Chapter 5
Large and Fast :
Exploiting the Memory Hierarchy

Cache Memory

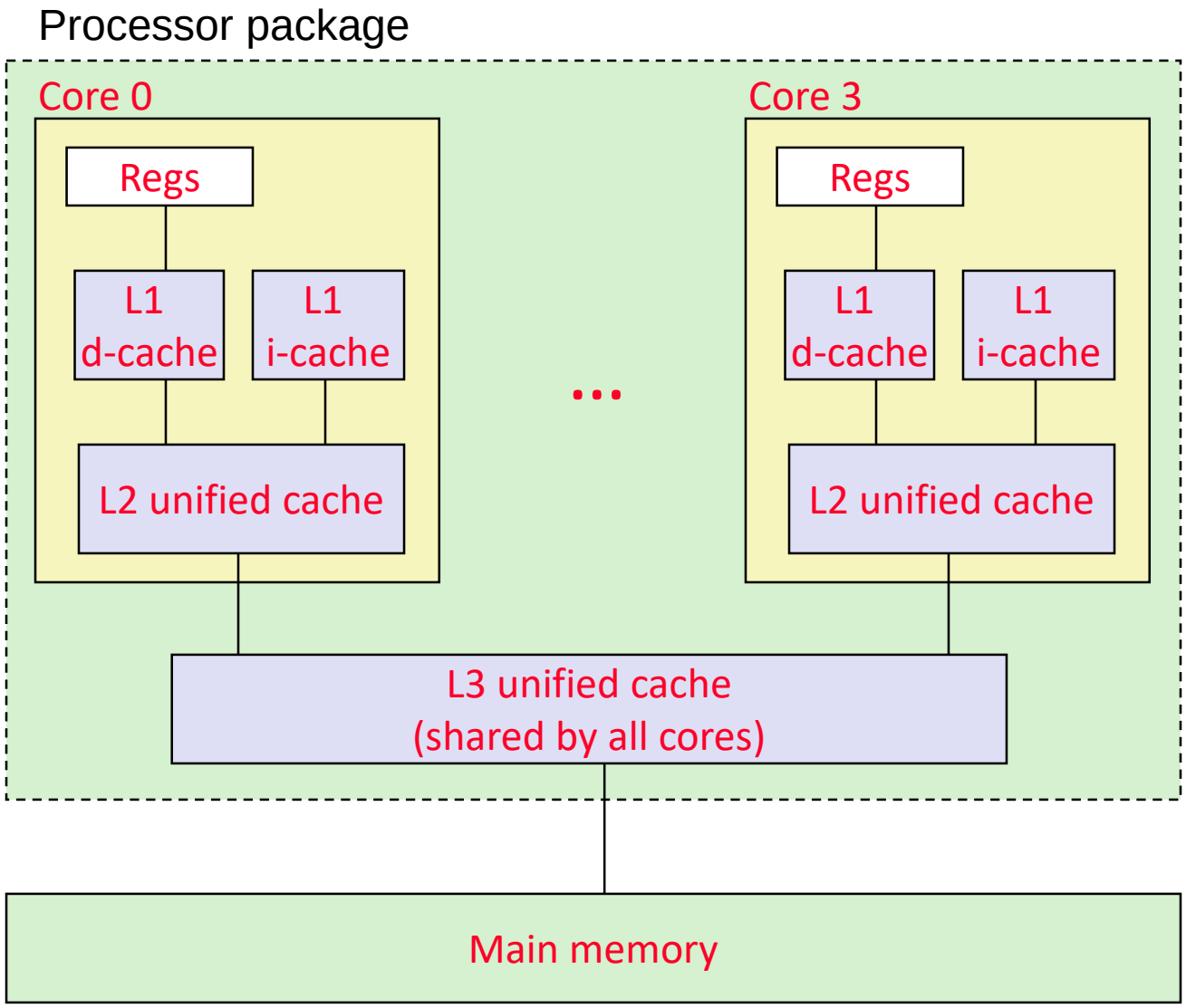
Review : Memory Hierarchies

- Fundamental idea of a memory hierarchy:
  - For each level k, the faster, smaller device at level k serves as a cache for the larger, slower device at level k+1

- Why do memory hierarchies work?
  - Because of locality, programs tend to access the data at level k more often than they access the data at level k+1
  - Thus, the storage at level k+1 can be slower, and thus larger and cheaper per bit

- *Big Idea:* The memory hierarchy creates a large pool of storage that costs as much as the cheap storage near the bottom, but that serves data to programs at the rate of the fast storage near the top

Example : Intel Core i7 Cache Hierarchy



Block size:
64 bytes for all caches

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

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

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

Cache Basics

- Two questions to answer (in hardware):
  - Q1: How do we know if a data item is in the cache?
  - Q2: If it is, how do we find it?

