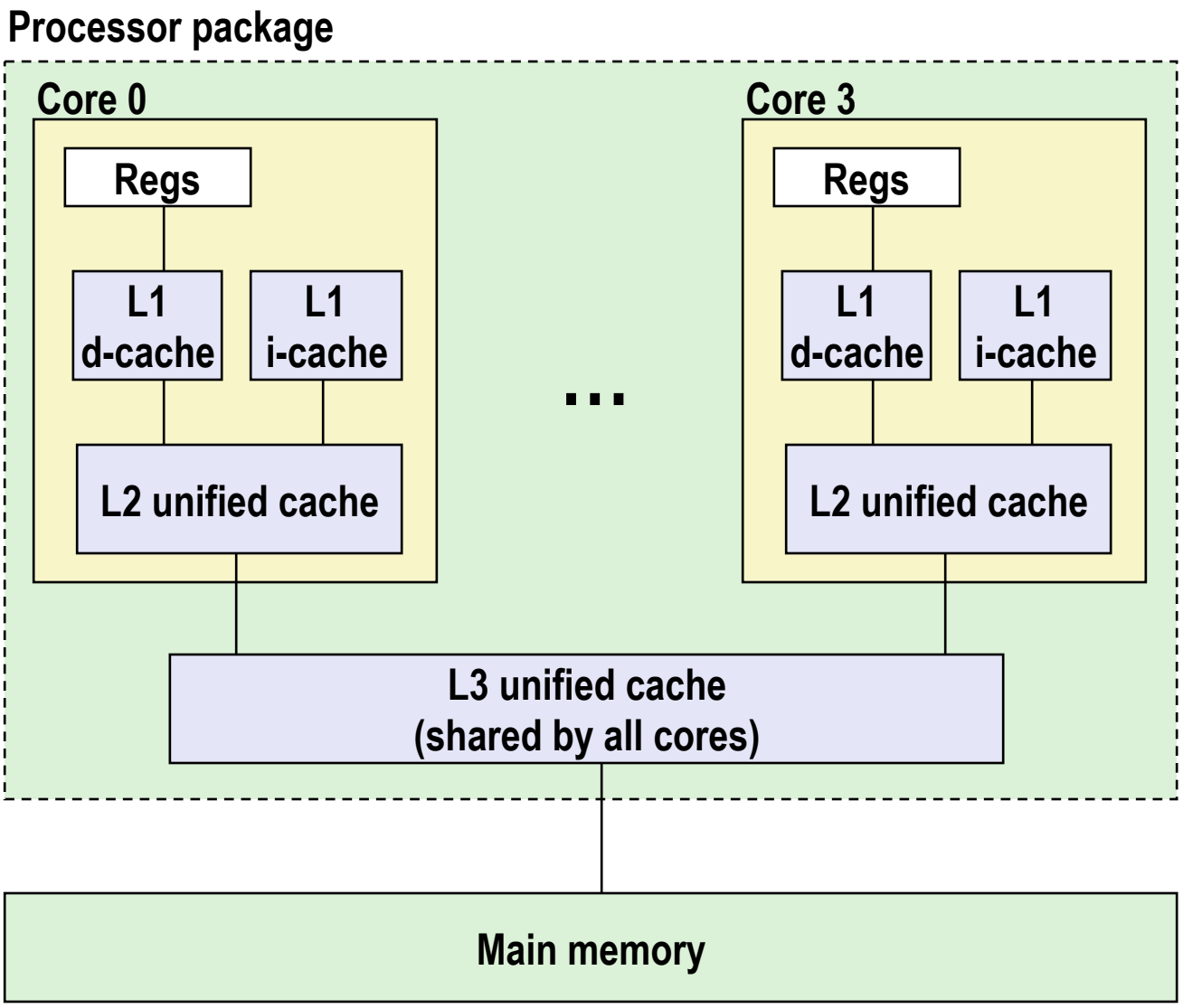
What about writes?

- **Multiple copies of data exist:**
  - L1, L2, possibly L3, main memory
- **What is the main problem with that?**

What about writes?

- **Multiple copies of data exist:**
  - L1, L2, possibly L3, main memory
- **What to do on a write-hit?**
  - Write-through (write immediately to memory)
  - Write-back (defer write to memory until line is evicted)
    - Need a *dirty bit* to indicate if line is different from memory or not
- **What to do on a write-miss?**
  - Write-allocate (load into cache, update line in cache)
    - Good if more writes to the location follow
  - No-write-allocate (just write immediately to memory)
- **Typical caches:**
  - Write-back + Write-allocate, usually
  - Write-through + No-write-allocate, occasionally

Intel Core i7 Cache Hierarchy



L1 i-cache and d-cache:
32 KB, 8-way,
Access: 4 cycles

L2 unified cache:
256 KB, 8-way,
Access: 11 cycles

L3 unified cache:
8 MB, 16-way,
Access: 30-40 cycles

Block size: 64 bytes for all caches.