Cache Memory and Performance

Many of the following slides are taken with permission from

Complete Powerpoint Lecture Notes for
Computer Systems: A Programmer's Perspective (CS:APP)

Randal E. Bryant ***and*** *David R. O'Hallaron*

http://csapp.cs.cmu.edu/public/lectures.html

The book is used explicitly in CS 2505 and CS 3214 and as a reference in CS 2506.

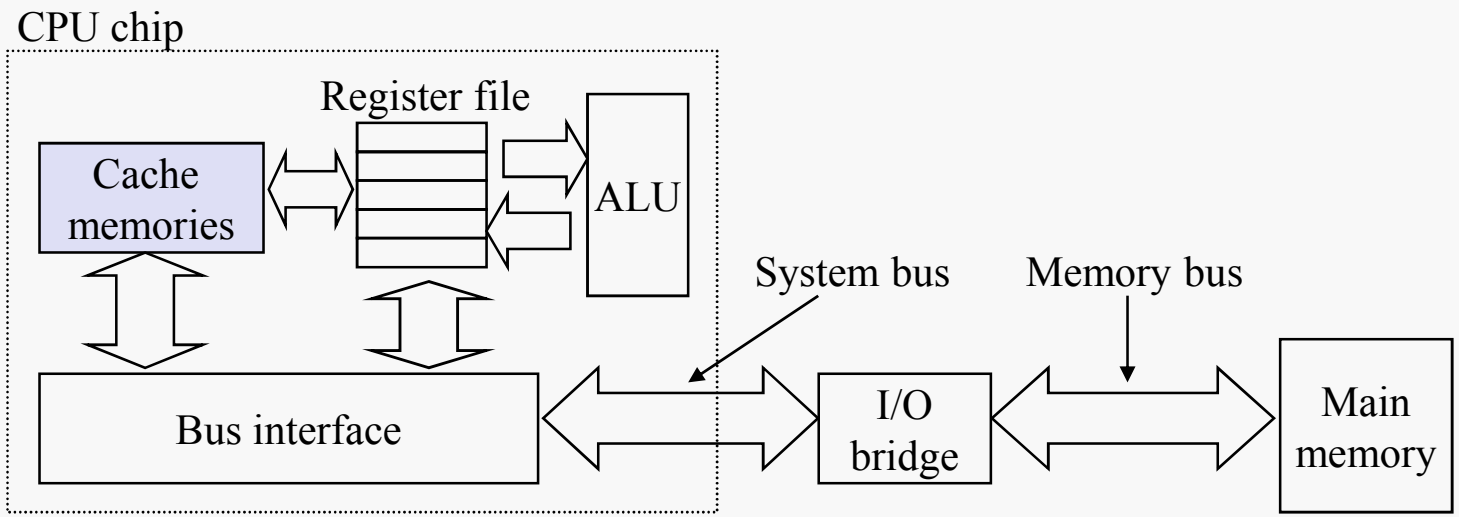
Cache Memories

Cache memories are small, fast SRAM-based memories managed automatically in hardware.

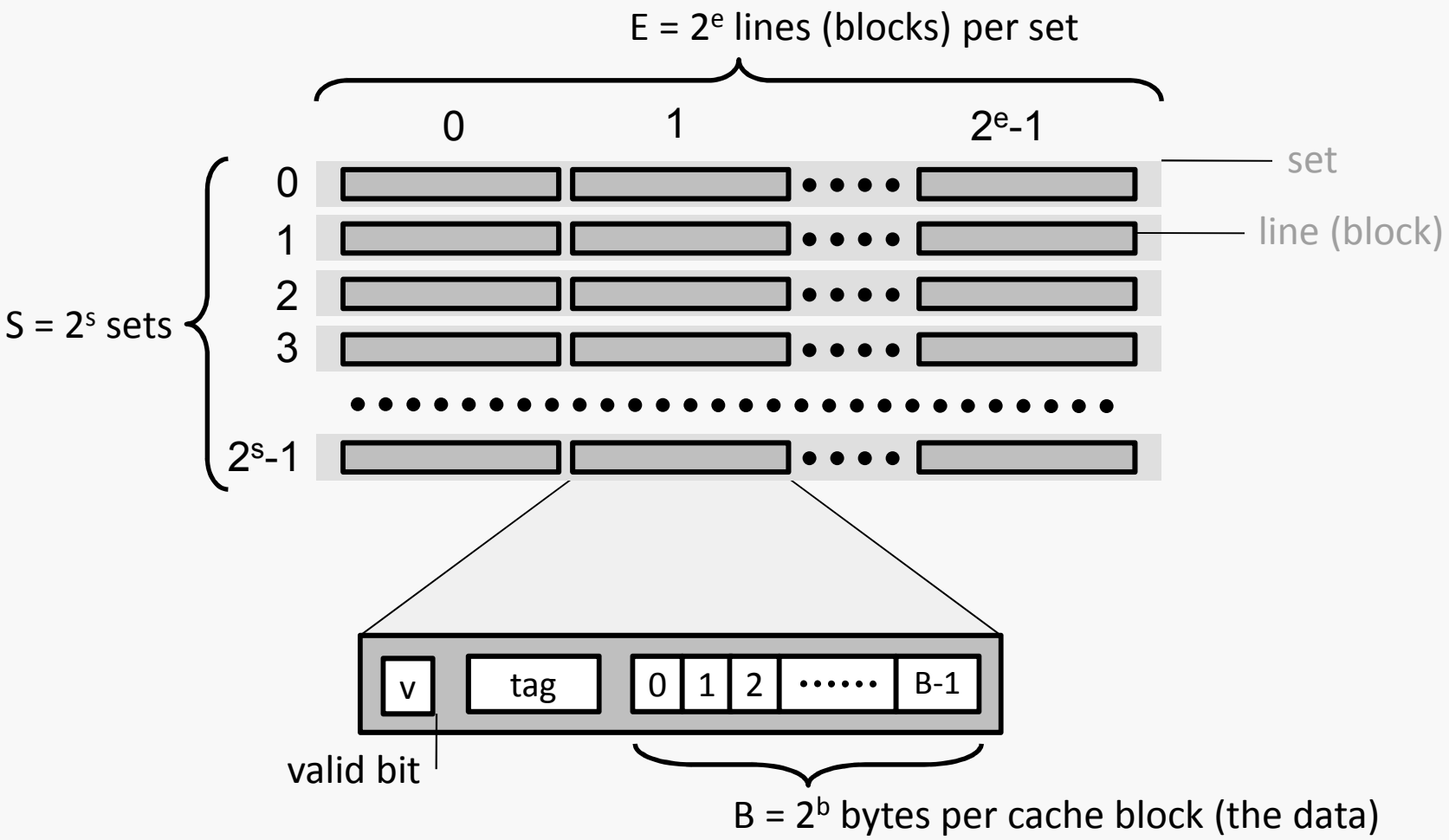
- Hold frequently accessed blocks of main memory

CPU looks first for data in caches (e.g., L1, L2, and L3), then in main memory.

Typical system structure:



General Cache Organization (S, E, B)



Cache size:

C = S x E x B data bytes

Cache Defines View of DRAM

The "geometry" of the cache is defined by:

$S = 2^s$ the number of sets in the cache
$E = 2^e$ the number of lines (blocks) in a set
$B = 2^b$ the number of bytes in a line (block)

These values define a related way to think about the organization of DRAM:

DRAM consists of a sequence of blocks of B bytes.
The bytes in a block (line) can be indexed by using b bits.

