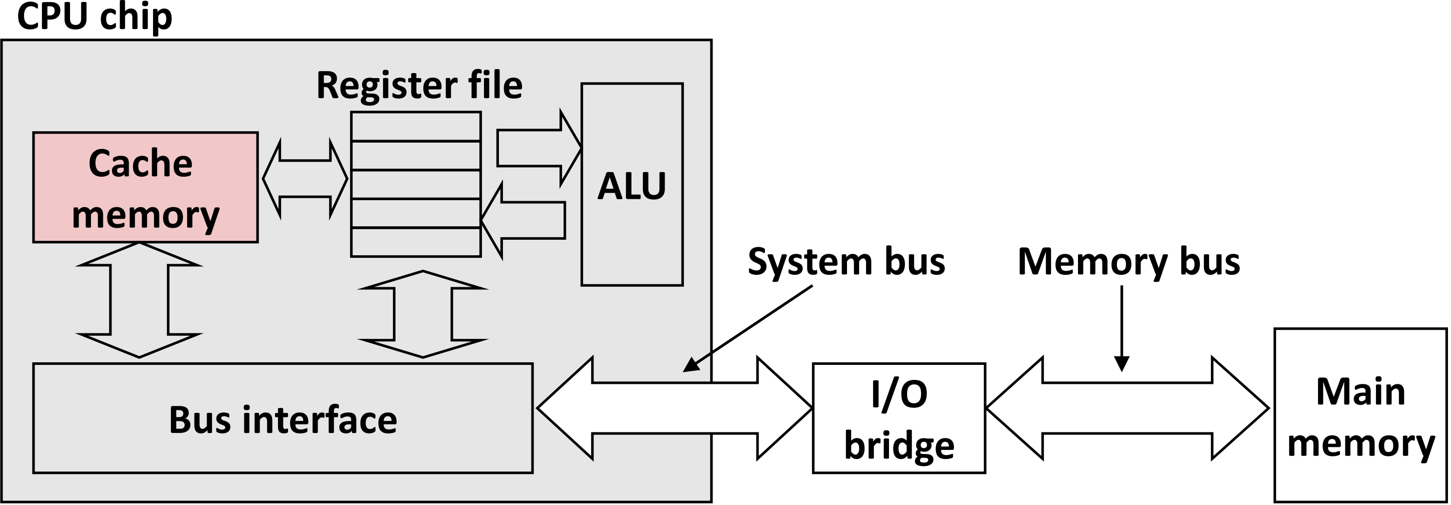
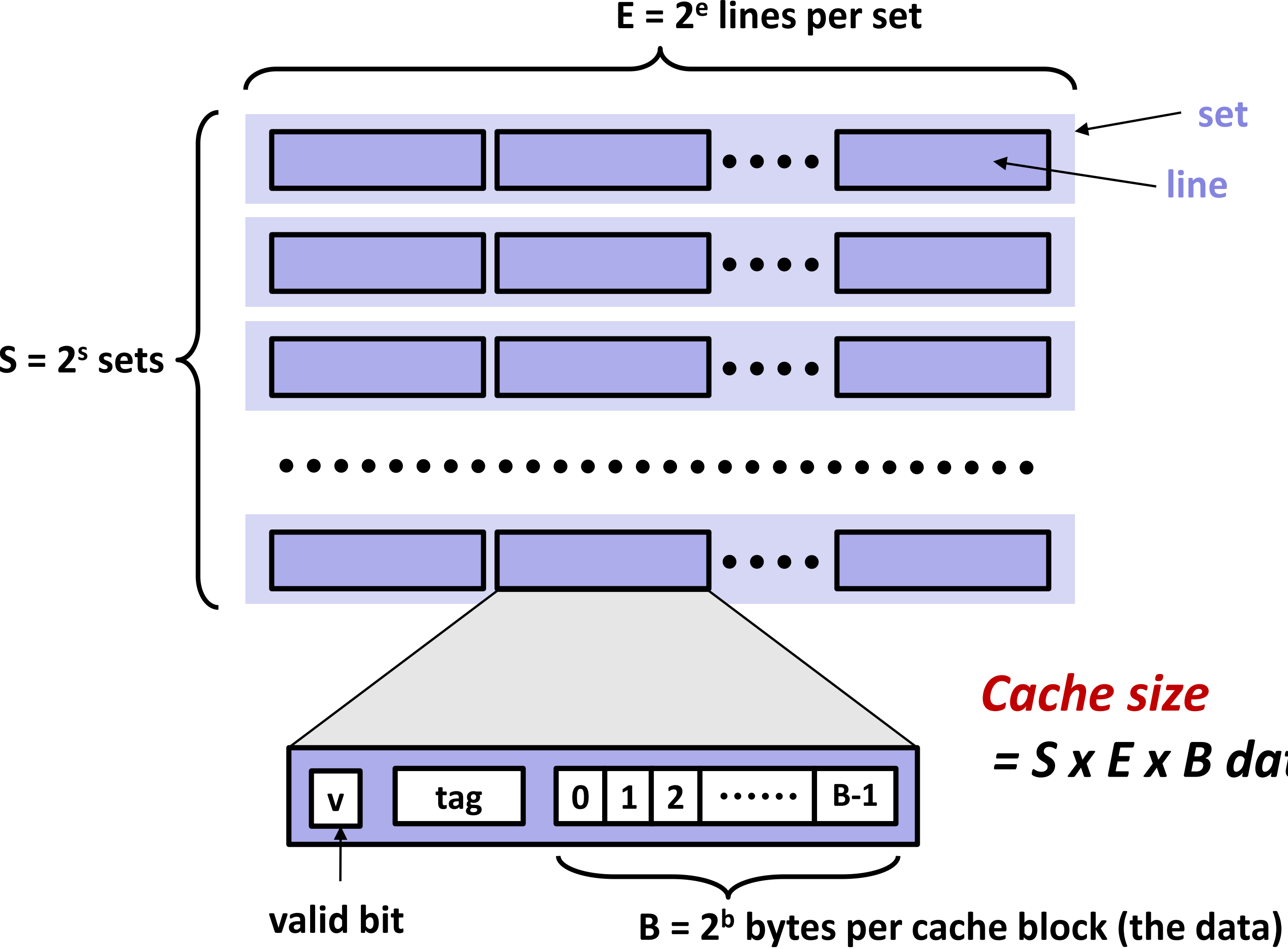

