SCHOOL *of* ENGINEERING

Department of Computer Science

Lecture #19
Ch. 6 The Memory Hierarchy (3)

Prof. Soraya Abad-Mota, PhD

Questions (answers must be clear from today)

Cache organization

1. What is a line of cache?
2. What is the size of a line of cache? What is stored in a line?
3. How are memory addresses used when storing information in the cache?
4. What is `m`? how does it relate to the size of the cache?
5. List and describe the parameters that define a cache.

Direct-Mapped Cache Simulation (recall)

t=1	s=2	b=1
x	xx	x

$t = m - (s+b) = 4-3 = 1$

M=16 bytes (4-bit addresses), B=2 bytes/block,
S=4 sets, E=1 line/set
(S,E,B,m) = (4,1,2,4) C = 8 bytes

Address trace (reads, one byte per read):

0	$[0\underline{00}0_2]$,	miss
1	$[0\underline{00}1_2]$,	hit
7	$[0\underline{11}1_2]$,	miss
8	$[1\underline{00}0_2]$,	miss
0	$[0\underline{00}0_2]$	miss

underlined set Index.
Order in which these addesses are required

	v	Tag	Block
Set 0	1	0	M[0-1]
Set 1			
Set 2			
Set 3	1	0	M[6-7]

Direct-Mapped Cache Simulation

(new requests)

t=1	s=2	b=1
x	xx	x

$t = m - (s+b) = 4-3 = 1$

M=16 bytes (4-bit addresses), B=2 bytes/block,
S=4 sets, E=1 line/set
(S,E,B,m) = (4,1,2,4) C = 8 bytes

Address trace (reads, one byte per read):

1	$[0\underline{00}1_2]$,
13	$[\underline{110}1_2]$,

underlined set Index.
Order in which these addesses are required

	v	Tag	Block
Set 0	1	0	M[0-1]
Set 1			
Set 2			
Set 3	1	0	M[6-7]

Continue the simulation with this cache

- Status as shown in previous slide
- Read word at address 1 = 0001, which set?
  - hit or miss?

- Read word at address 13 = 1101
  - hit or miss?

Questions about the process?

- How do you know if it is a hit or miss?

The cachelab is coming up

- Last big project of the semester
- All you need for this project is to understand how the cache works.
- Statement will be published when we have covered all of the cache behavior.
- First a cache homework, to make sure you understand

Issue with thrashing

- Read example with function dotprod (p. 622)
- To illustrate Thrashing (similar to what happened in the previous example) but now we try to find a solution.

- Cache parameters and memory addresses for this e.g.

Thrashing example #2

```
float dotprod(float x[8], float y[8]){
  float sum = 0.0;
  for (int i=0; i < 8; i++)
        sum += x[i] * y[i];
  return sum;
}
```

- Access pattern for x and y?
- Direct-mapped cache, 2 sets, block of 16 bytes

Thrashing example #2

- x stored starting at address 0 up to address 28
- y stored from address 32, elem y[7] at address 60

		set			set
x[0]	0$\underline{0}$0000	0	y[0] 32	1$\underline{0}$0000	0
x[1]	0$\underline{0}$0001	0	y[1] 36	1$\underline{0}$0000	0
…					
x[4]	0$\underline{1}$0000	1	y[4] 48	1$\underline{1}$0000	1
…					
x[7]	0$\underline{1}$1100	1	y[7] 48	1$\underline{1}$0000	1

Thrashing example #2

- given the cache characteristics and sizes of x, y each element of `x` falls in the same slot of cache as the corresponding element of `y`

Solution:

- add padding to x, to make it x[12] instead of 8 floats,
- now x addresses go from 0 to 44 and if y starts immediately after x, y's first address is 48 instead of 32.
- 32 to 44 map to set 0,

  48 to 60 map to set 1

Practice Problem 6.10

- Problem 6.10 p. 624
    - with changes in addresses, what fraction of the accesses to x and y will be hit?
    - verify on your own how the number of hits in this variation compares to the first example without the padding

Practice Problem 6.11 (on your own)

- Problem 6.11 p. 624 is good to understand why we do not use the high-order bits in the address to determine the set
- We rather use the middle bits to determine the set