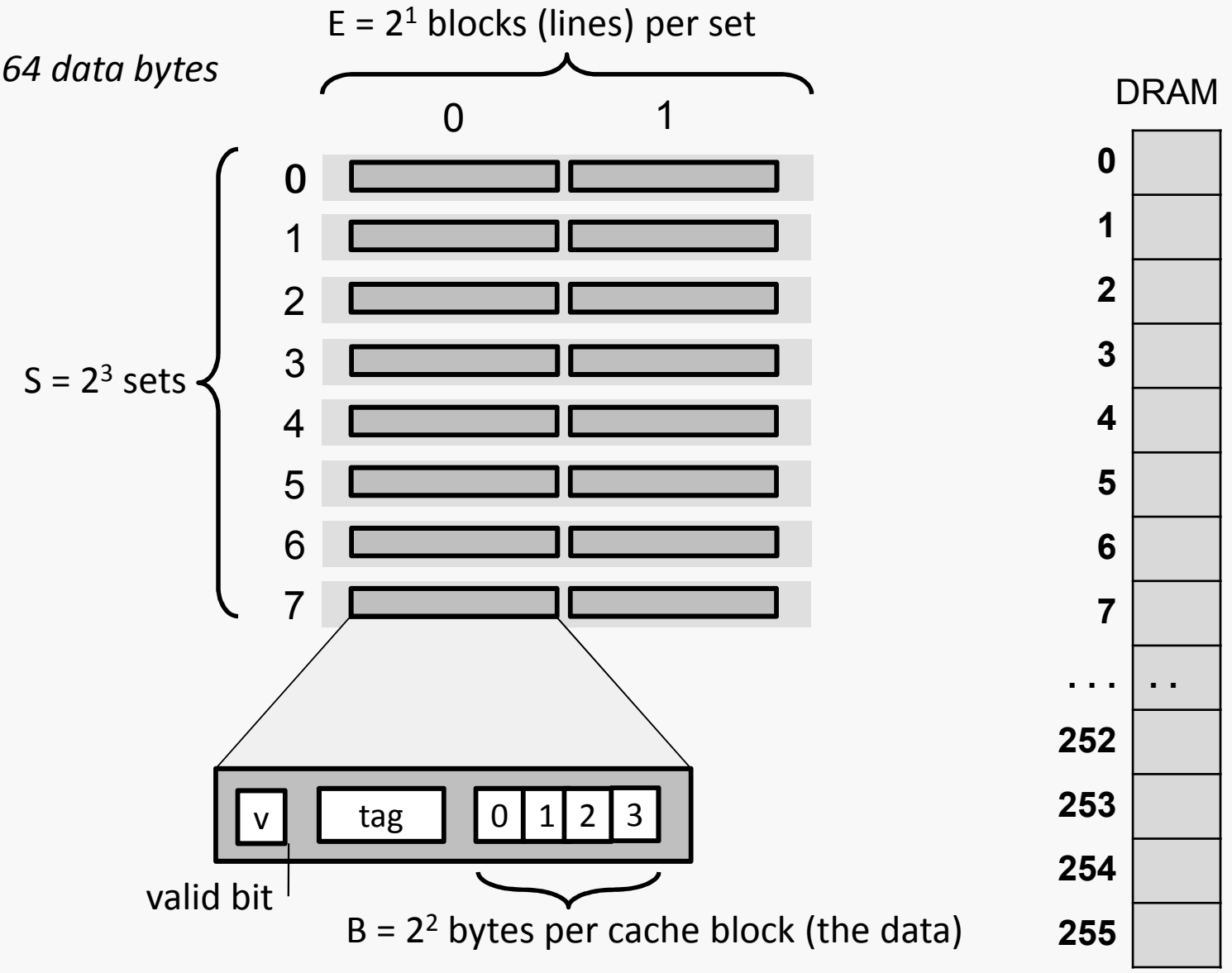
DRAM consists of a sequence of groups of S blocks (lines).
The blocks (lines) in a group can be indexed by using s bits.

Each group contains SxB bytes, which can be indexed by using s + b bits.

Cache (8, 2, 4) and 256-Byte DRAM

Cache size:

C = S x E x B = 64 data bytes



Example of Cache View of DRAM

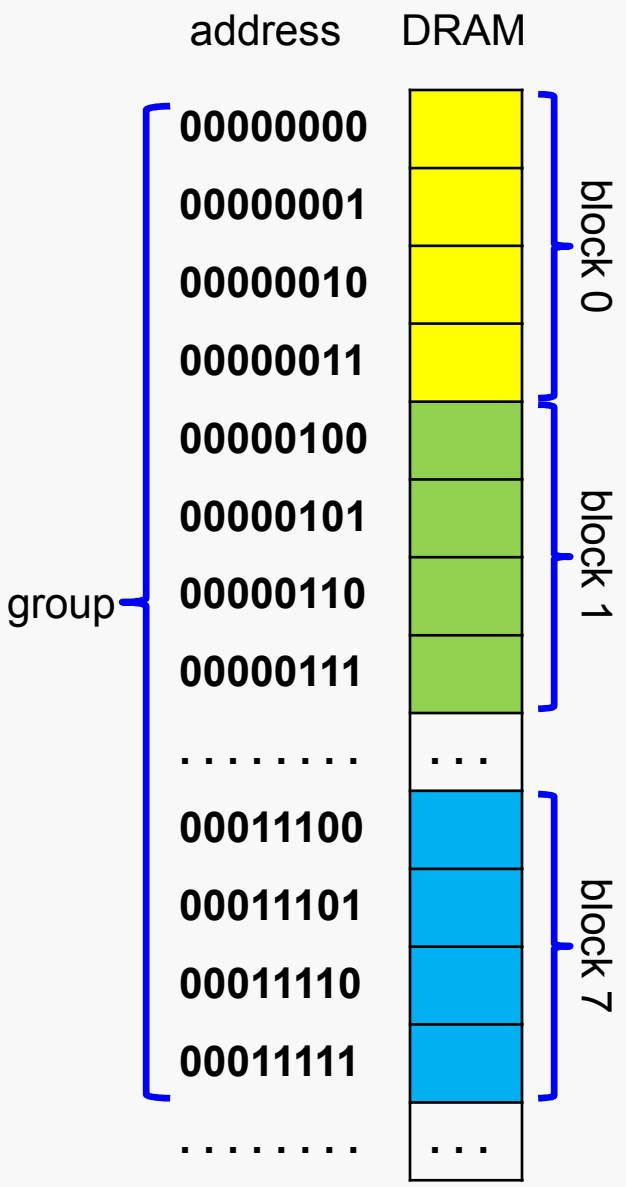
Assume a cache has the following geometry:

$S = 2^2 = 8$ the number of sets in the cache
$E = 2^1 = 2$ the number of lines (blocks) in a set
$B = 2^2 = 4$ the number of bytes in a line (block)

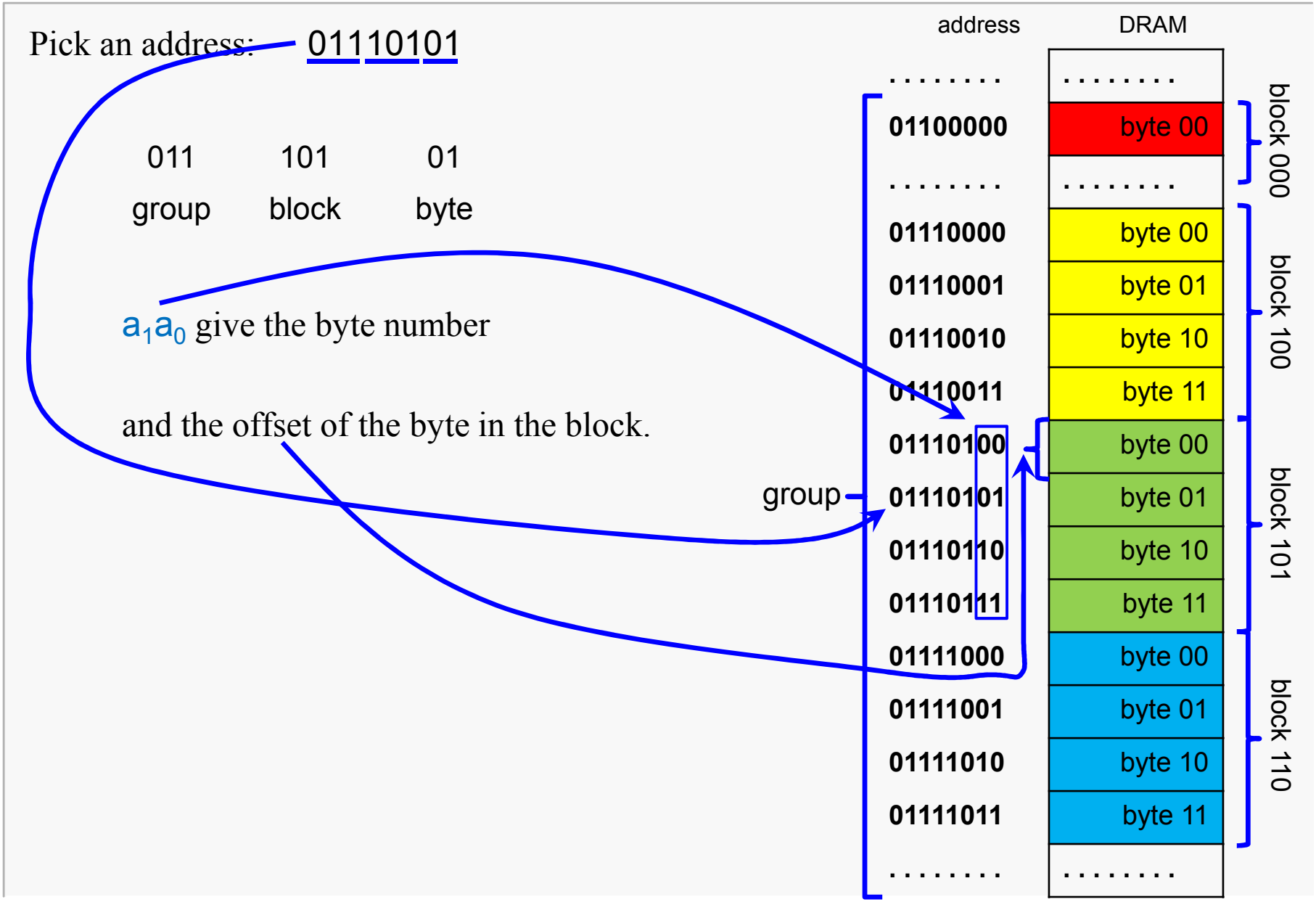
Suppose that DRAM consists of 256 bytes,
so we have 8-bit addresses.