- **Direct mapped**
  - Each memory block is mapped to exactly one block in the cache
    - lots of lower level blocks must share blocks in the cache
  - Address mapping (to answer Q2): cache index =

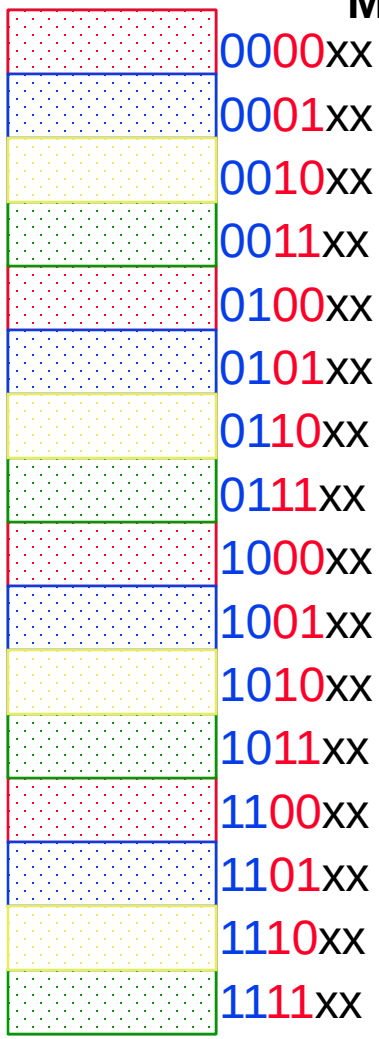
    (block address) **modulo** (# of blocks in the cache)
  - Have a tag associated with each cache block that contains the address information (the upper portion of the address) required to identify the block (to answer Q1)

Caching: A Simple First Example

Cache

Index	Valid	Tag	Data
00			
01			
10			
11			

Main Memory



One word blocks
Two low order bits define the byte in the word (32b words)

Q1: Is it there?