CPU chip
Register file
Cache memory
ALU
System bus
Memory bus
Bus interface
I/O bridge
Main memory



E = 2^e lines per set

set

line

S = 2^s sets

v

tag

0

1

2

······

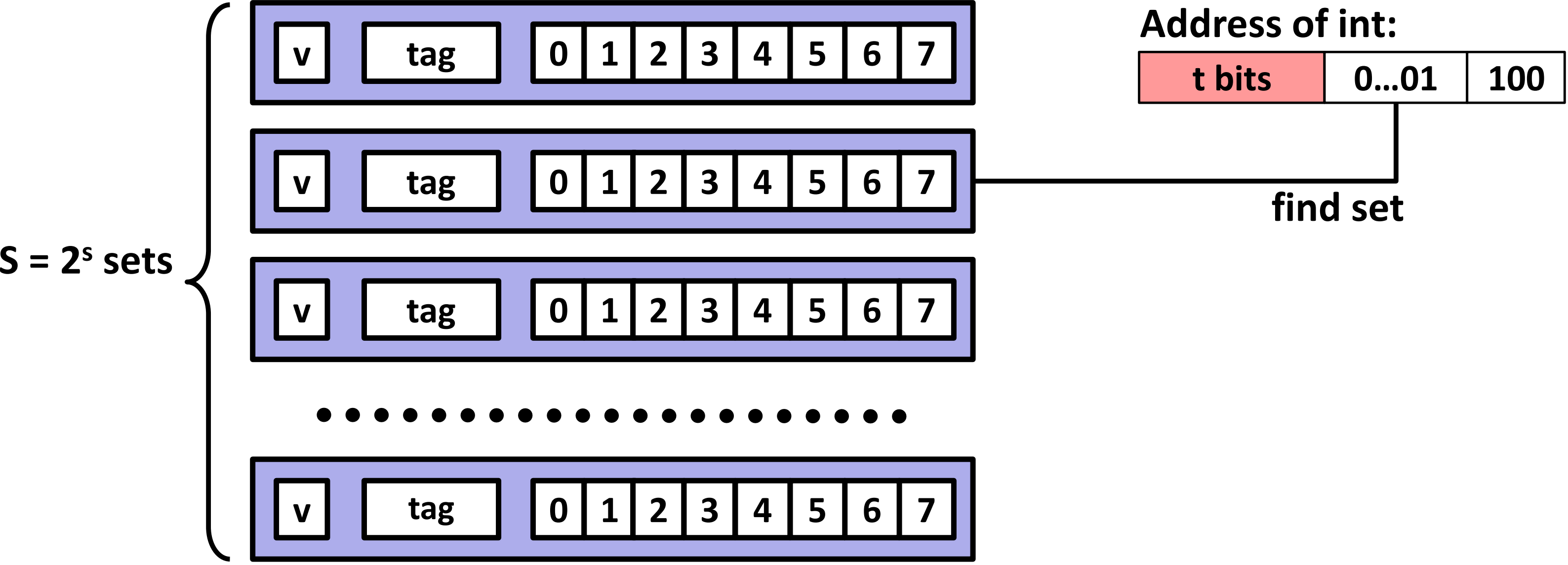
B-1

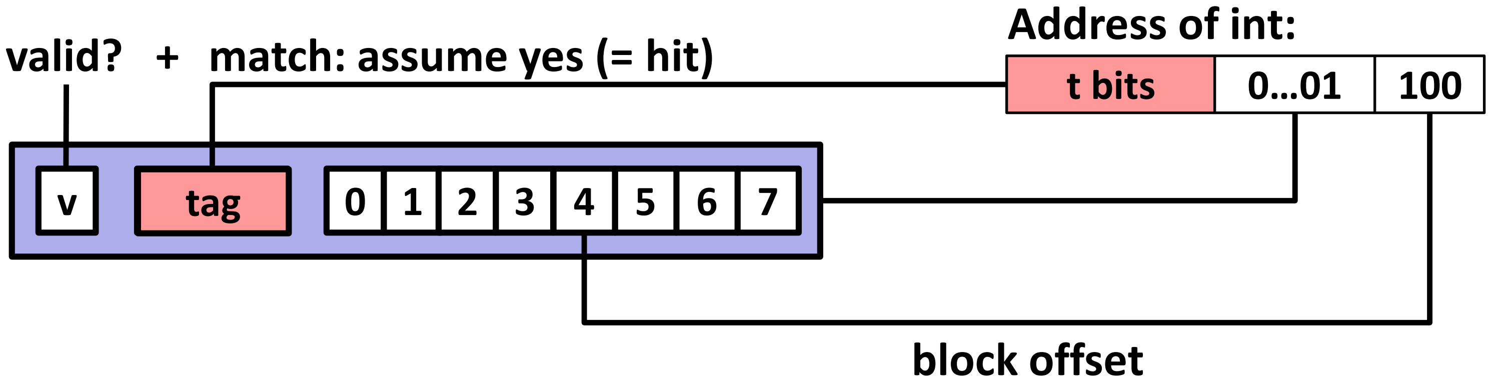
valid bit

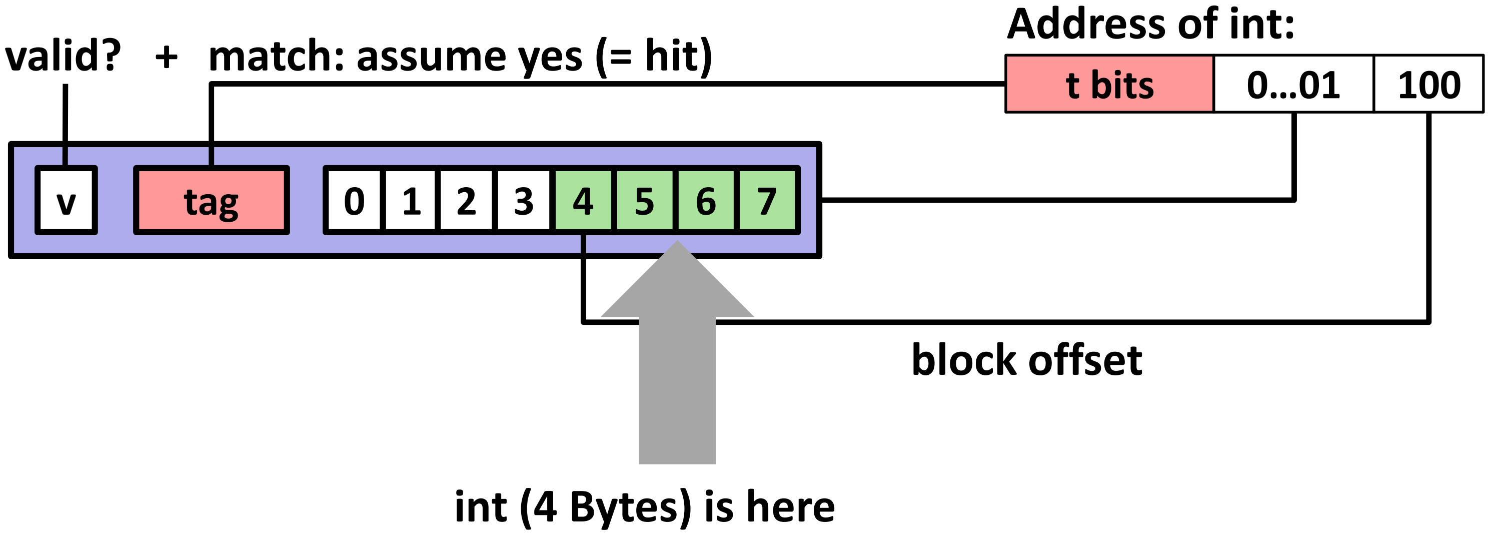
B = 2^b bytes per cache block (the data)