Then DRAM consists of:

- 64 blocks, each holding 4 bytes
- 8 groups, each holding 8 blocks



Example of Cache View of DRAM



Example of Cache View of DRAM

Pick an address: 01110101

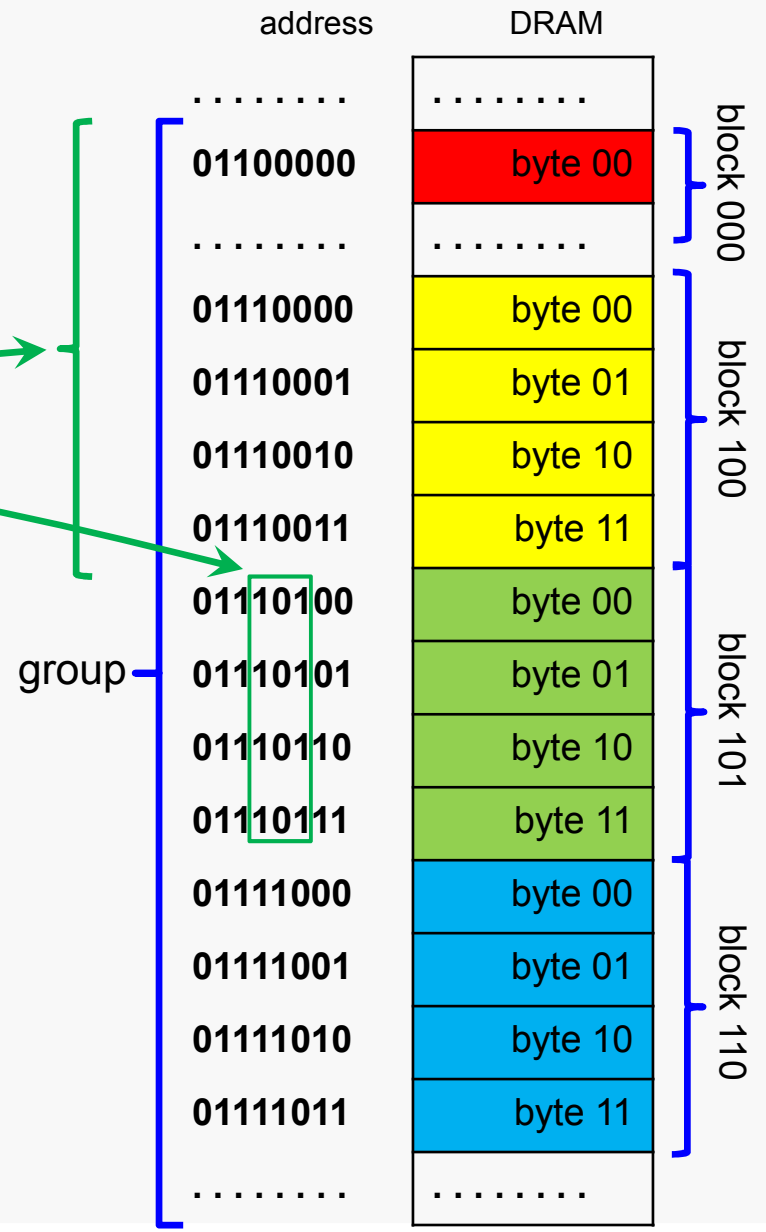
011	101	01
group	block	byte

$a_4a_3a_2$ give the block number

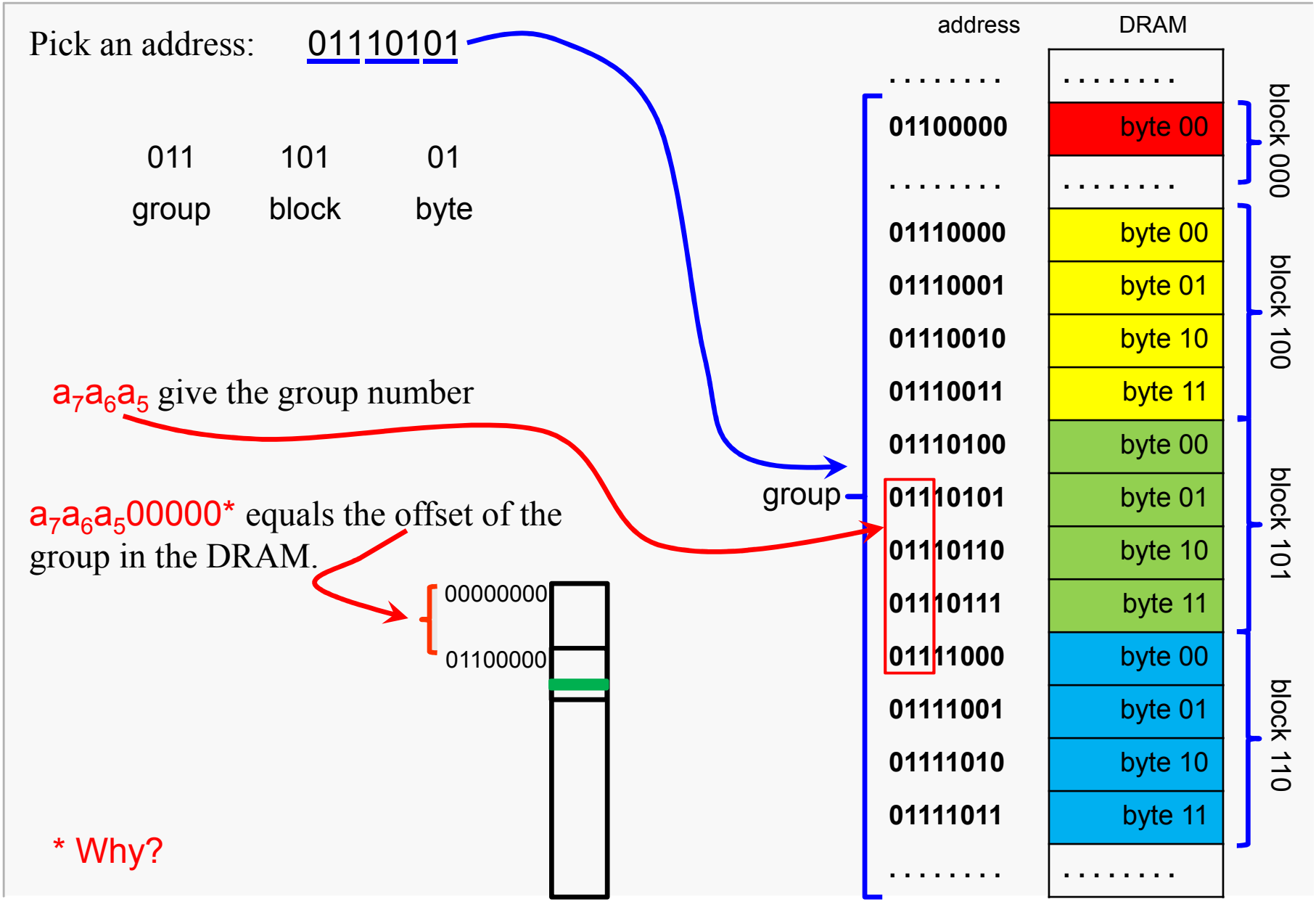
$a_4a_3a_200$* equals the offset of the block in the group.

