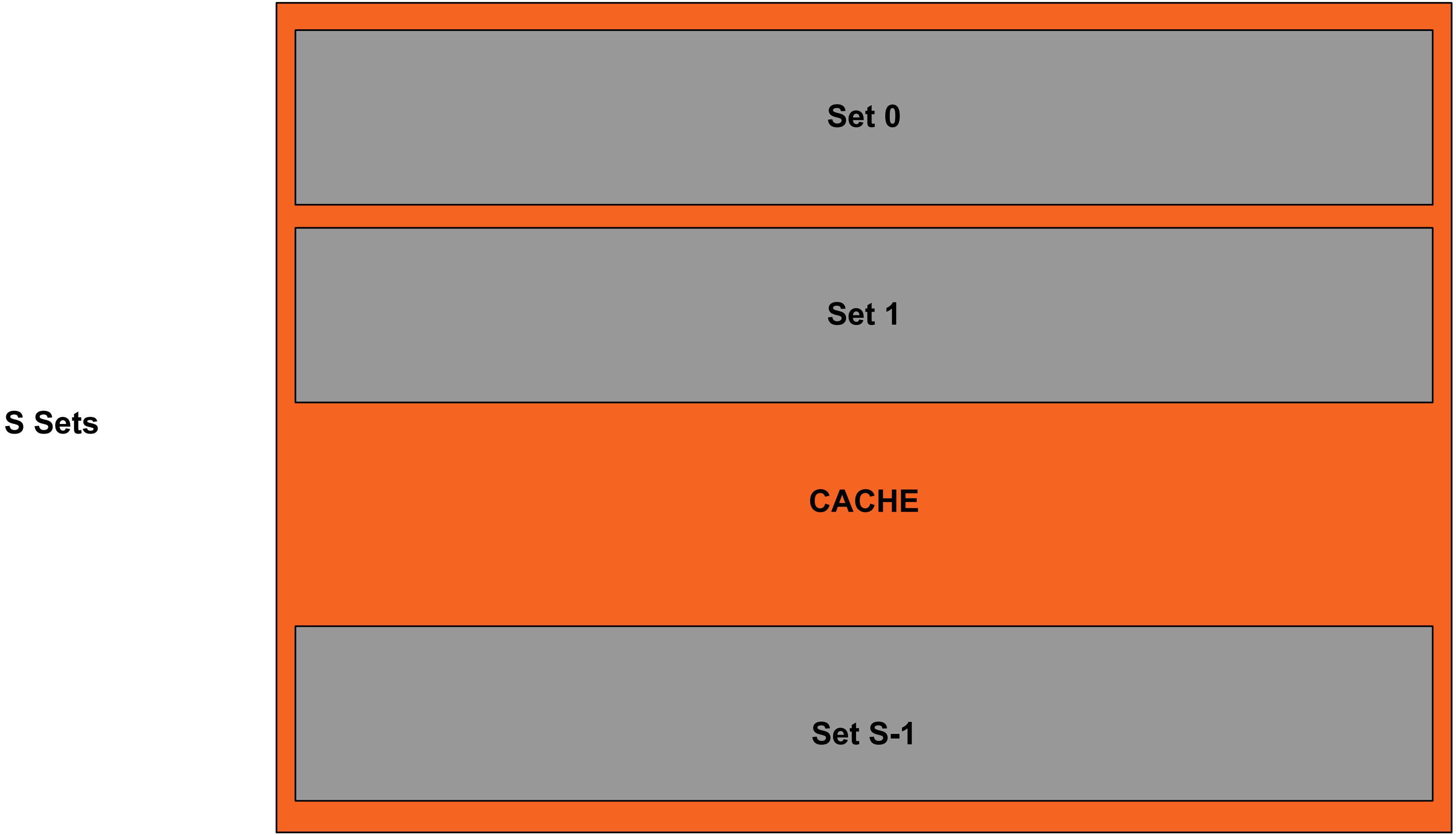
Cache Organization

March 18, 2016

CACHE



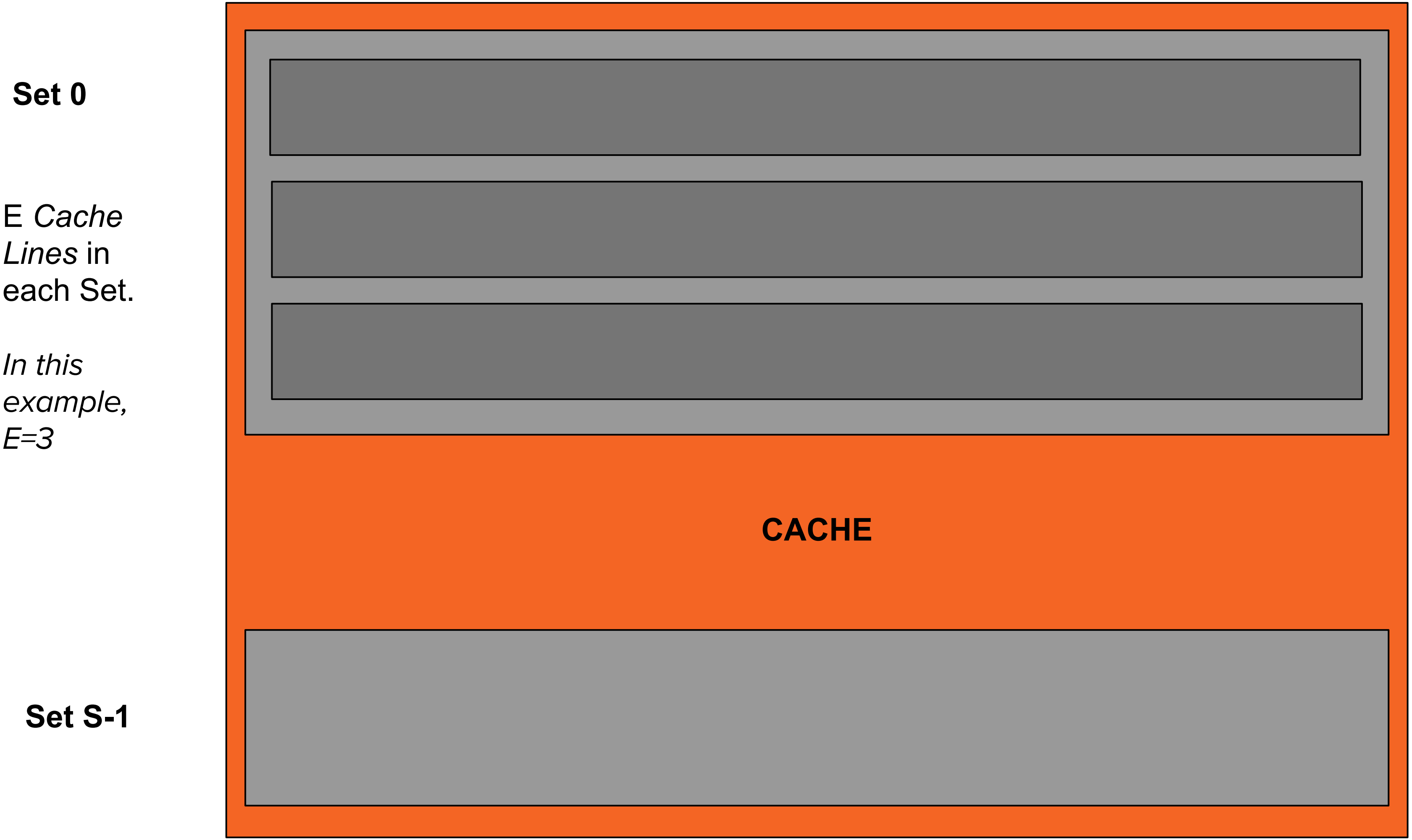