Cache size

= S x E x B data bytes


Address of int:
t bits
0…01
100
find set
S = 2^s sets
v
tag
0 1 2 3 4 5 6 7


Address of int:
valid? + match: assume yes (= hit)
t bits
0…01
100
v
tag
0 1 2 3 4 5 6 7
block offset


Address of int:
valid? + match: assume yes (= hit)
t bits
0…01
100
v
tag
0 1 2 3 4 5 6 7
block offset
int (4 Bytes) is here

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

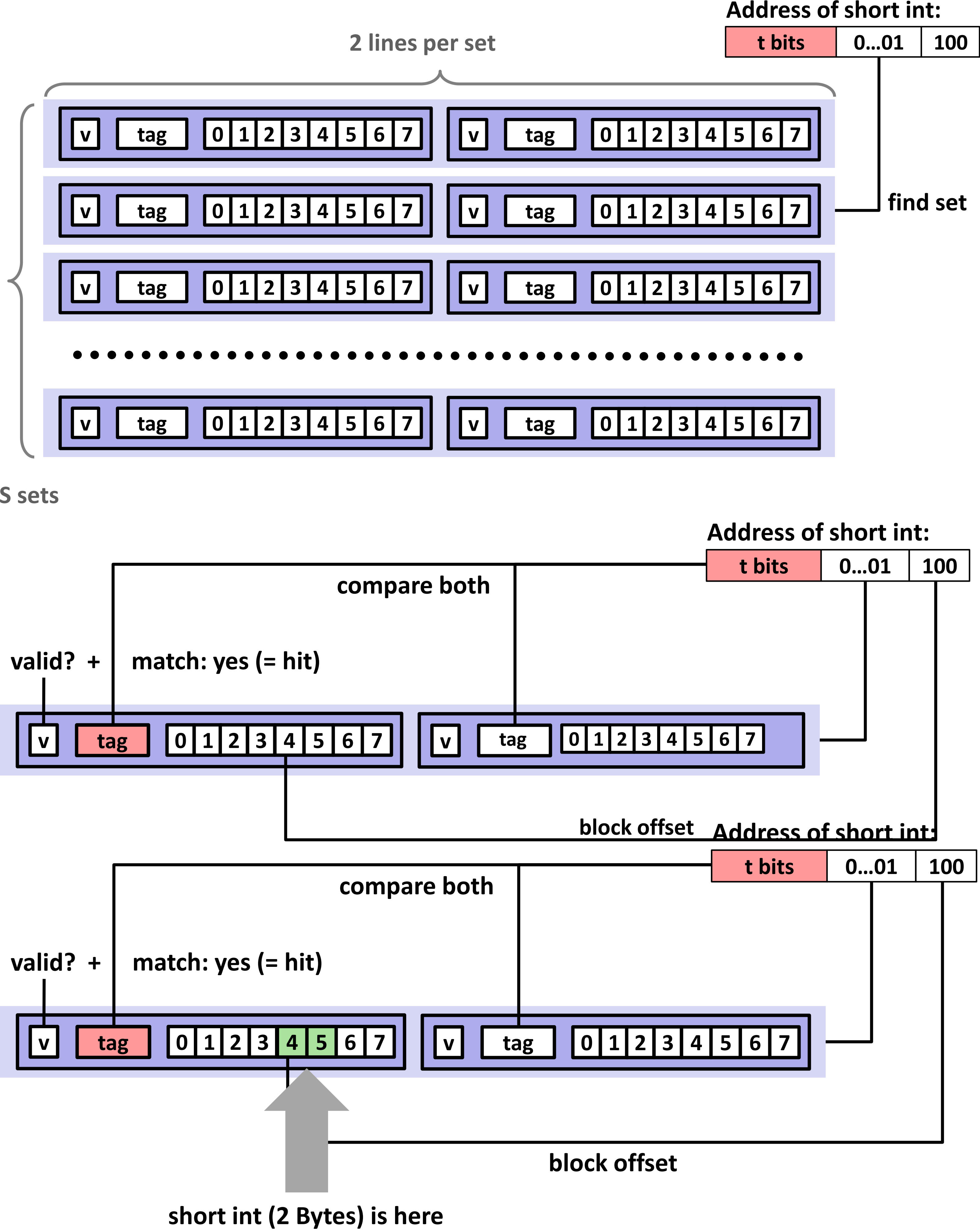
4-bit addresses (address space size M=16 bytes)