* $a_4a_3a_200 = a_4a_3a_2 \times 2^2$

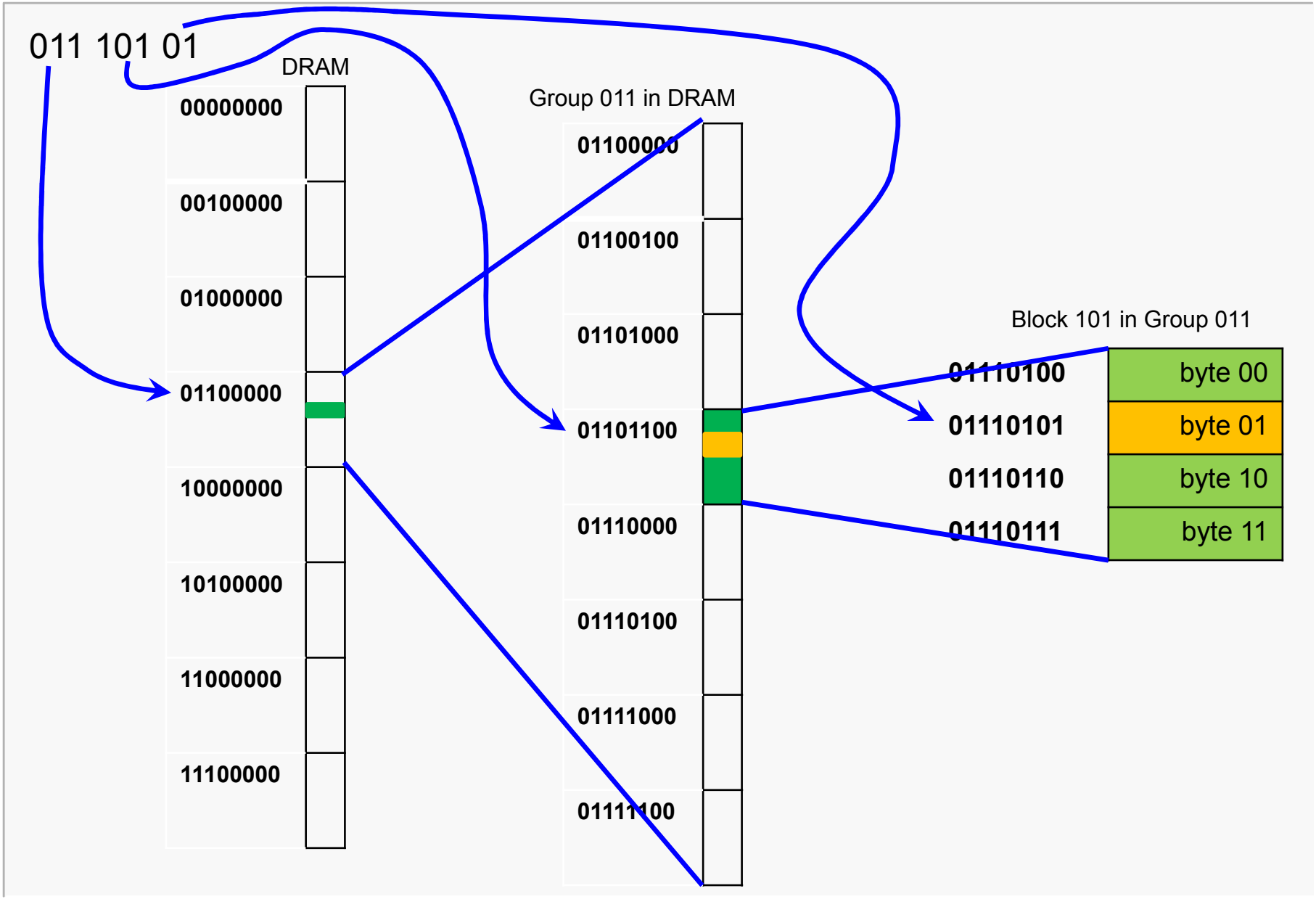
$= a_4a_3a_2 \times$ (size of a block)



Example of Cache View of DRAM



The BIG Picture



Example of Cache View of DRAM

Pick an address: 01110101

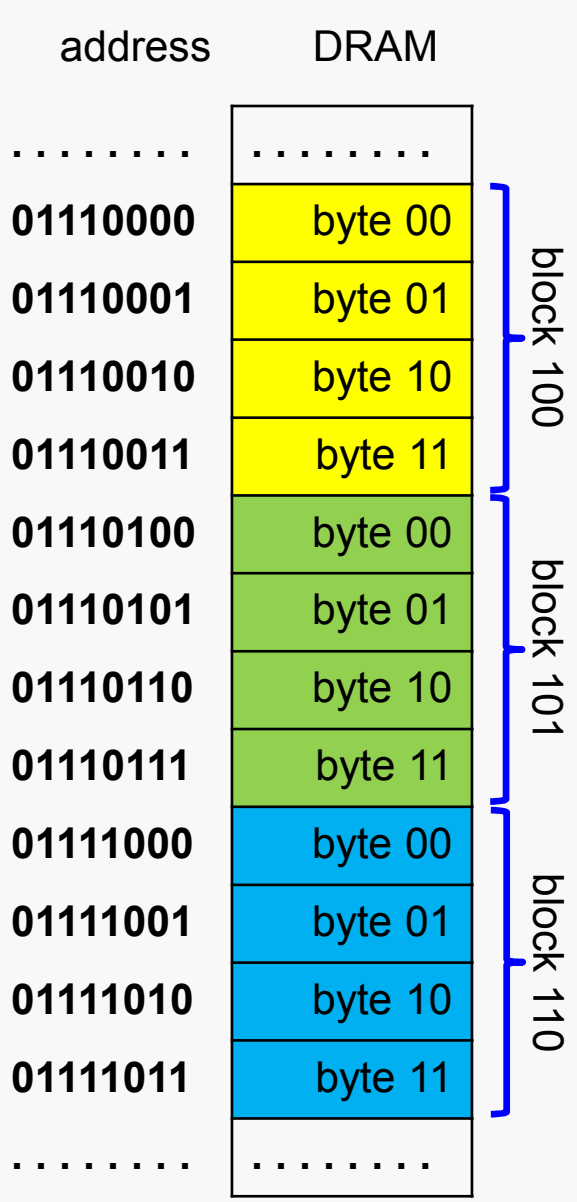
How does this address map into the cache?

The DRAM block number determines the cache set used to store the block.