S Sets

Set 0

Set 1

CACHE

Set S-1



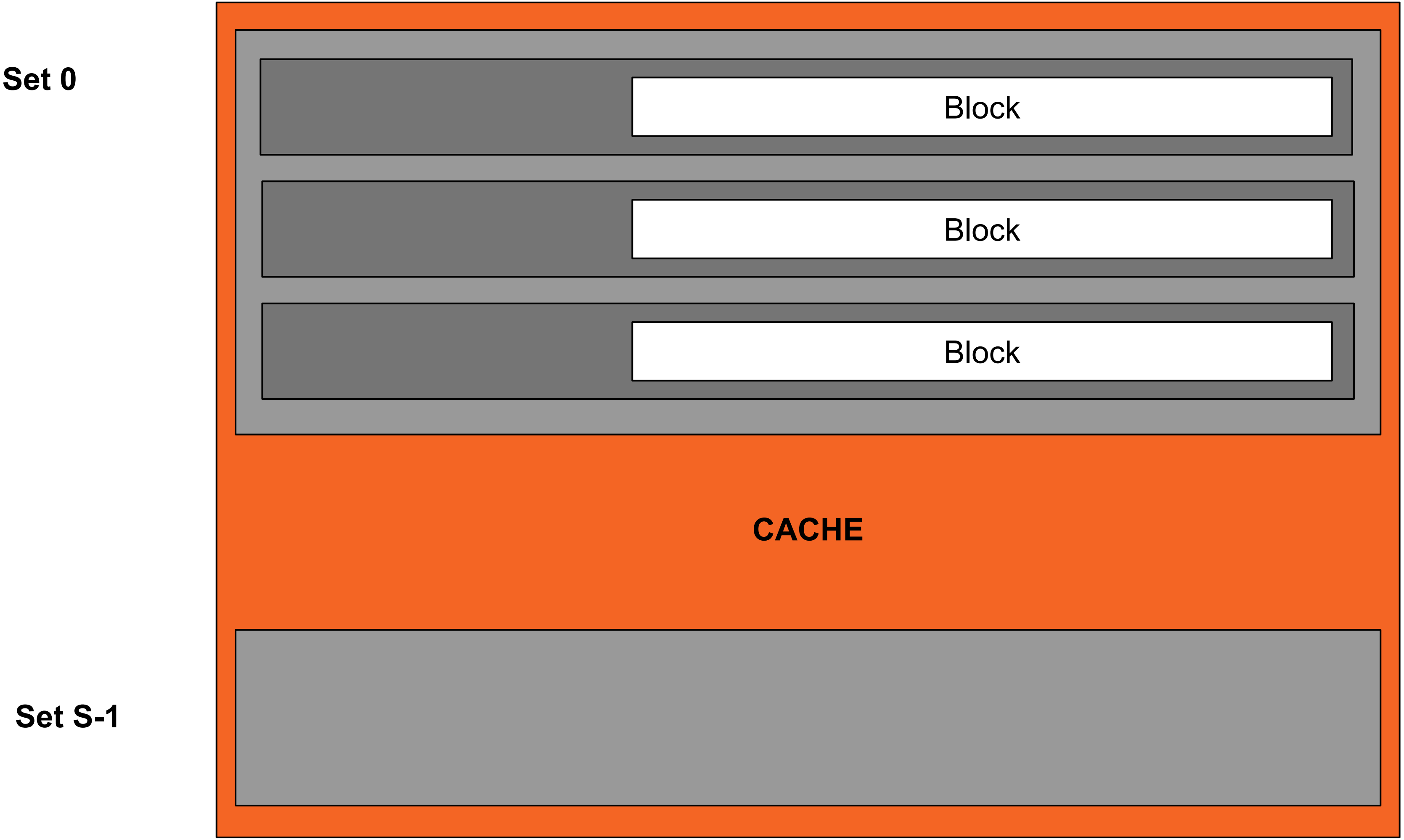
Set 0

E *Cache Lines* in each Set.