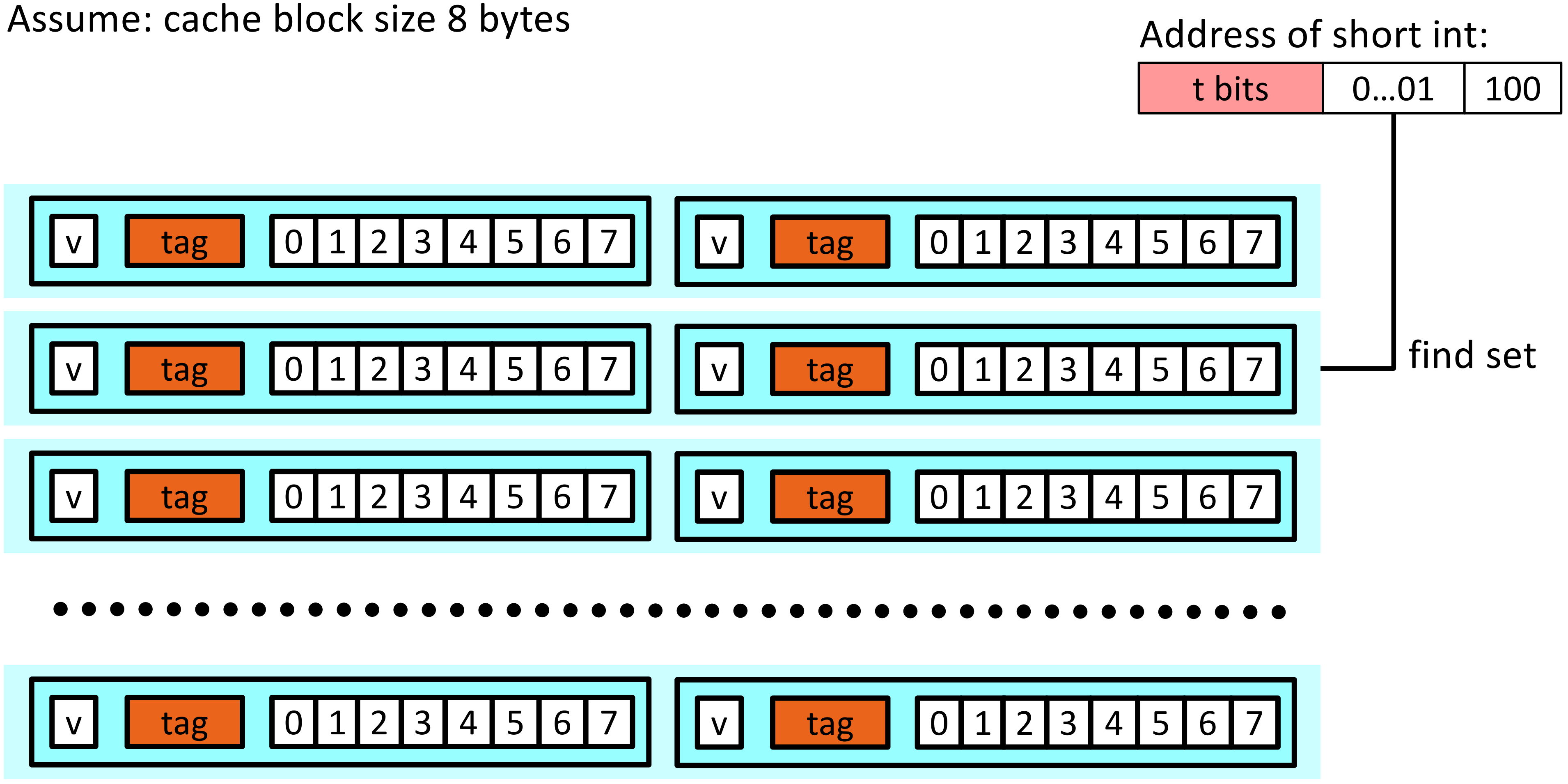
Detailed topics

- Cache memory organization and operation
  - Mapping of memory address to cache in detail
  - Simulation example of reads from memory, hit or miss?
  - E-way set associative cache
  - Writes
  - Cache metrics