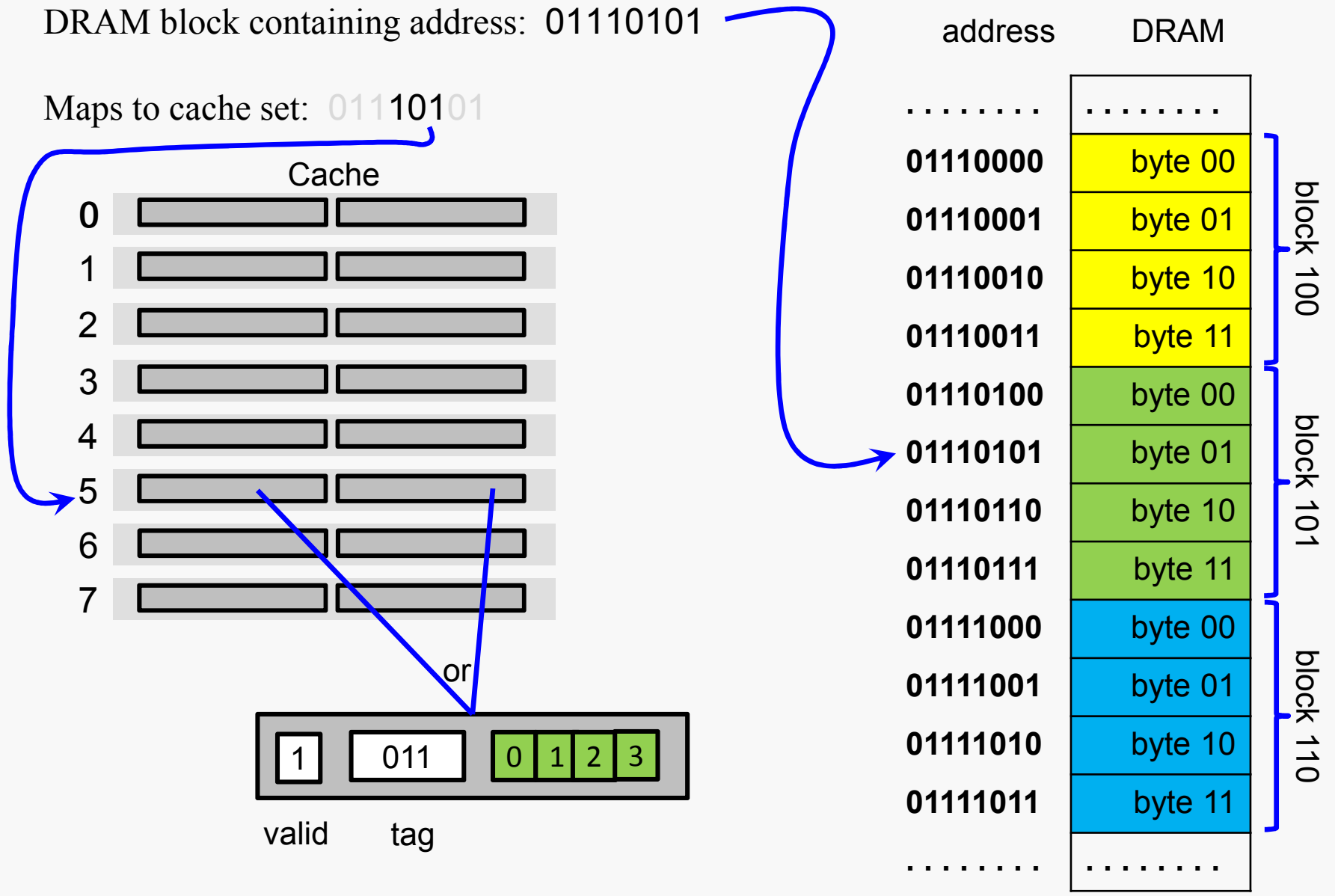
Note this means that two DRAM blocks from the same DRAM group cannot map into the same cache set.

So the address: 01110101 maps to set 101 in the cache.

Each set in our cache can hold 2 blocks. This block could be stored at either location within the corresponding set.



Example of Cache View of DRAM



Cache View of DRAM

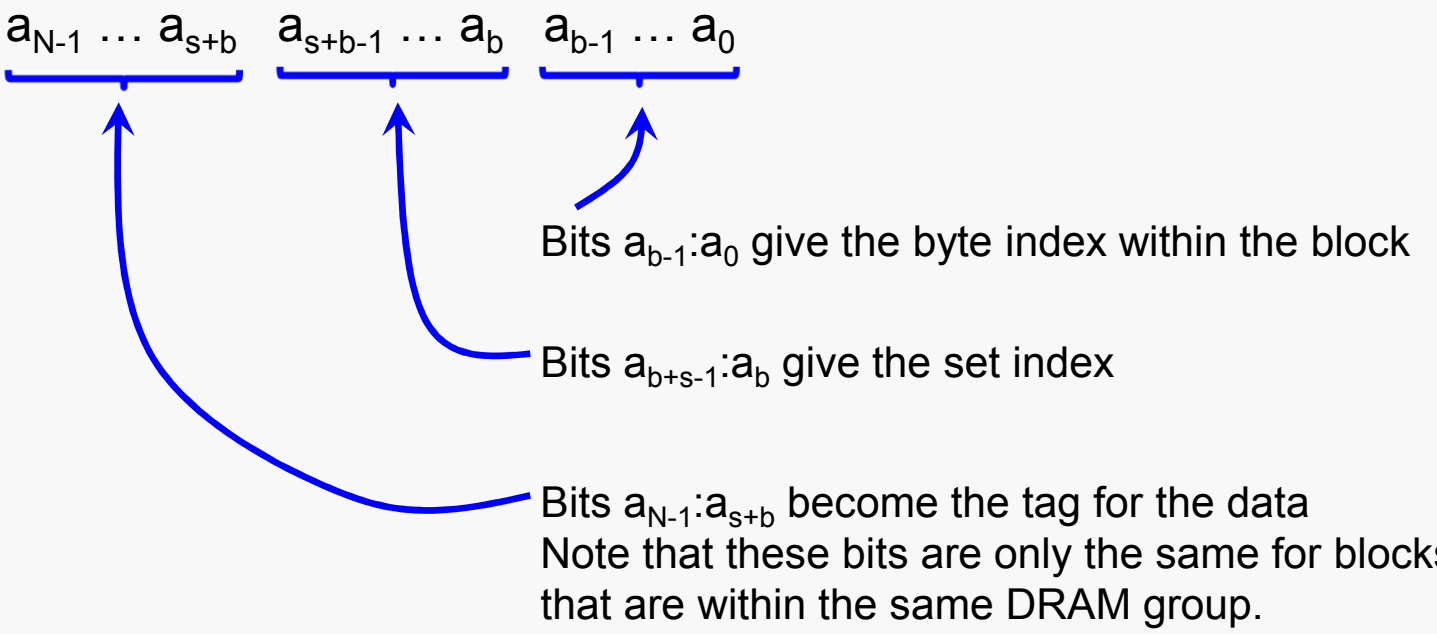
So, to generalize, suppose a cache has:

$S = 2^s$ sets

$E = 2^e$ blocks (lines) per set

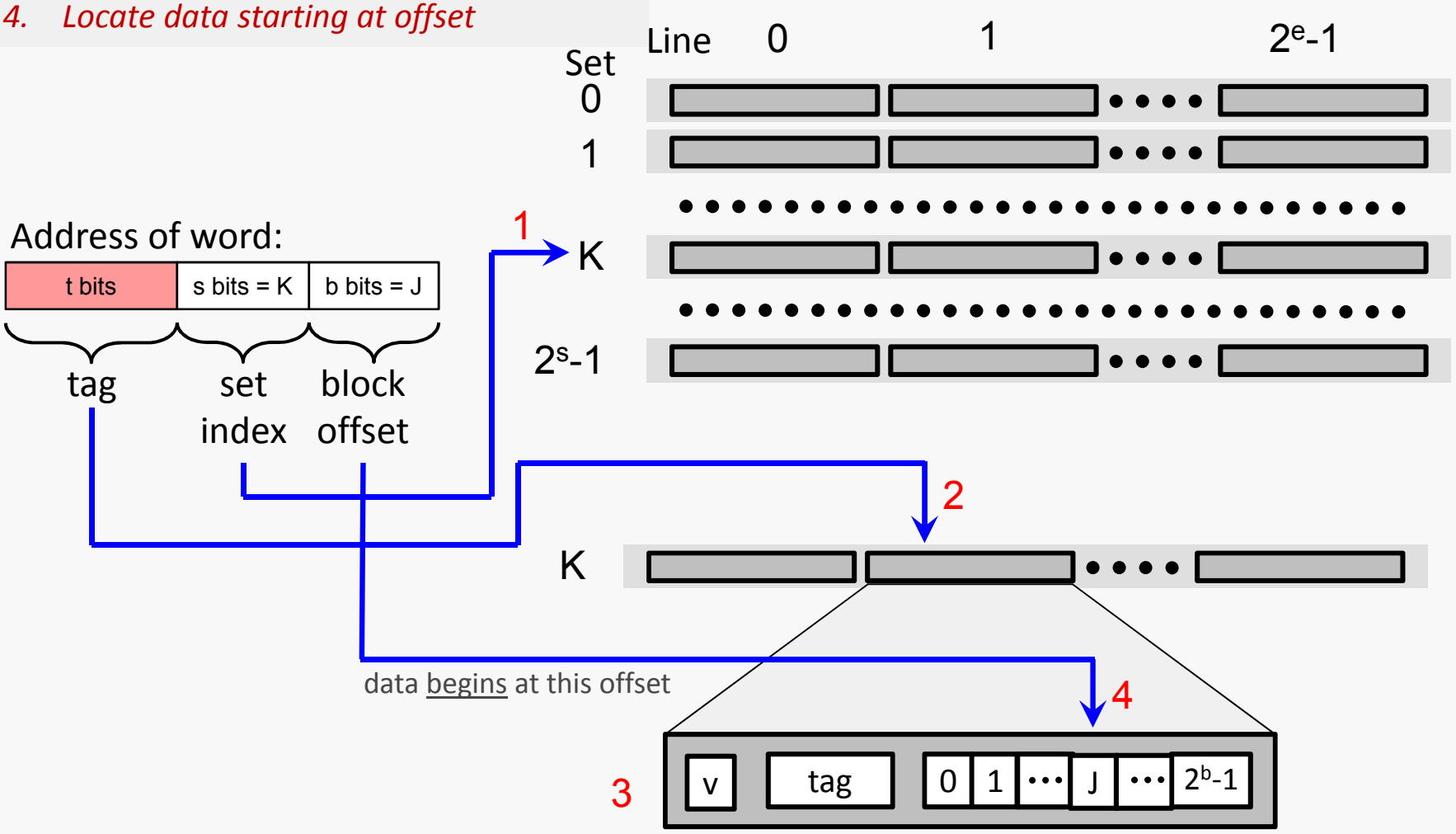
$B = 2^b$ bytes per block (line)