In this example, E=3

CACHE

Set S-1



Set 0

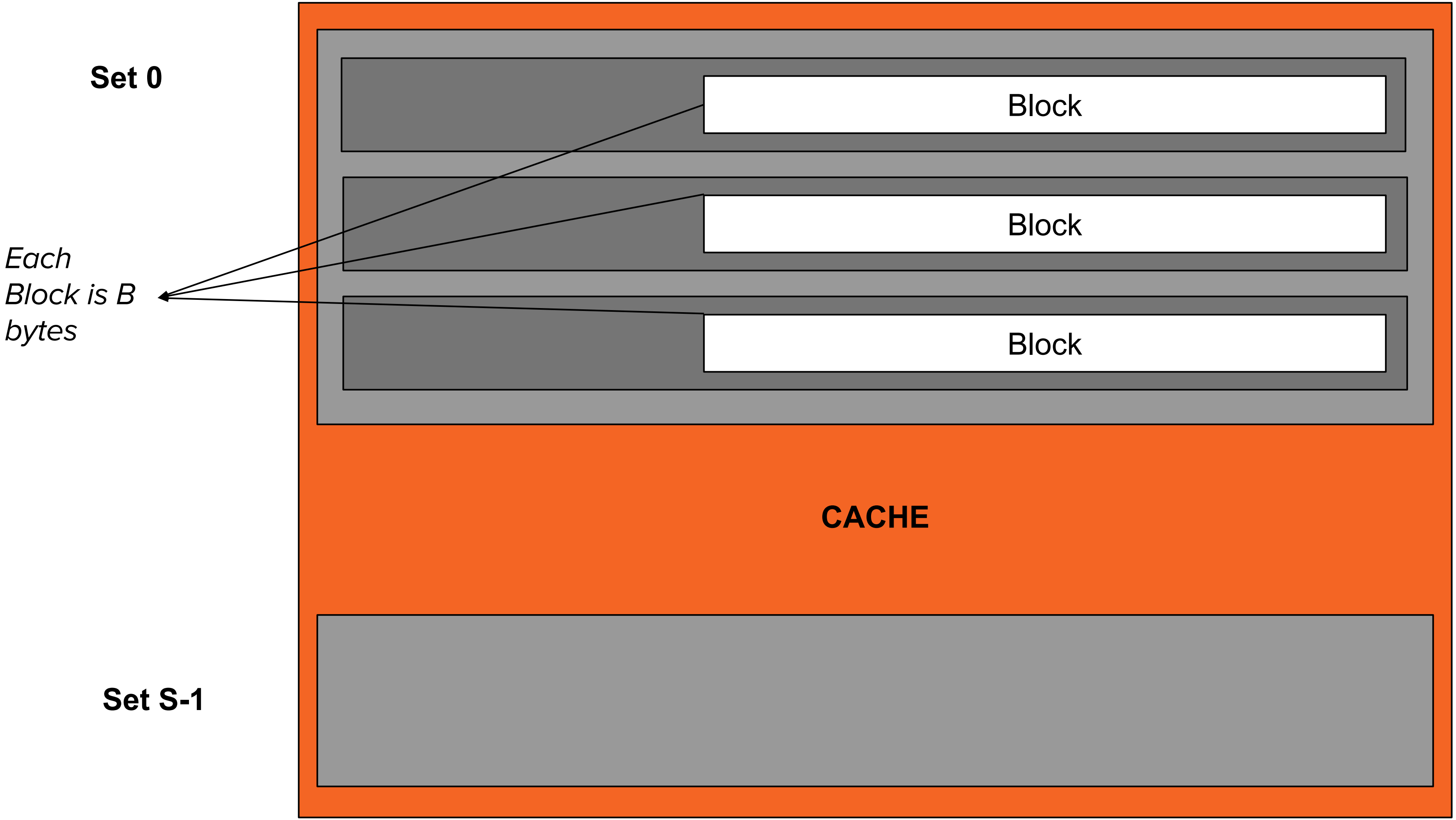
Block

Block

Block

CACHE

Set S-1



Set 0

Block

Block

Block

Each Block is B bytes

CACHE

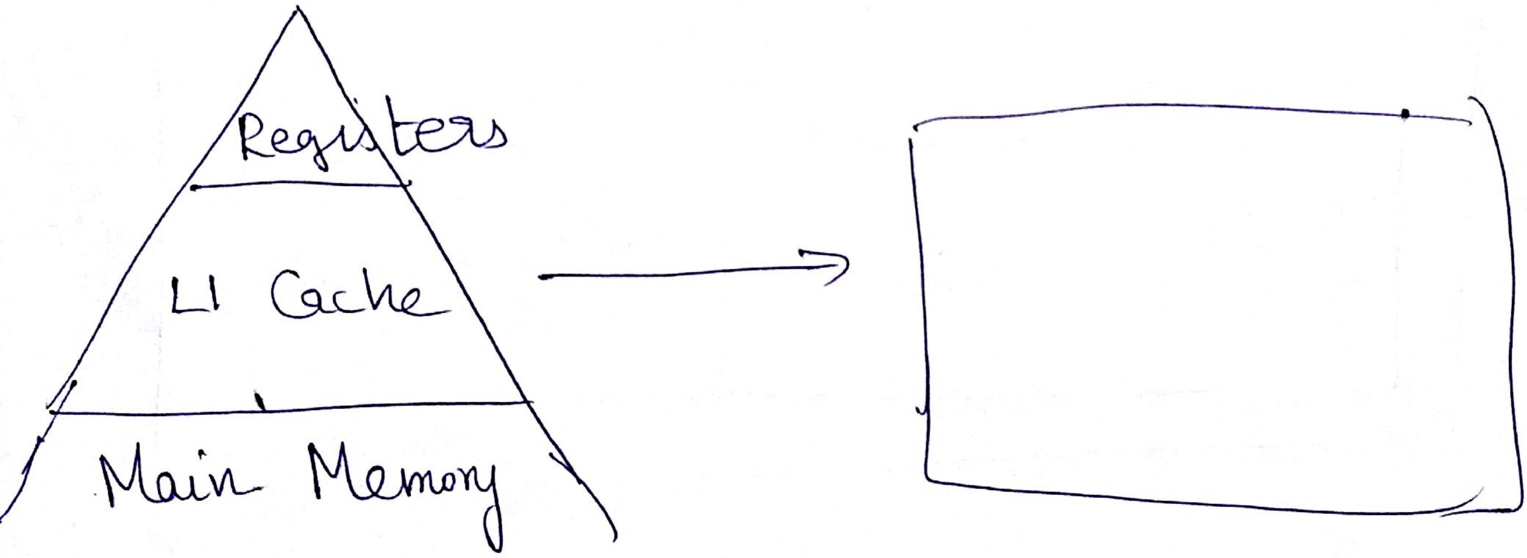
Set S-1

This week (Chapter 6)

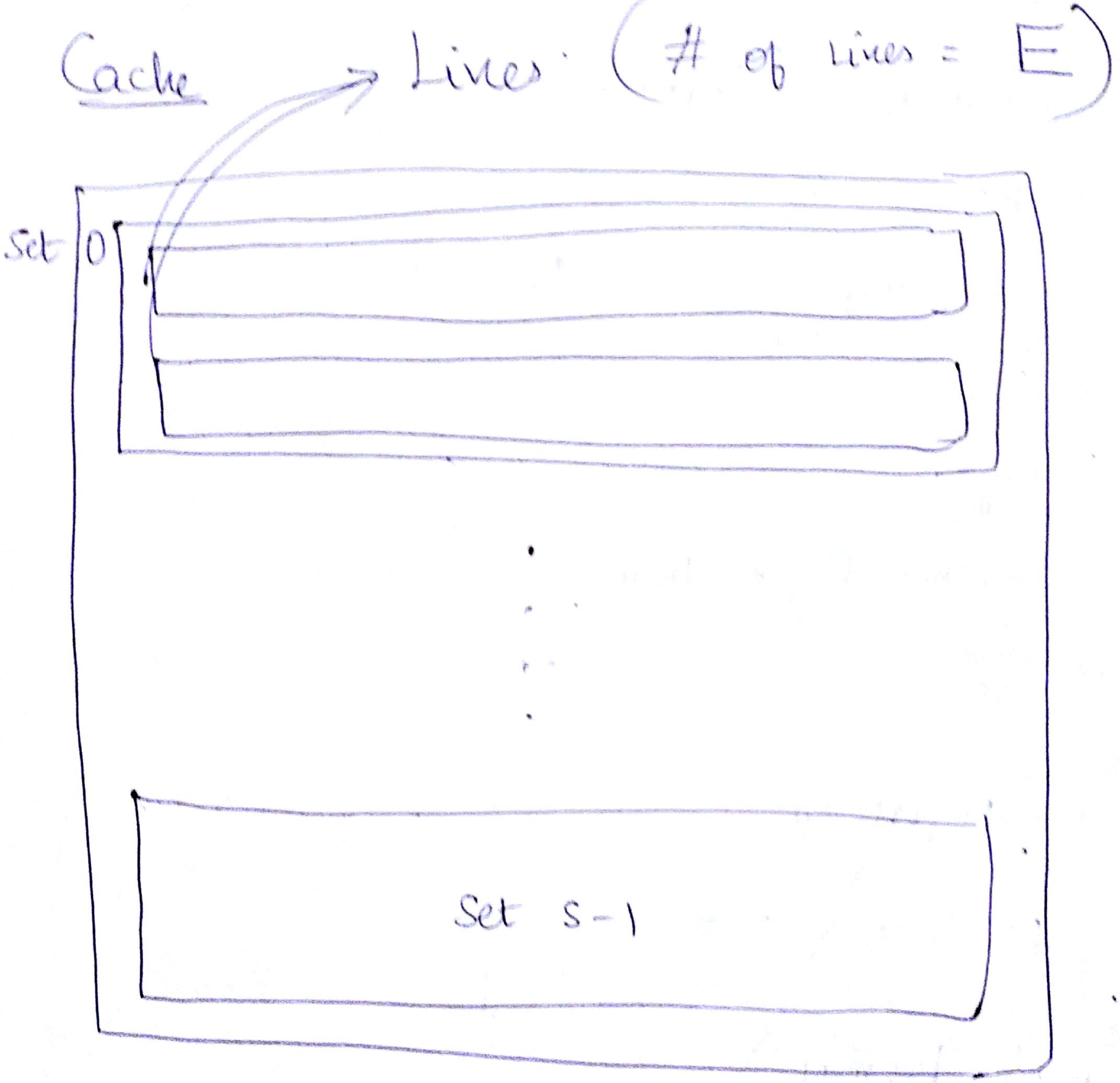
- Memory Hierarchy
- Locality
- Cache (Idea)