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

E = 2: Two lines per set

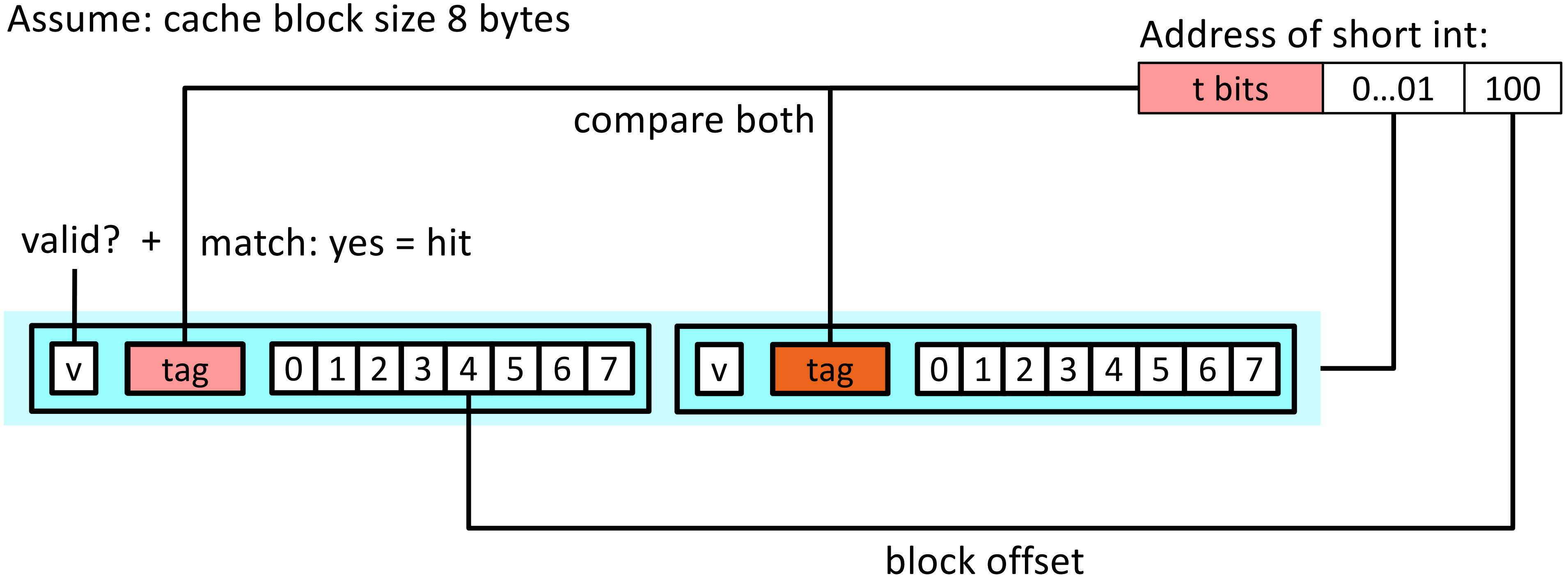
Assume: cache block size 8 bytes



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

E = 2: Two lines per set

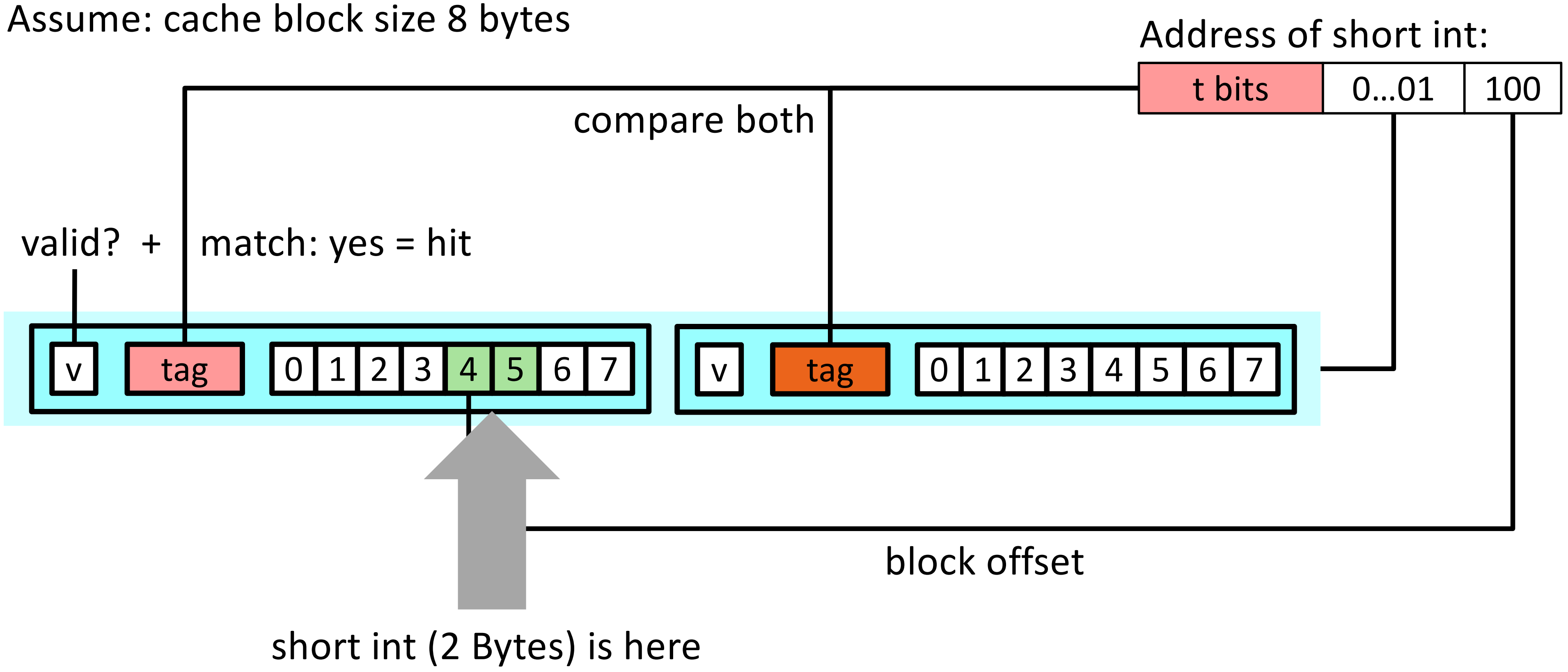
Assume: cache block size 8 bytes



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