And, suppose that DRAM uses N-bit addresses. then for any address:



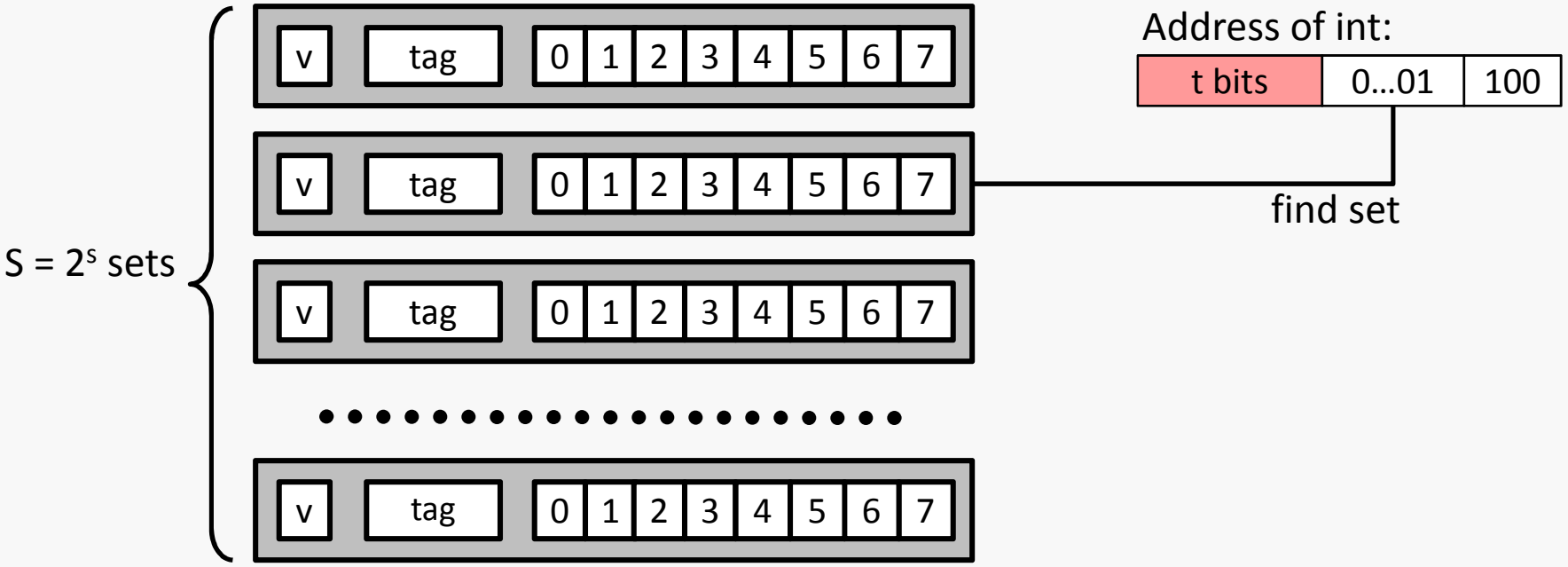
Cache Read

1. *Locate set*
2. *Check if any line in set has matching tag*
3. *Yes + line valid: hit*
4. *Locate data starting at offset*



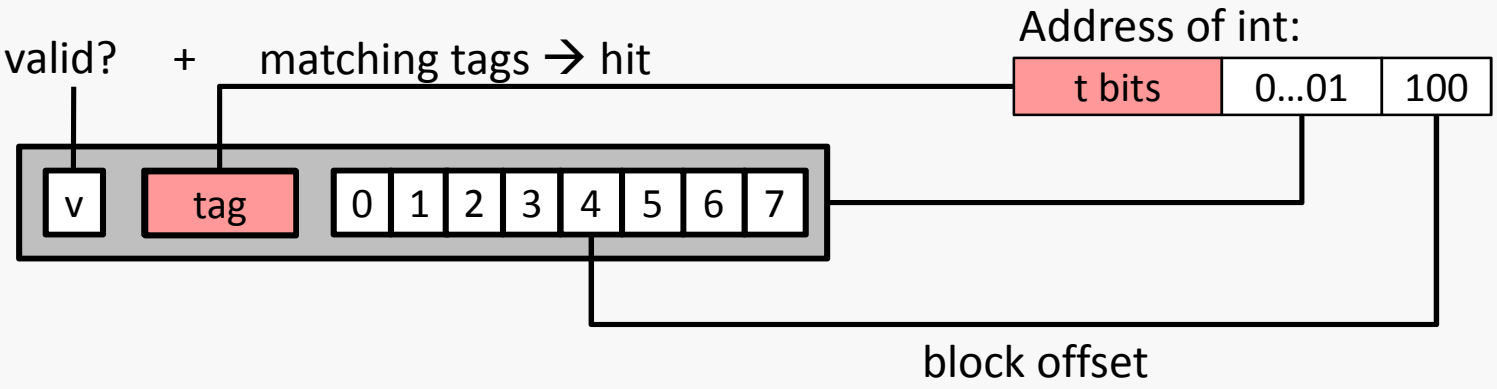
Example: Direct Mapped Cache (E = 1)

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



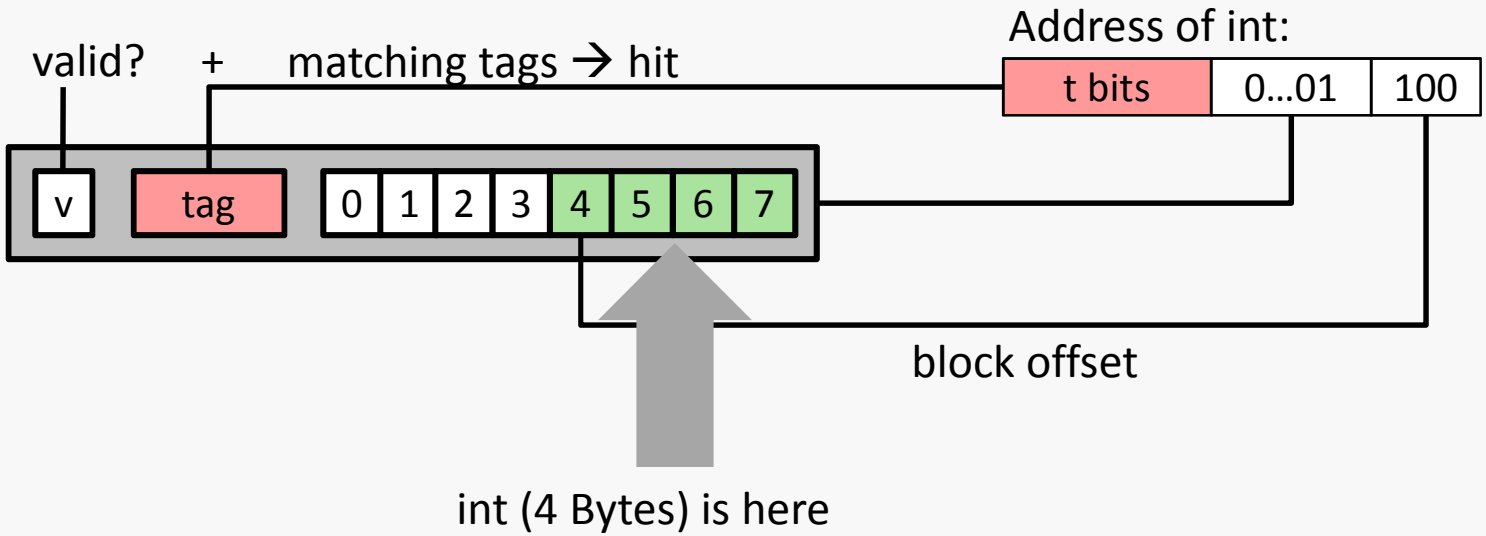
Example: Direct Mapped Cache (E = 1)