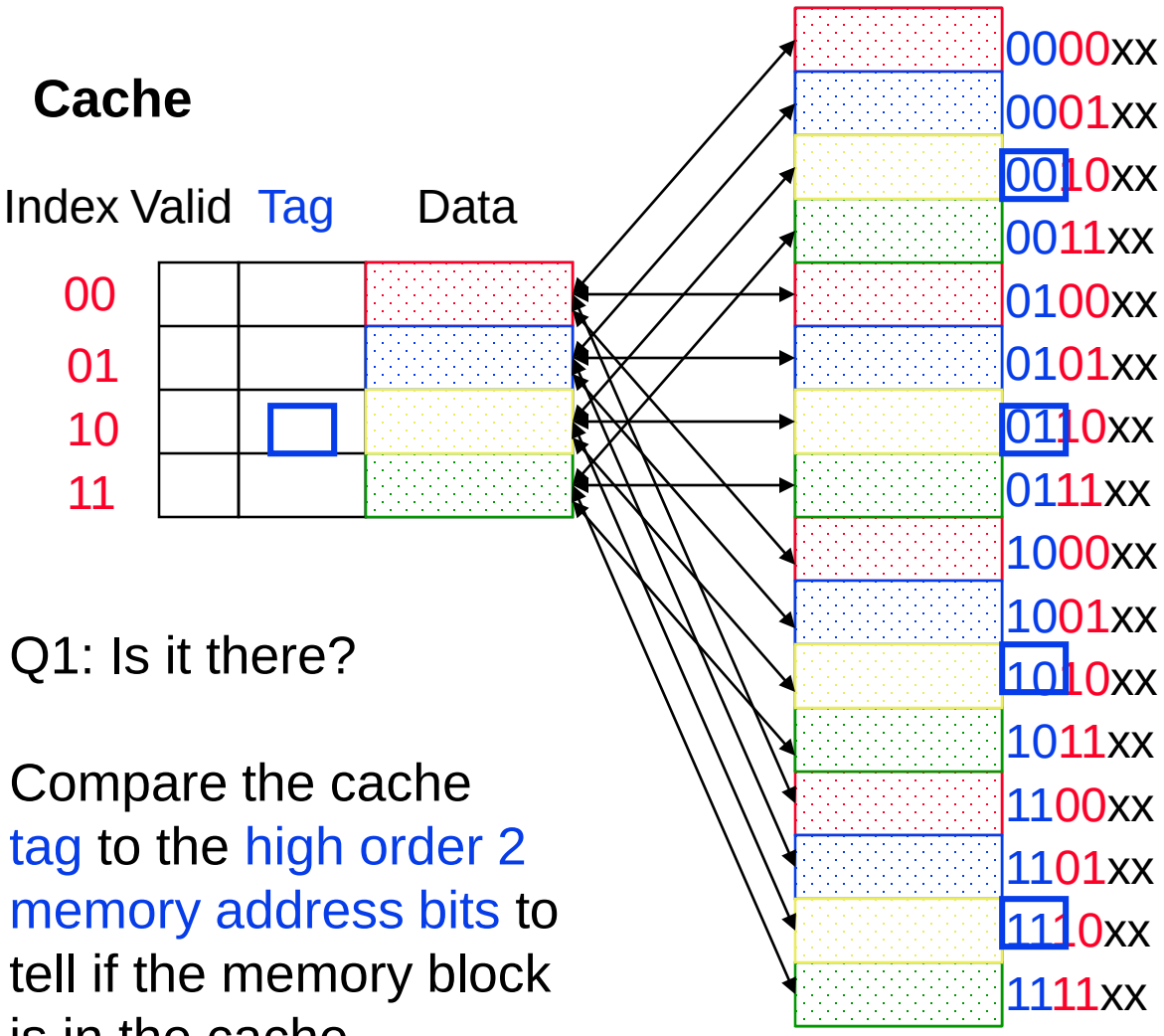
Compare the cache tag to the high order 2 memory address bits to tell if the memory block is in the cache

Q2: How do we find it?

Use next 2 low order memory address bits – the index – to determine which cache block (i.e., modulo the number of blocks in the cache)

(block address) modulo (# of blocks in the cache)

Caching: A Simple First Example



One word blocks
Two low order bits define the byte in the word (32b words)

Q1: Is it there?

Compare the cache tag to the high order 2 memory address bits to tell if the memory block is in the cache

Q2: How do we find it?

Use next 2 low order memory address bits – the index – to determine which cache block (i.e., modulo the number of blocks in the cache)

(block address) modulo (# of blocks in the cache)

- Place Data in Cache by Hashing Address!