S=4 sets, E=1 Blocks/set, B=2 bytes/block

Address trace (reads, one byte per read):

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

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


Address of short int:
t bits
0…01
100
2 lines per set
v
tag
0 1 2 3 4 5 6 7
find set
S sets
Address of short int:
t bits
0…01
100
compare both
valid? +
match: yes (= hit)
block offset
Address of short int:
t bits
0…01
100
compare both
valid? +
match: yes (= hit)
block offset
short int (2 Bytes) is here

2-Way Set Associative Cache Simulation

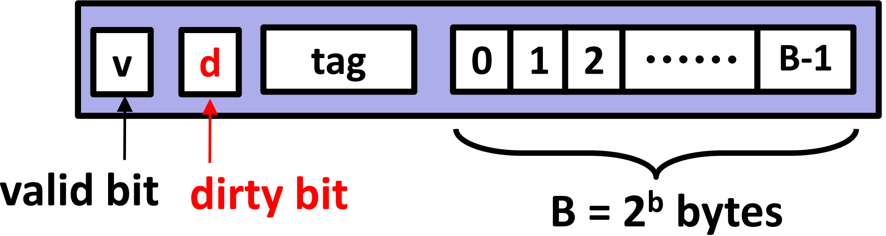
t=2	s=1	b=1
xx	x	x

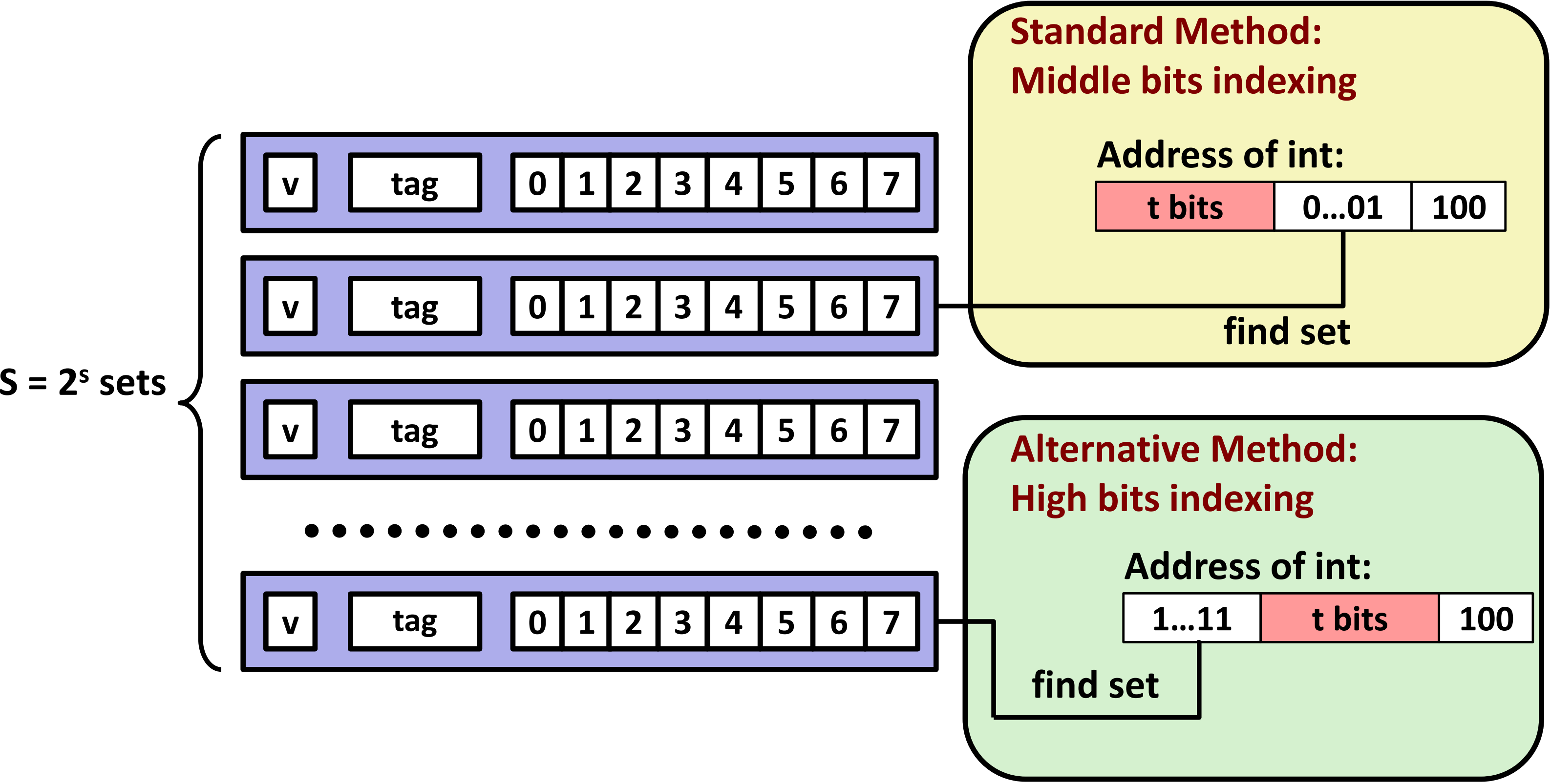