Today

- Cache Organization.



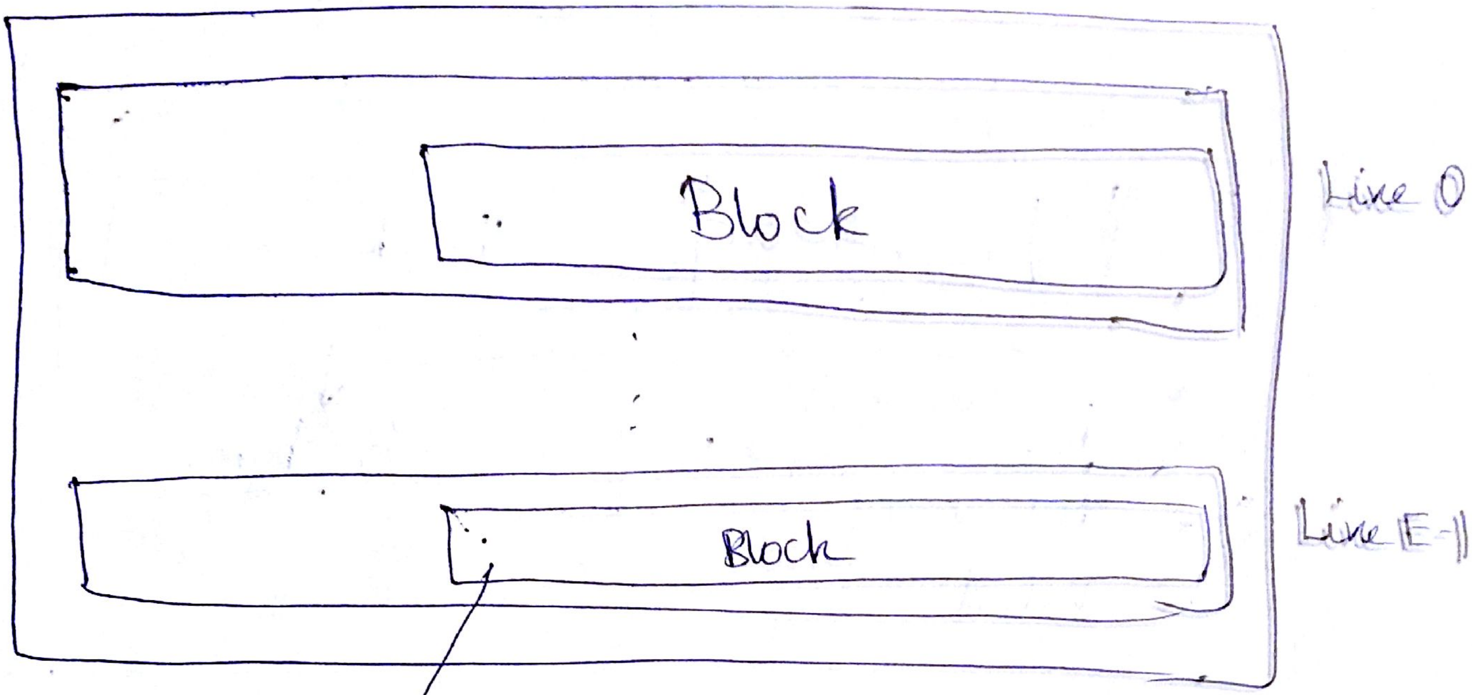
Cache



In total => S sets.