E = 2: Two lines per set

Assume: cache block size 8 bytes



if no match:

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

2-Way Set Associative Cache Simulation

t=2	s=1	b=1
xx	x	x

M=16 byte 4-bit addresses, B=2 bytes/block,
S=2 sets, E=2 lines/set (i.o.w. 2 blocks/set)

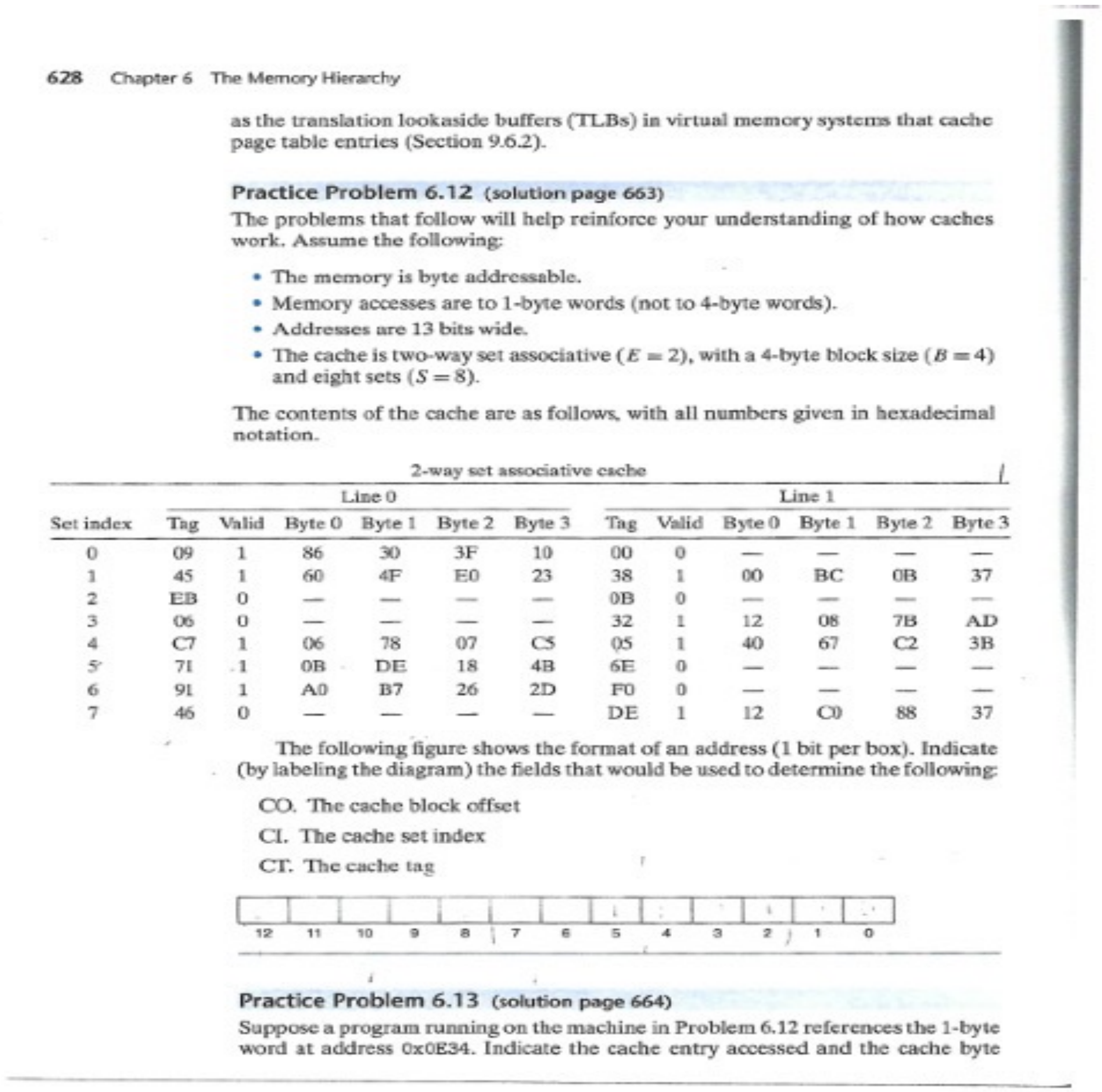
Address trace (reads, one byte per read):