4-bit addresses (M=16 bytes)
S=2 sets, E=2 blocks/set, B=2 bytes/block

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		


v
d
tag
0
1
2
• • • • • •
B-1
valid bit
dirty bit
B = 2^b bytes



Standard Method:
Middle bits indexing

Address of int:

t bits	0…01	100

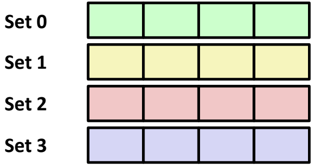
Alternative Method:
High bits indexing

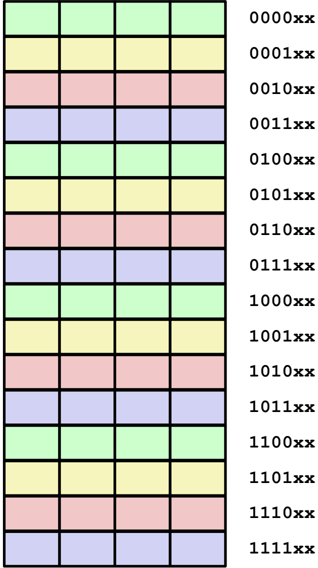
Address of int:

1…11	t bits	100

Middle Bits Indexing

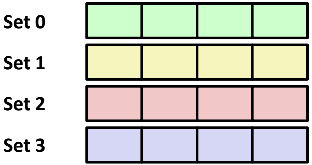
- Addresses of form TTSSBB
  - TT Tag bits
  - SS Set index bits
  - BB Offset bits
- Makes good use of spatial locality