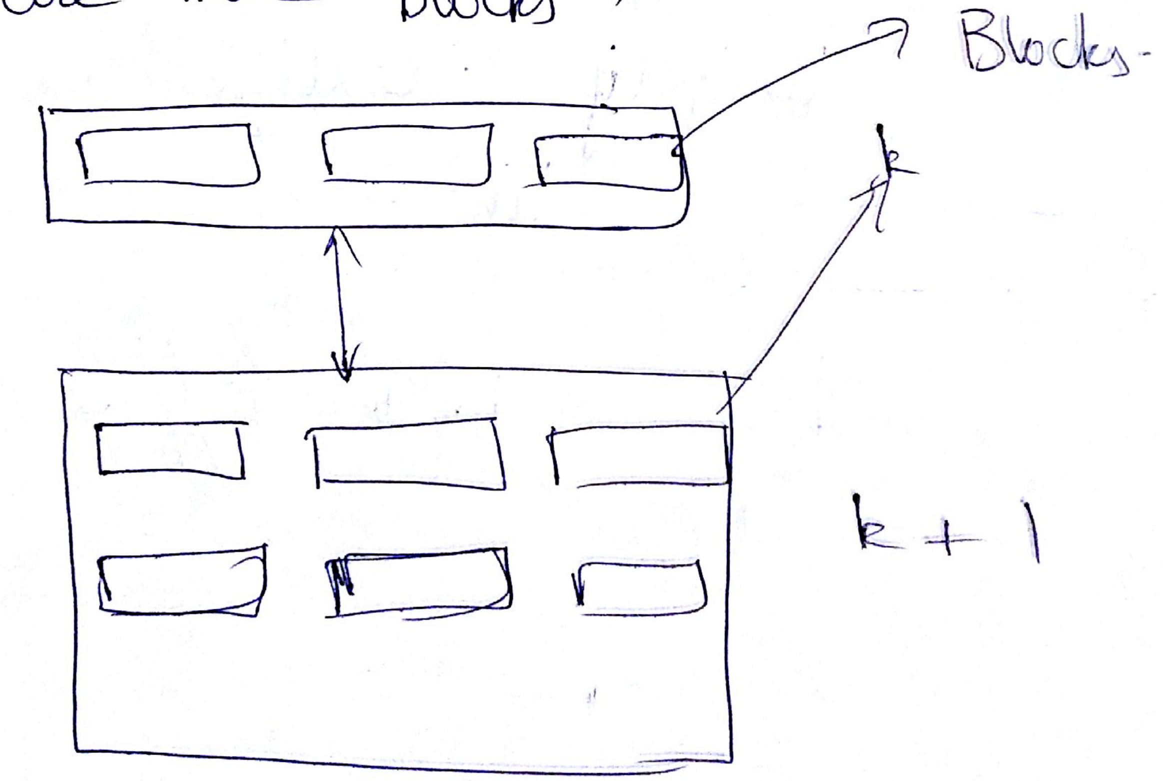
Each set has E lines

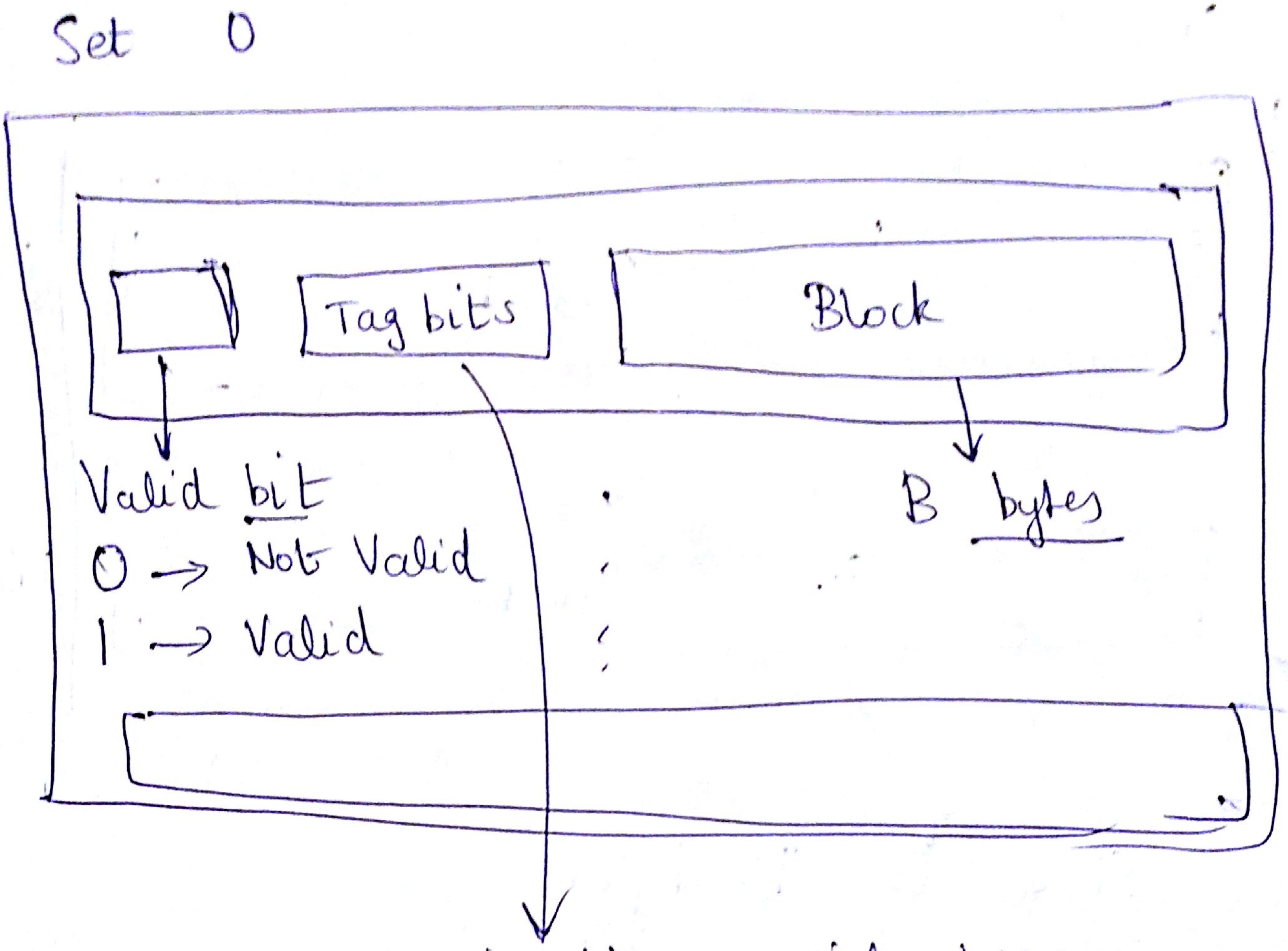
SET 0



Size is B bytes

What are these blocks?