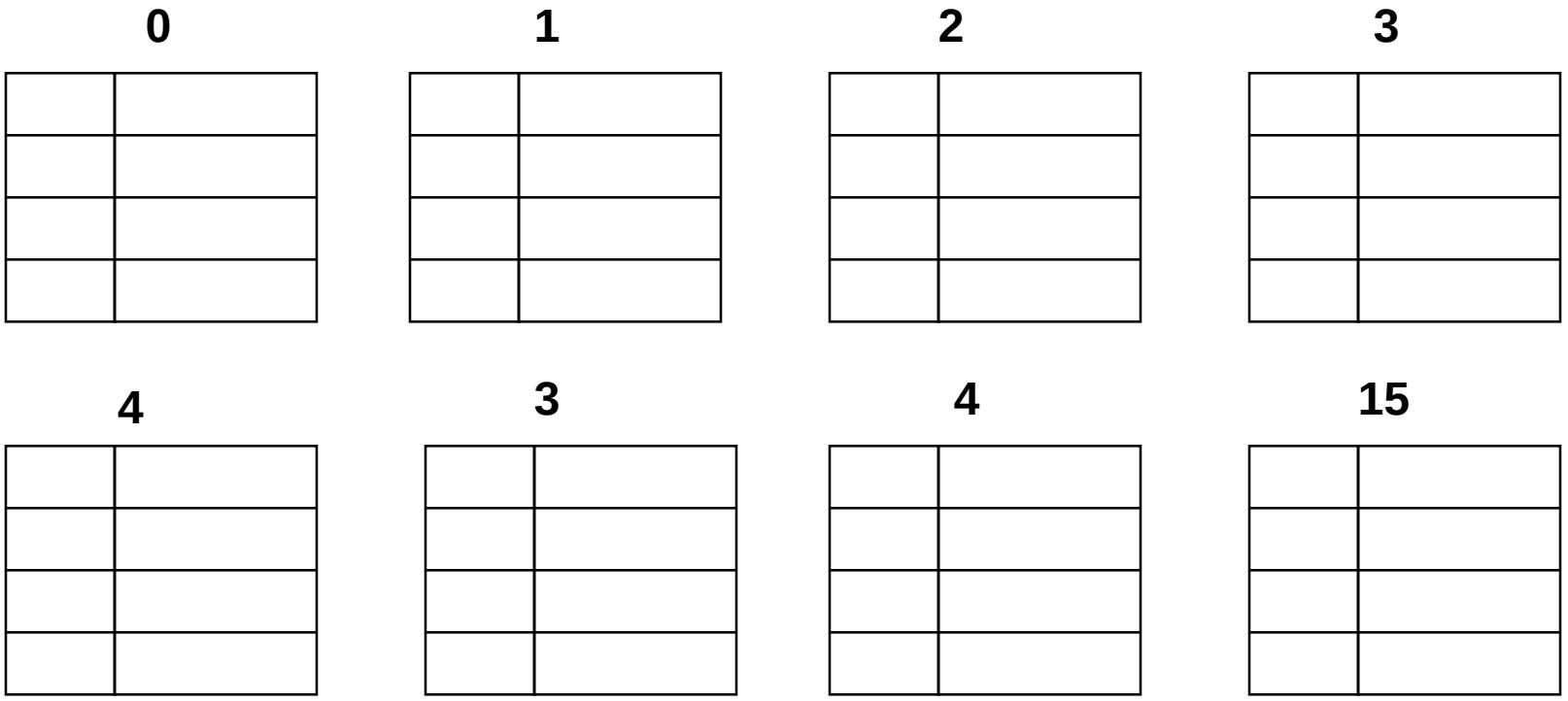
- Tags Differentiate Blocks in Same Index

Direct Mapped Cache

- Consider the main memory word reference string

Start with an empty cache - all blocks initially marked as not valid

0 1 2 3 4 3 4 15



Direct Mapped Cache

- Consider the main memory word reference string

Start with an empty cache - all blocks initially marked as not valid

0 1 2 3 4 3 4 15

0 miss

00	Mem(0)

1 miss

00	Mem(0)
00	Mem(1)

2 miss

00	Mem(0)
00	Mem(1)
00	Mem(2)

3 miss

00	Mem(0)
00	Mem(1)
00	Mem(2)
00	Mem(3)

4 miss

~~00~~ 01	~~Mem(0)~~ Mem(4)
00	Mem(1)
00	Mem(2)
00	Mem(3)