0	$[00\underline{0}0_2]$,	miss
1	$[00\underline{0}1_2]$,	hit
7	$[01\underline{1}1_2]$,	miss
8	$[10\underline{0}0_2]$,	miss
0	$[00\underline{0}0_2]$	hit

	v	Tag	Block
Set 0	1	00	M[0-1]
	1	10	M[8-9]
Set 1	1	01	M[6-7]
	0		

Practice (pp. 628-630)

- Problem 6.12 and 6.13 now
- Problems 6.14 and 6.15 are similar to 6.13 (do them on your own they are good practice for set associative caches. The solutions are in the book.).
- Problem 6.16 is important to verify that you understand the mapping between memory addresses and cache.



as the translation lookaside buffers (TLBs) in virtual memory systems that cache page table entries (Section 9.6.2).

Practice Problem 6.12 (solution page 663)

The problems that follow will help reinforce your understanding of how caches work. Assume the following:

- The memory is byte addressable.
- Memory accesses are to 1-byte words (not to 4-byte words).
- Addresses are 13 bits wide.
- The cache is two-way set associative ($E = 2$), with a 4-byte block size ($B = 4$) and eight sets ($S = 8$).

The contents of the cache are as follows, with all numbers given in hexadecimal notation.

2-way set associative cache

	Line 0						Line 1					
Set index	Tag	Valid	Byte 0	Byte 1	Byte 2	Byte 3	Tag	Valid	Byte 0	Byte 1	Byte 2	Byte 3
0	09	1	86	30	3F	10	00	0	—	—	—	—
1	45	1	60	4F	E0	23	38	1	00	BC	0B	37
2	EB	0	—	—	—	—	0B	0	—	—	—	—
3	06	0	—	—	—	—	32	1	12	08	7B	AD
4	C7	1	06	78	07	C5	05	1	40	67	C2	3B
5	71	1	0B	DE	18	4B	6E	0	—	—	—	—
6	91	1	A0	B7	26	2D	F0	0	—	—	—	—
7	46	0	—	—	—	—	DE	1	12	C0	88	37

The following figure shows the format of an address (1 bit per box). Indicate (by labeling the diagram) the fields that would be used to determine the following:

CO. The cache block offset

CI. The cache set index

CT. The cache tag

12	11	10	9	8	7	6	5	4	3	2	1	0

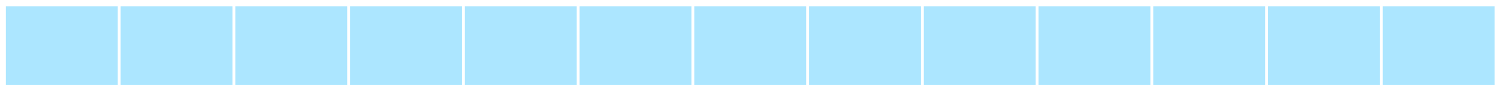
Practice Problem 6.13 (solution page 664)

Suppose a program running on the machine in Problem 6.12 references the 1-byte word at address 0x0E34. Indicate the cache entry accessed and the cache byte

Cache characteristics for problem 6.12

p. 628 (content of cache is in this same page)

- Memory is byte-addressable
- Memory accesses are to 1-byte words
- Addresses are 13 bits wide
- Two-way set associative cache (lines per set? __), with 4-byte block size and 8 sets. s = b = t=
- A cache address uses: which bits for each?