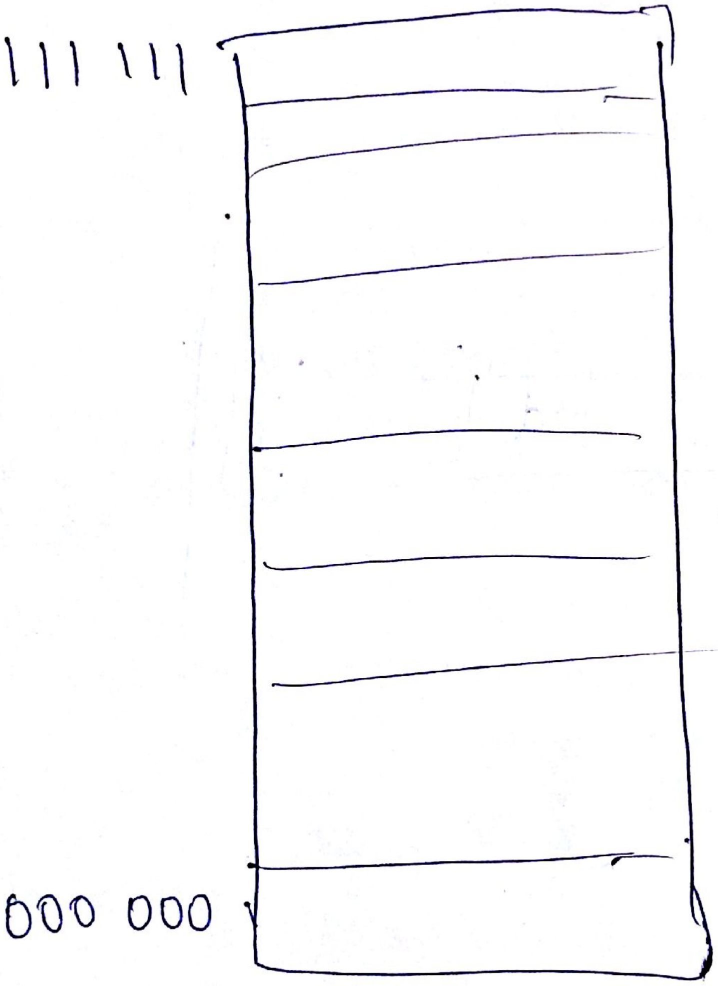




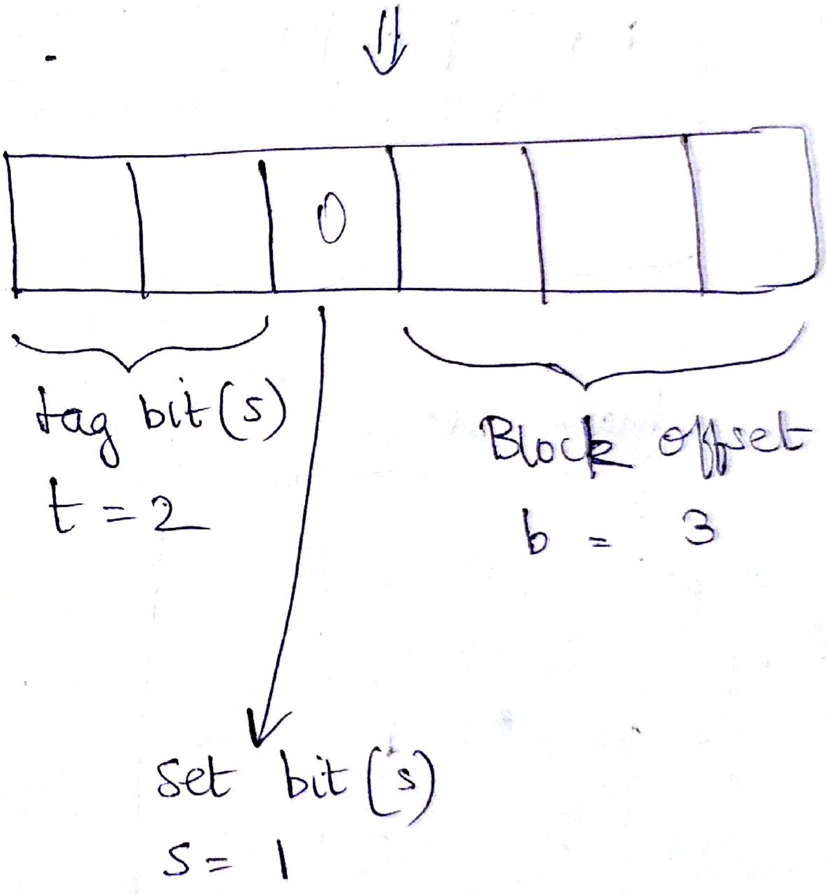
It is like a identifier Uniqely identifies each line.

Basic Structure //.

6 - bit address space. (Example)

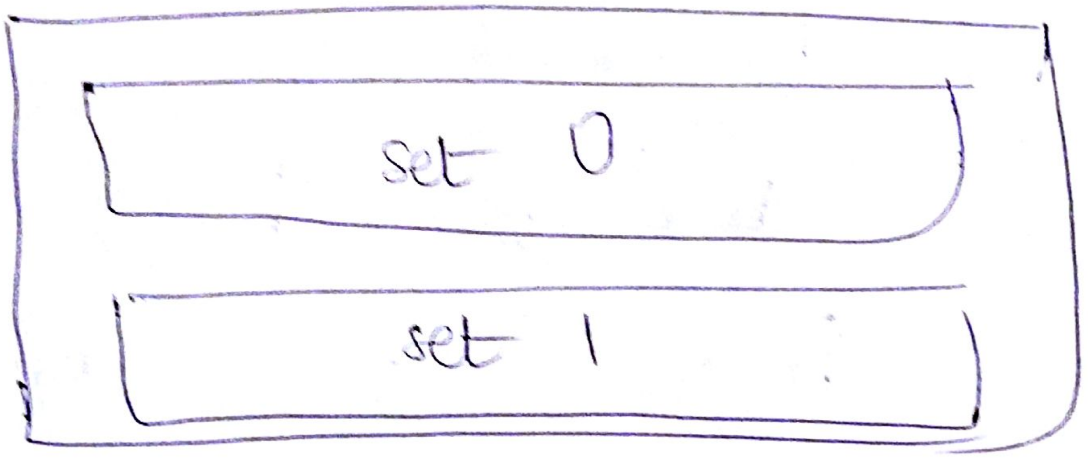


Partition this address



$S = 2^s = 2$ sets.

Cache has 2 sets.



Example.

Value is at address $\underbrace{1\ 0}_{t}\,|\,\underbrace{\underline{0}}_{S}\,|\,\underbrace{1\ 1\ 0}_{b}$

Set 0!

$\underline{0\ 0\ \underline{1}\ 1\ 0\ 0}$ => set 1.

If we did $s = 2$

then $S = 2^s = 4$.

0 0 -> set 0

0 1 -> set 1

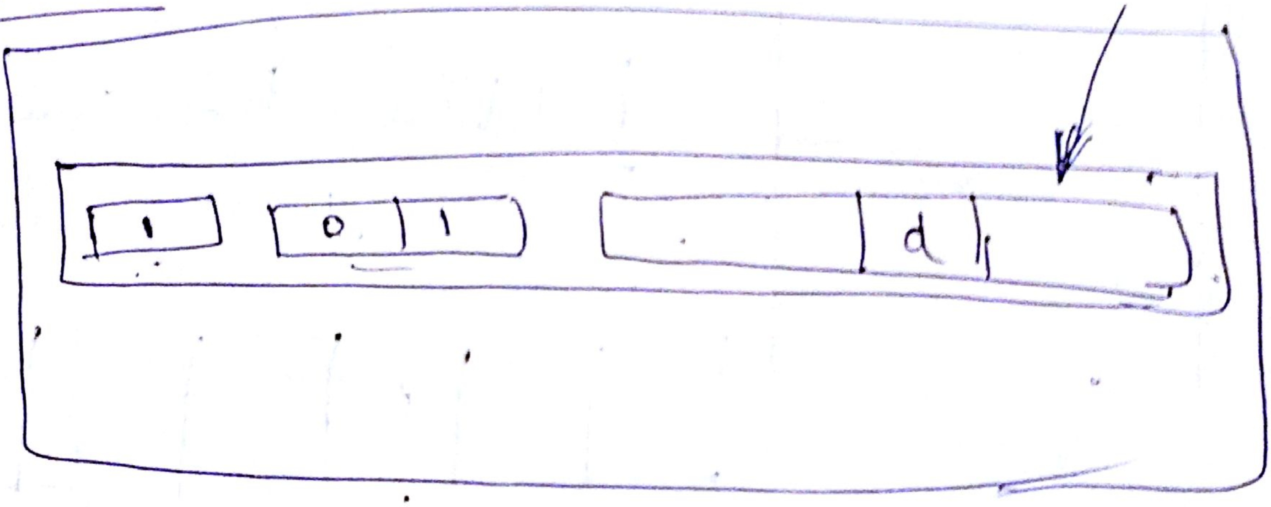
1 0 -> set 2

1 1 -> set 3.