3 hit

01	Mem(4)
00	Mem(1)
00	Mem(2)
00	Mem(3)

4 hit

01	Mem(4)
00	Mem(1)
00	Mem(2)
00	Mem(3)

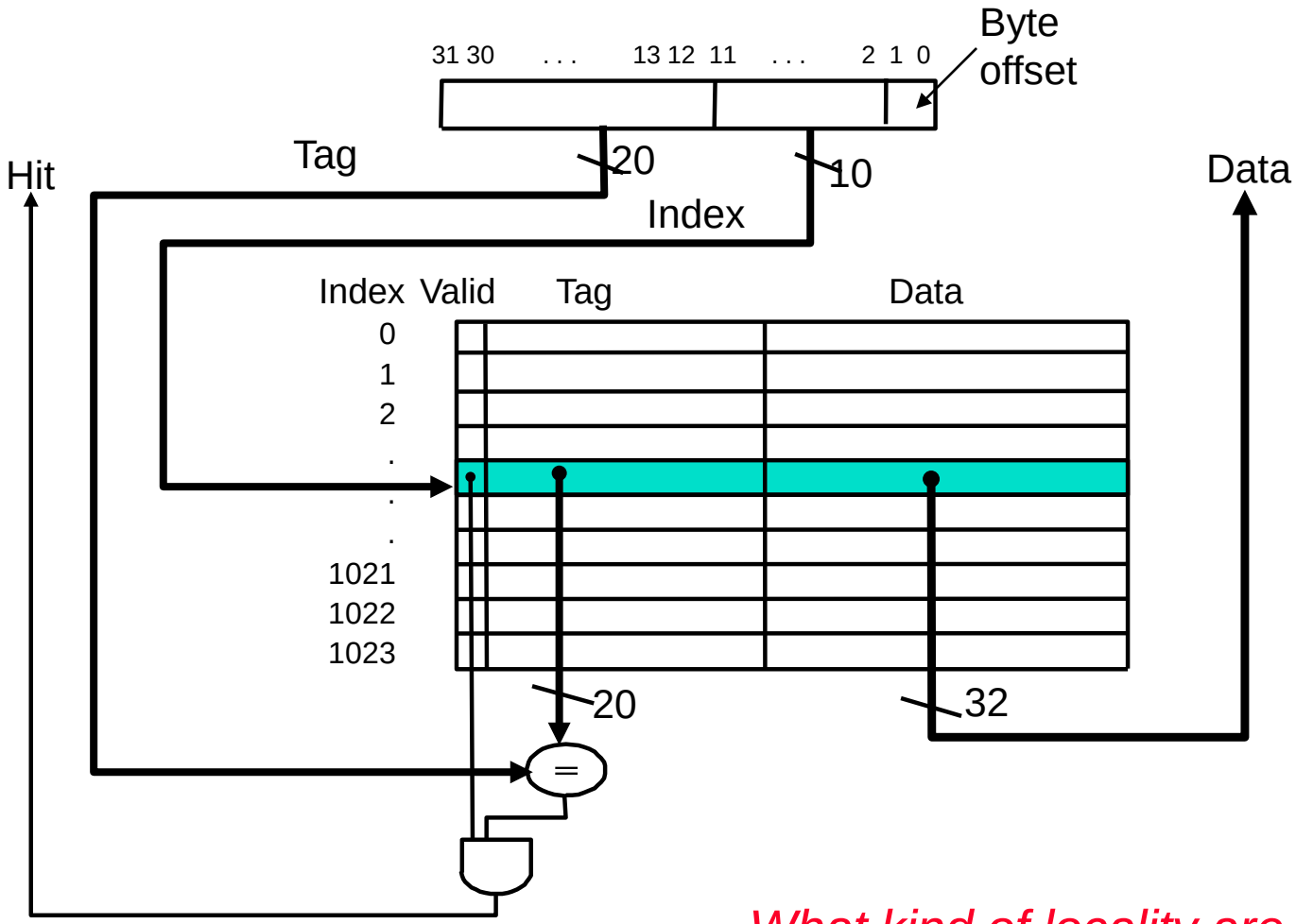
15 miss

01	Mem(4)
00	Mem(1)
00	Mem(2)
~~00~~ 11	~~Mem(3)~~ Mem(15)

- 8 requests, 6 misses

Direct Mapped Cache Example (Fig. 5.10)

- One word blocks, cache size = 1K words (or 4KB)



When is there a cache hit?
(Valid[index] = TRUE) **AND**
(Tag[index] = Tag[memory address])

What kind of locality are we taking advantage of?

Cache Field Sizes

- The number of bits in a cache includes both the storage for data and for the tags
  - 32-bit byte address
  - For a direct mapped cache with 2^n blocks, *n* bits are used for the index
  - For a block size of 2^m words (2^{m+2} bytes), *m* bits are used to address the word within the block and 2 bits are used to address the byte within the word

- What is the size of the tag field?

- The total number of bits in a direct-mapped cache is then

$$2^n \times (\text{block size} + \text{tag field size} + \text{valid field size})$$

Cache Field Sizes Examples (pp. 390)

- How many total bits are required for a direct mapped cache with 16KiB of data and 4-word blocks, assuming a 32-bit address?

- Mapping an Address to a Multiword Cache Block

Summary