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 (block) is evicted and replaced by requested block from DRAM

Direct-Mapped Cache Simulation

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

M=16 byte addresses, B=2 bytes/block,
S=4 sets, E=1 Blocks/set

Address trace (reads, one byte per read):

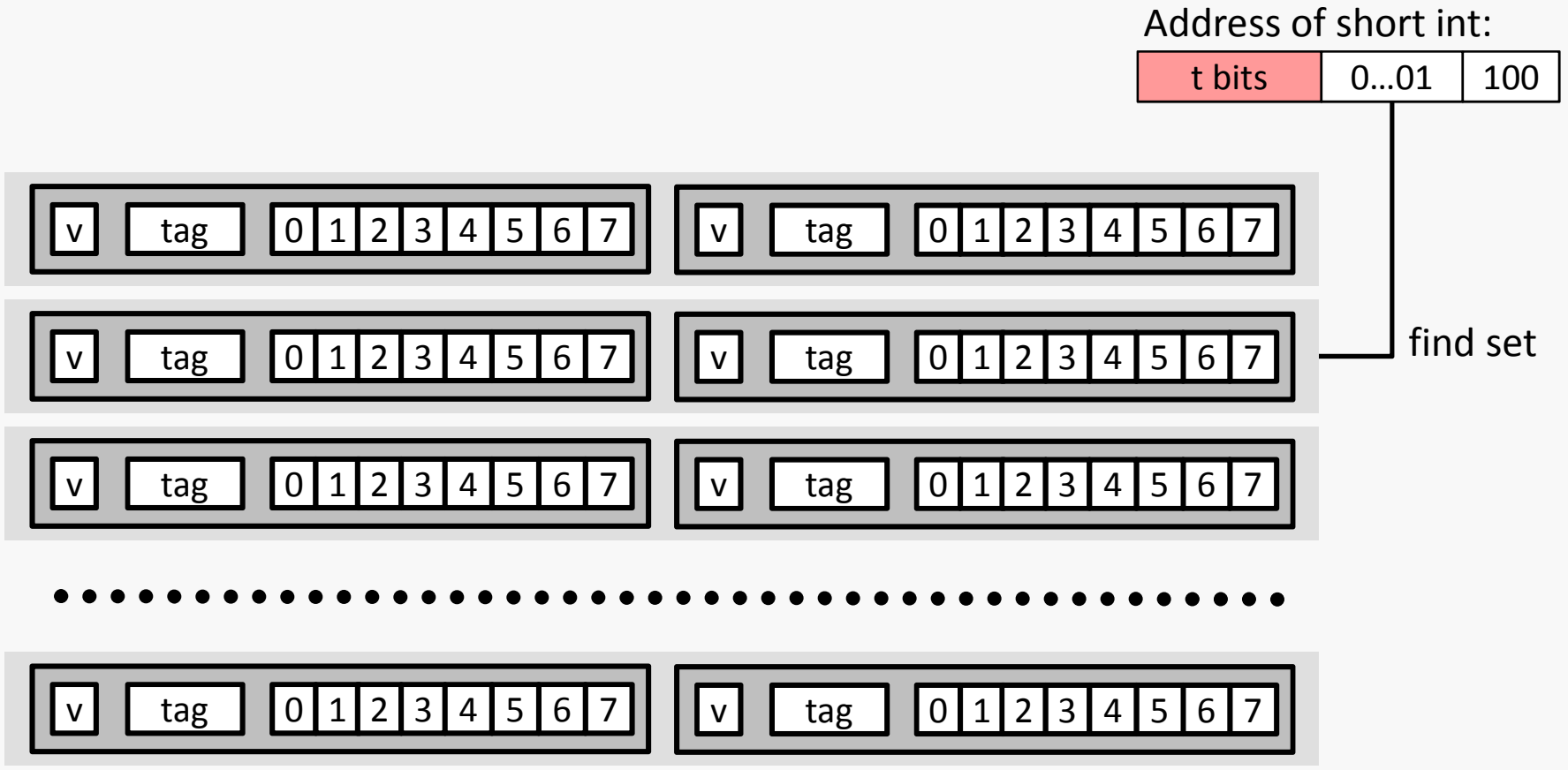
0	[$0\underline{00}0_2$],	miss
1	[$0\underline{00}1_2$],	hit
7	[$0\underline{11}1_2$],	miss
8	[$1\underline{00}0_2$],	miss
0	[$0\underline{00}0_2$]	miss

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

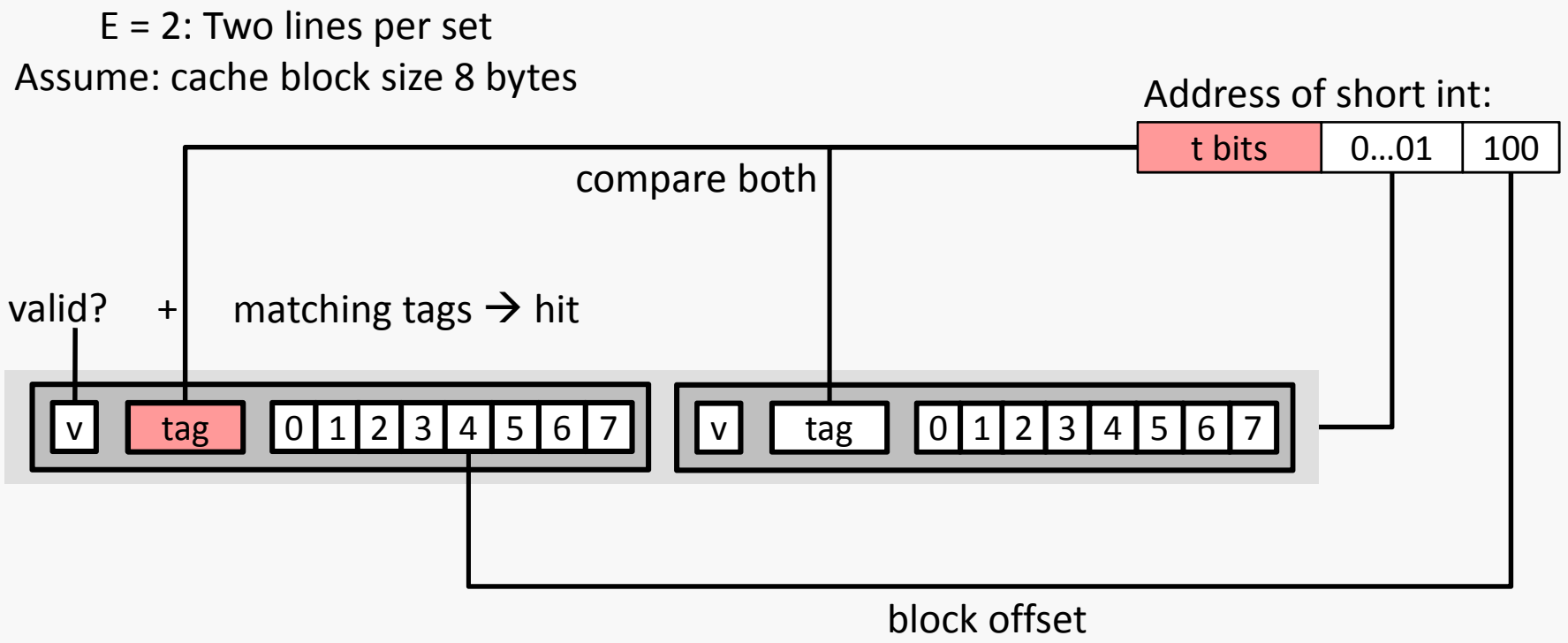
E-way Set Associative Cache (Here: E = 2)