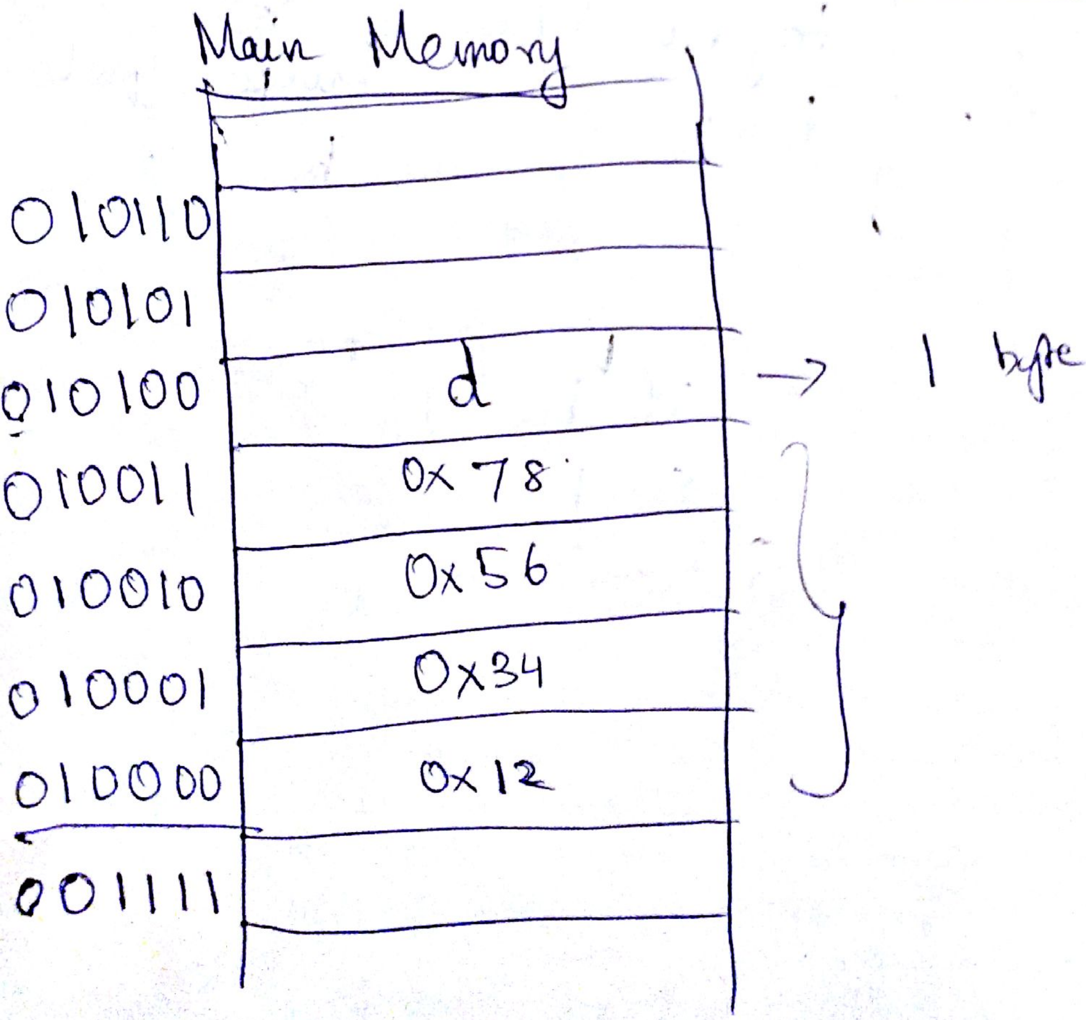
Tag bits:

Eg. 0 1 0 1 0 1 ⇒ The data item d at this address is somewhere in here

Set 0

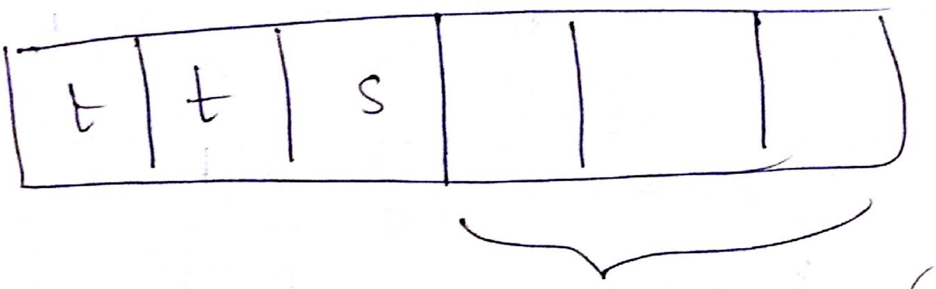


Main Memory



010110	
010101	
010100	d
010011	0x78
010010	0x56
010001	0x34
010000	0x12
001111	

→ 1 byte



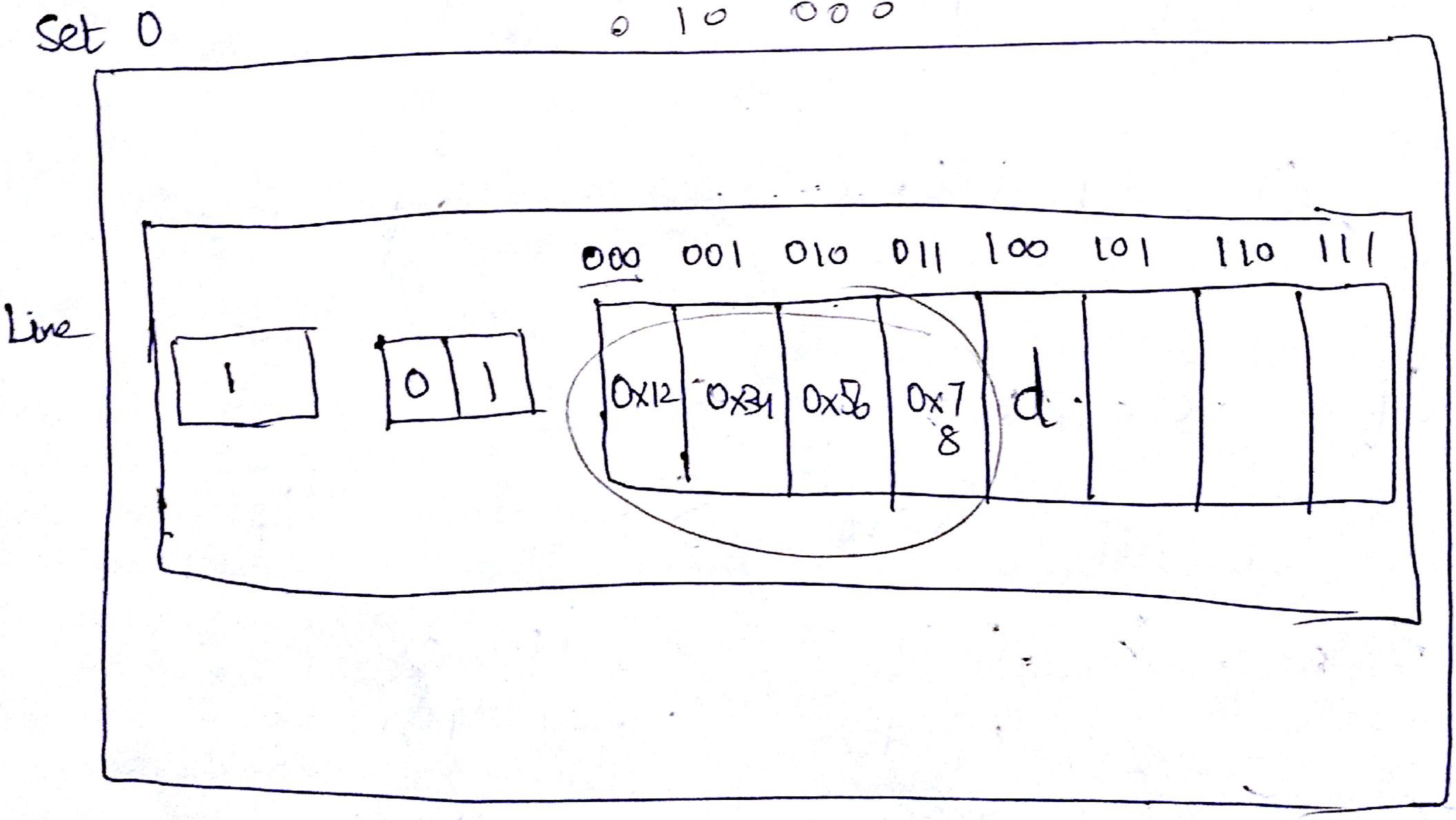
Block offset (b = 3)