Cache characteristics for problem 6.12

p. 628 (content of cache is in this same page)

- Memory is byte-addressable
- Memory accesses are to 1-byte words
- Addresses are 13 bits wide
- Two-way set associative cache (lines per set? 2), with 4-byte block size and 8 sets. s = 3 b = 2 t= 8
- A cache address uses: which bits for each?

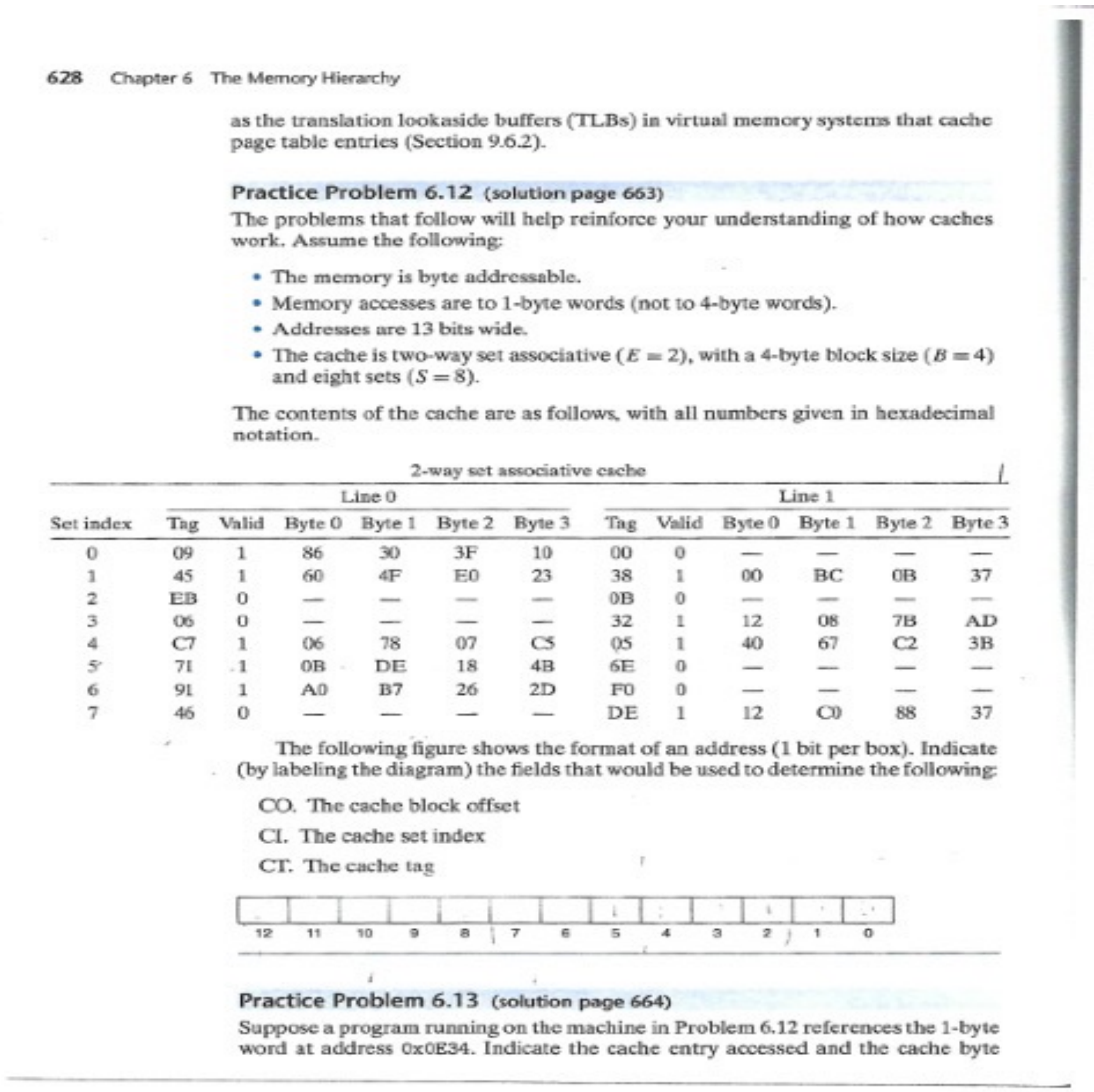
CT	CT	CT	CT	CT	CT	CT	CT	CI	CI	CI	co	CO

6.13 Program references one byte at address 0x0E34

- Place it in the format of the cache address

Memory reference:

Parameter	Value
Cache block offset (CO)	0x
Cache set index (CI)	0x
Cache tag (CT)	0x
Cache hit (Y/N)	
Cache byte returned	



628 Chapter 6 The Memory Hierarchy

as the translation lookaside buffers (TLBs) in virtual memory systems that cache page table entries (Section 9.6.2).