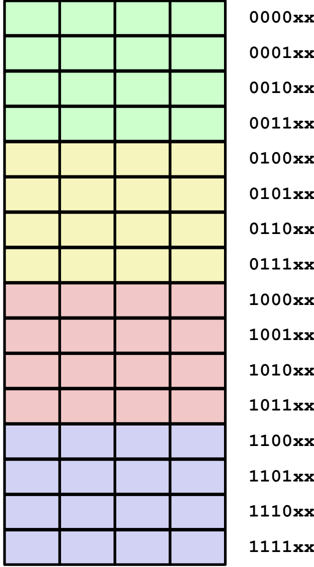


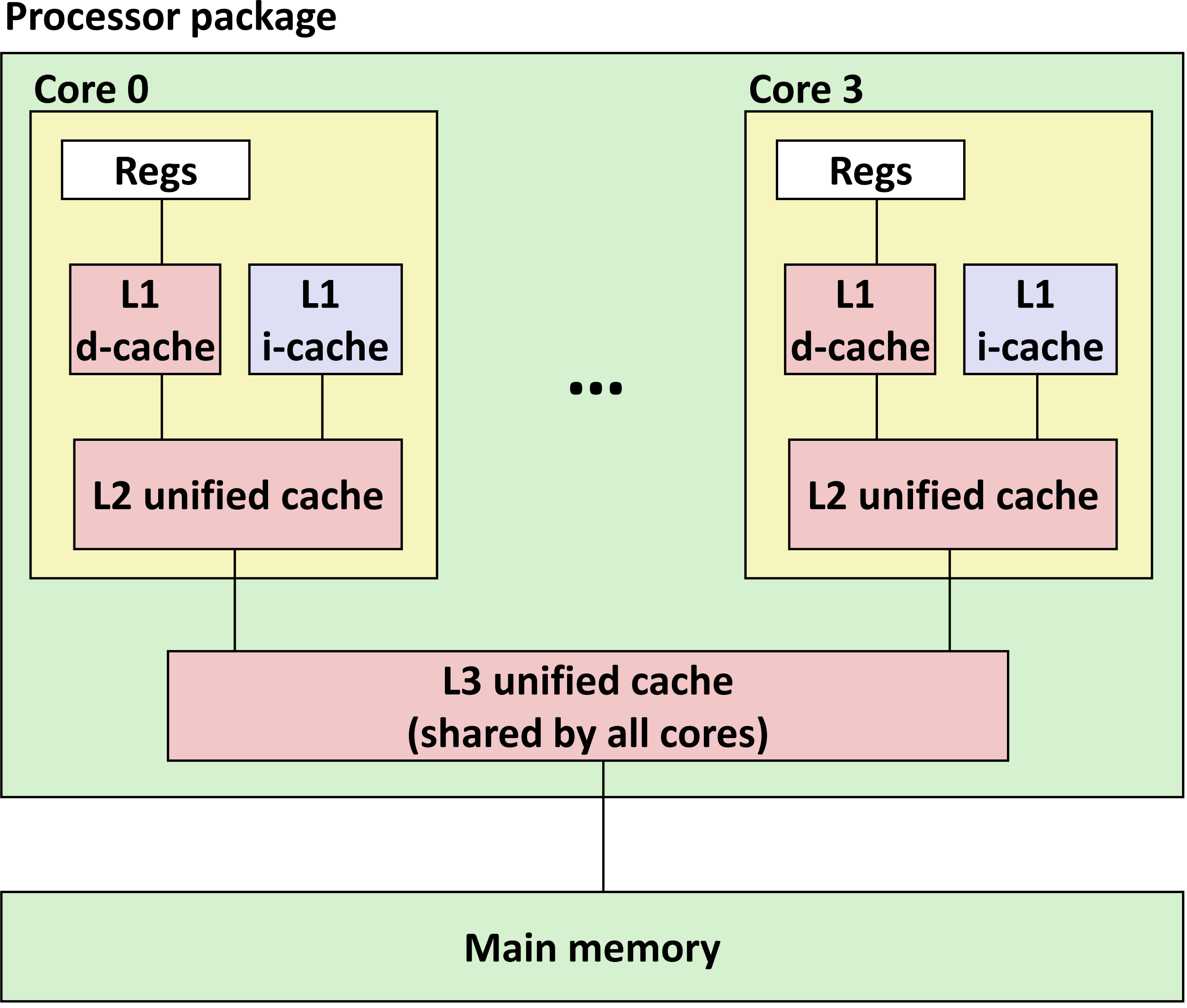


High Bits Indexing

- Addresses of form SSTTBB
  - SS Set index bits
  - TT Tag bits
  - BB Offset bits
- Program with high spatial locality would generate lots of conflicts







Processor package

Core 0

Regs

L1
d-cache

L1
i-cache

L2 unified cache

Core 3

Regs

L1
d-cache

L1
i-cache

L2 unified cache

…

L3 unified cache
(shared by all cores)

Main memory