E = 2: Two lines per set

Assume: cache block size 8 bytes



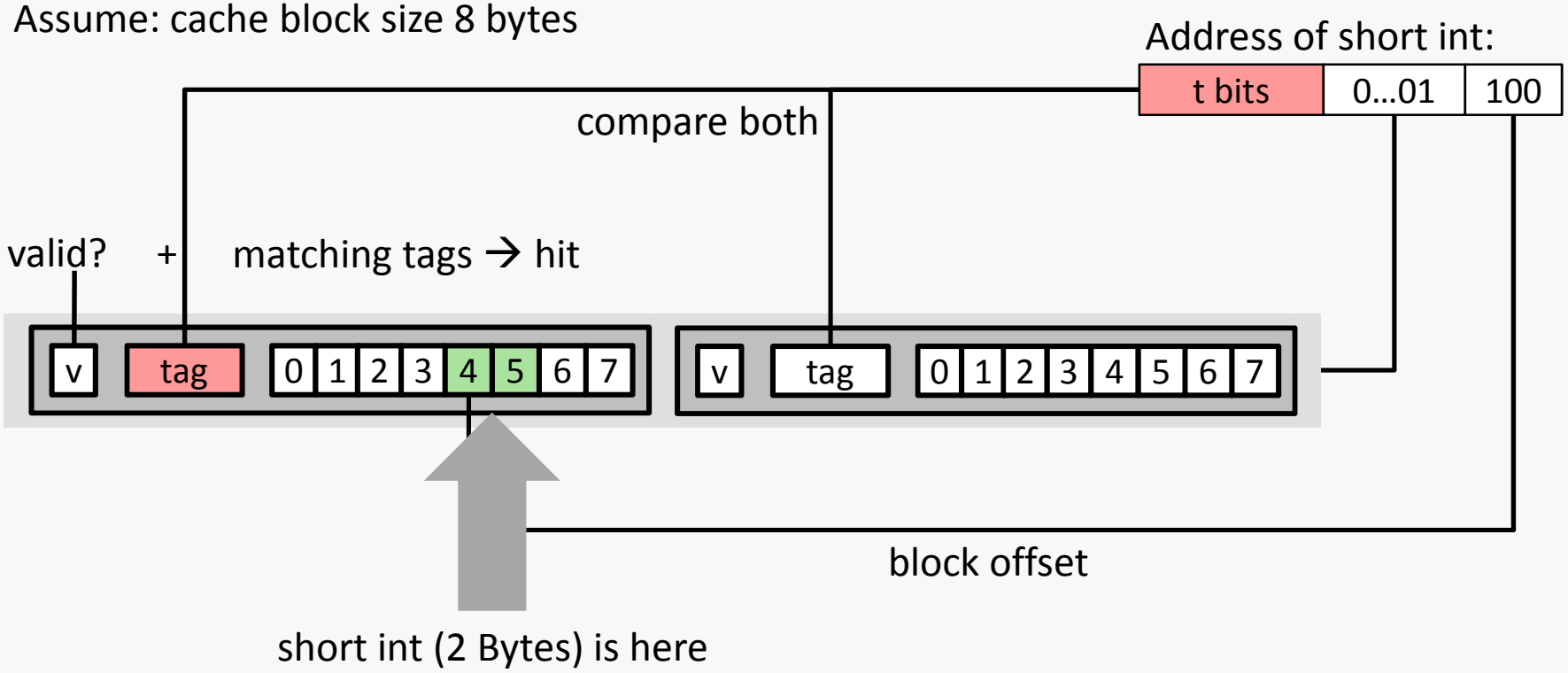
E-way Set Associative Cache (Here: E = 2)



E-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), …

2-Way Set Associative Cache Simulation

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

M=16 byte addresses, B=2 bytes/block,
S=2 sets, E=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		

Cache Organization Types

The "geometry" of the cache is defined by:

$S = 2^s$ the number of sets in the cache
$E = 2^e$ the number of lines (blocks) in a set
$B = 2^b$ the number of bytes in a line (block)

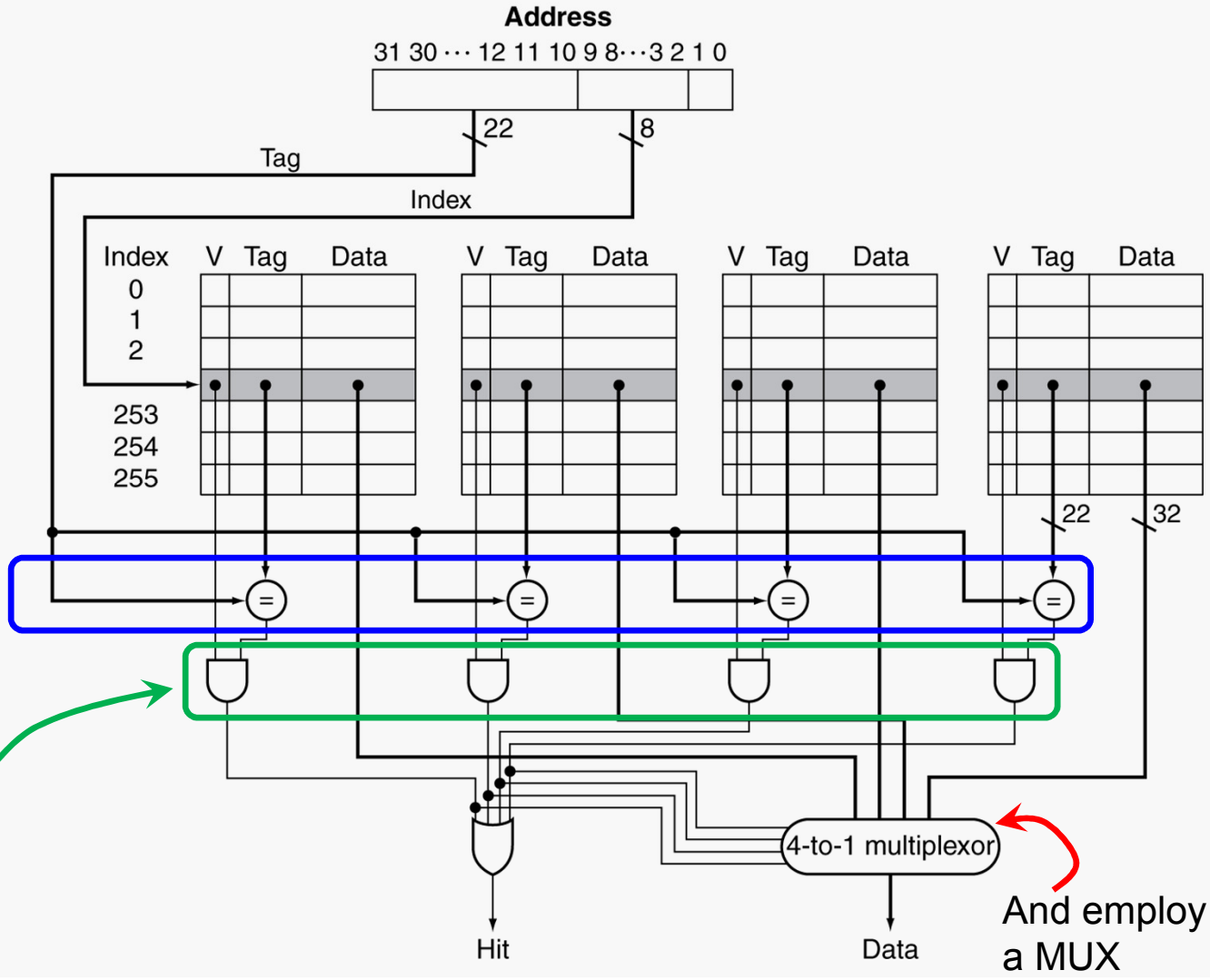
E = 1 (e = 0) *direct-mapped cache*
only one possible location in cache for each DRAM block

S > 1
E = K > 1 K-*way associative cache*
K possible locations (in same cache set) for each DRAM block

S = 1 (only one set) *fully-associative cache*
E = # of cache blocks each DRAM block can be at any location in the cache

Searching a Set

If we have an associative cache (K-way or fully), how do we determine if a given DRAM block occurs within a set?



Compare the tag we're trying to match to all of the tags for blocks in the relevant set at the same time!

Then factor in the valid bits, also in parallel.

And employ a MUX