Practice Problem 6.12 (solution page 663)

The problems that follow will help reinforce your understanding of how caches work. Assume the following:

- The memory is byte addressable.
- Memory accesses are to 1-byte words (not to 4-byte words).
- Addresses are 13 bits wide.
- The cache is two-way set associative ($E = 2$), with a 4-byte block size ($B = 4$) and eight sets ($S = 8$).

The contents of the cache are as follows, with all numbers given in hexadecimal notation.

2-way set associative cache

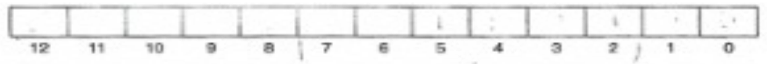
	Line 0						Line 1					
Set index	Tag	Valid	Byte 0	Byte 1	Byte 2	Byte 3	Tag	Valid	Byte 0	Byte 1	Byte 2	Byte 3
0	09	1	86	30	3F	10	00	0	—	—	—	—
1	45	1	60	4F	E0	23	38	1	00	BC	0B	37
2	EB	0	—	—	—	—	0B	0	—	—	—	—
3	06	0	—	—	—	—	32	1	12	08	7B	AD
4	C7	1	06	78	07	C5	05	1	40	67	C2	3B
5	71	1	0B	DE	18	4B	6E	0	—	—	—	—
6	91	1	A0	B7	26	2D	F0	0	—	—	—	—
7	46	0	—	—	—	—	DE	1	12	C0	88	37

The following figure shows the format of an address (1 bit per box). Indicate (by labeling the diagram) the fields that would be used to determine the following:

CO. The cache block offset

CI. The cache set index

CT. The cache tag



Practice Problem 6.13 (solution page 664)

Suppose a program running on the machine in Problem 6.12 references the 1-byte word at address 0x0E34. Indicate the cache entry accessed and the cache byte

Recall the cache characteristics from problem 6.12 p. 628

- Memory is byte-addressable
- Memory accesses are to 1-byte words
- Addresses are 13 bits wide
- Two-way set associative cache, with 4-byte block size and 8 sets.
- A cache address uses:

CT	CT	CT	CT	CT	CT	CT	CT	CI	CI	CI	CO	CO

Steps to find data in the cache (summary)

1. Convert the address to binary
2. Place the appropriate bits on the m-bit address pattern. (m for our problem is? ____)
3. Search for the address in the cache:
   - 3.1 Find the set
   - 3.2 Match the tag with one of the lines in that set
   - 3.3 Verify that valid bit is 1
   - 3.4 Find block offset and grab how many bytes? 1 in 6.1

One more Practice Problem (similar to 6.13)

- Given the cache with the same characteristics as in problem 6.12 and its contents as shown on p. 628
- The CPU is trying to access a short at address 0x00B2

Actions:

- Place this address in the cache address format.
- Values for:
  - cache block offset (CO)
  - cache set index (CI)
  - cache tag (CT)

  cache hit? (Y/N)

  cache bytes returned?

Program references two bytes at address 0x00B2

- Place it in the format of the cache address

Memory reference:

Parameter	Value
Cache block offset (CO)	0x
Cache set index (CI)	0x
Cache tag (CT)	0x
Cache hit (Y/N)	
Cache byte returned	

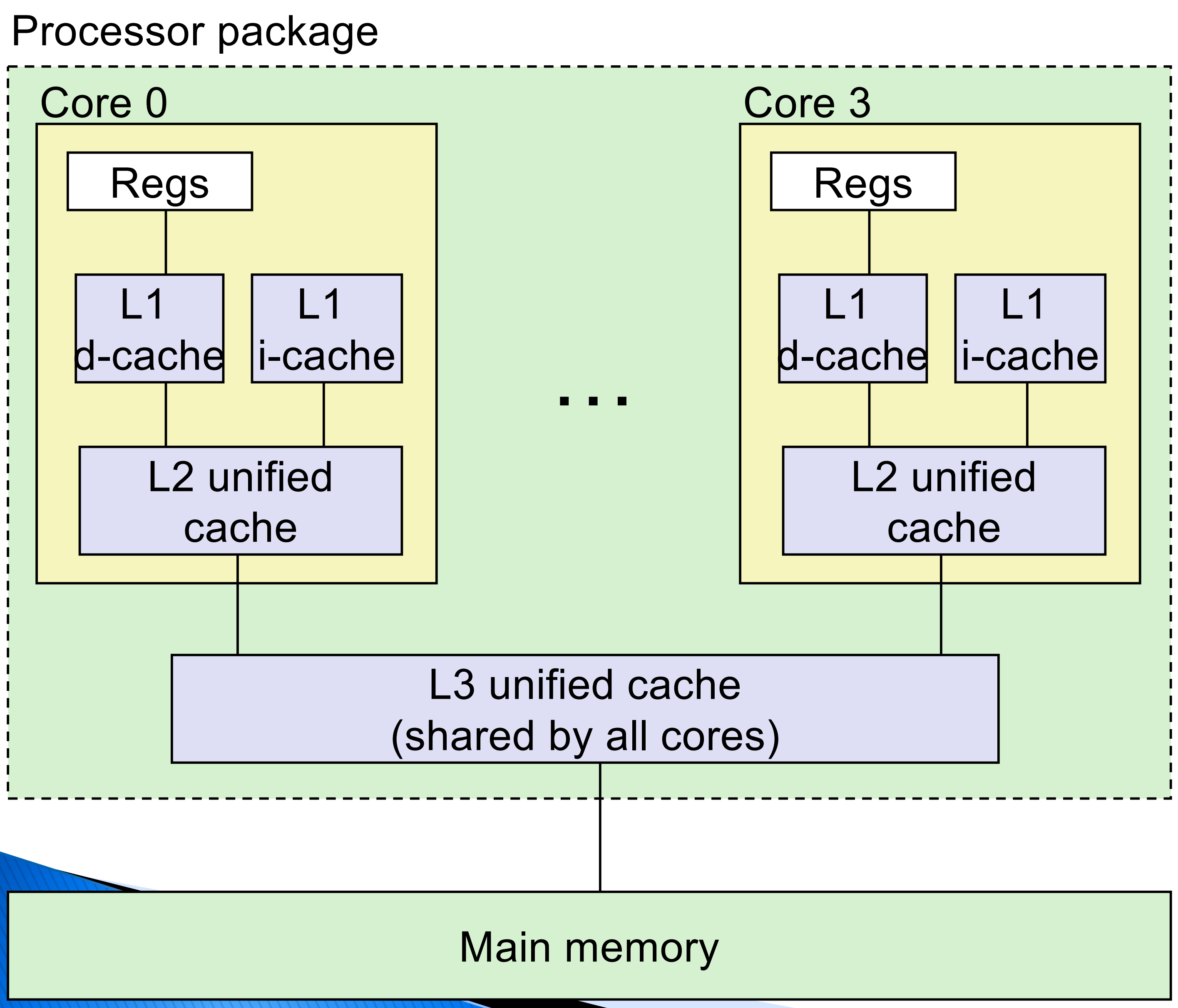
Topics

- Chapter 6: The Memory Hierarchy
  - Detailed calculations of problem 6.4
  - Locality of reference
  - Caching in the memory hierarchy (Sec. 6.3)
- Cache memory organization and operation (Sec. 6.4)
  - Writes
  - Cache metrics
- Performance impact of caches
  - The memory mountain
  - Rearranging loops to improve spatial locality
  - Using blocking to improve temporal locality

What about writes? (section 6.4.5, p.630)

- Multiple copies of data exist:
  - L1, L2, L3, Main Memory, Disk
- What to do on a write-hit?
  - Write-through (write immediately to memory)
  - Write-back (defer write to memory until replacement of line)
    - Need a dirty bit (line different from memory or not)
- What to do on a write-miss? (what is a write-miss?)
  - Write-allocate (load into cache, update line in cache)
    - Good if more writes to the location follow
  - No-write-allocate (writes straight to memory, no loading into cache)
- Typical
  - Write-through + No-write-allocate
  - **Write-back + Write-allocate** (current trend + symmetric to reads)

Intel Core i7 Cache Hierarchy (a real one)



L1 i-cache and d-cache:

32 KB, 8-way,

Access: 4 cycles

L2 unified cache:

256 KB, 8-way,

Access: 10 cycles

L3 unified cache:

8 MB, 16-way,

Access: 40-75 cycles

Block size: 64 bytes for all caches. (b = 6 bits)

All SRAM caches contained in CPU chip