$B = 2^b = 8$ bytes.

Each block can store 8 bytes!

d is at 010|100 ⇒ offset

010 000



Suppose we want to access an integer X at $\underline{01}\ \underline{0}\ \underline{000}$

Read 4 bytes starting at offset $\underline{000}$.

Suppose there is no line with the tag bit we were looking for!

⇓

cache miss.

Read the relevant data block from (k+1) and we evict / replace an existing line.

→ If the valid bit is not set (=0) then evict that line. (Simple)

→ If all lines have valid bit set (=1), use a replacement policy.

Typically we do not read in a single byte!

We usually read / access 4 bytes at a time. (In a 32-bit system).

4 bytes => 1 word.

Distinction

IA 32 (Instruction set)	32-bit processor.
1 byte = 1 byte.	1 word = 32 bits (4 bytes).
1 word = 2 bytes (16 bits)	
1 double word = 4 bytes (32 bits)	

Size of the cache

$= S \times E \times B$ bytes.

(excluding the tag bits and the valid bit).

$= 2 \times 3 \times 8 = 48$ bytes //.

How to describe a cache?

$$(S, E, B, m)$$

$$(2, \textcircled{3}, 8, 6)$$

How many bits for the set?

$$s = \log_2(S)$$

$$b = \log_2(B)$$

$$t = m - (s + b)$$

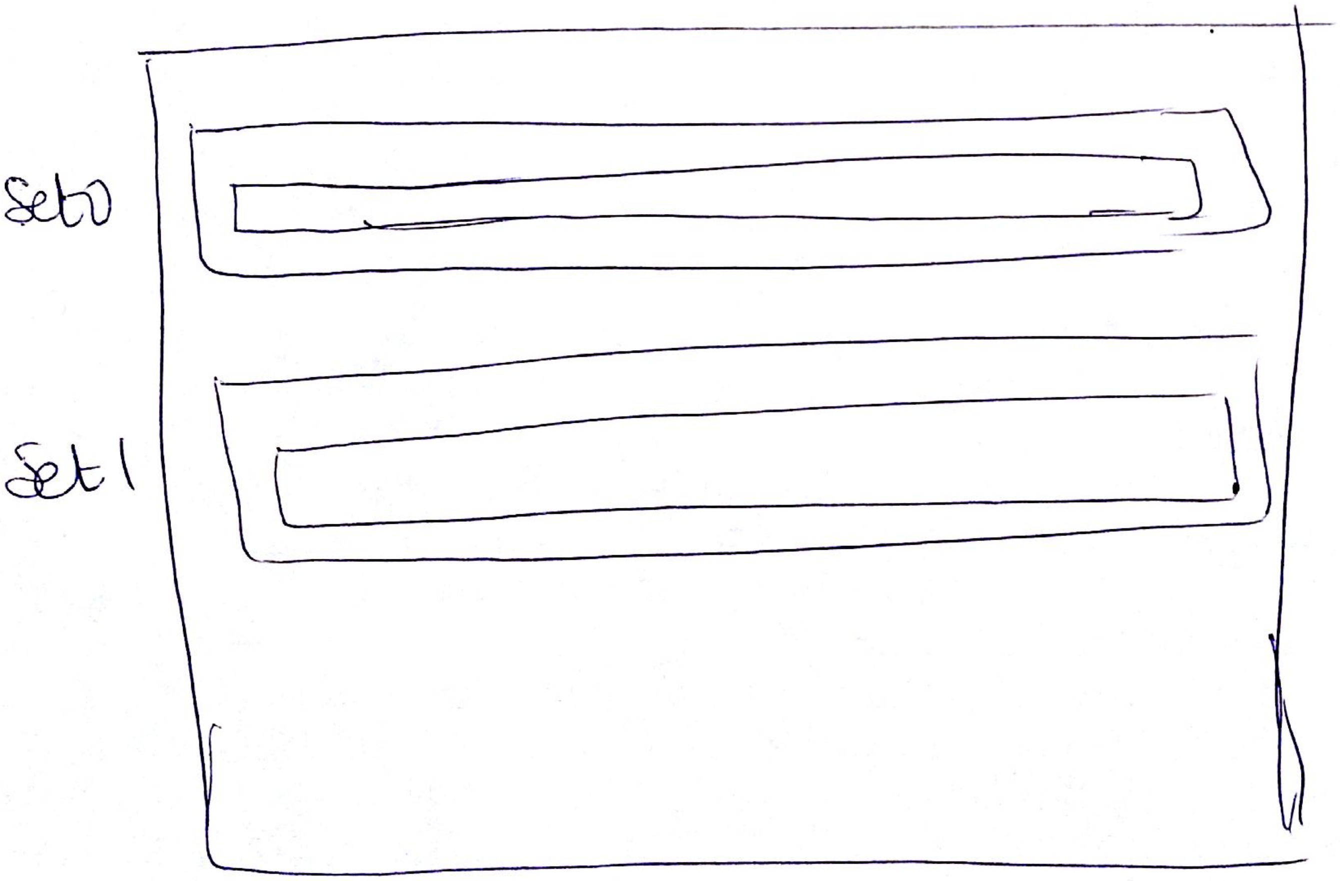
$$= 6 - (1 + 3) = 2 \,//$$

Different Classes of cache

↓

based on the number of cache lines. // (E)

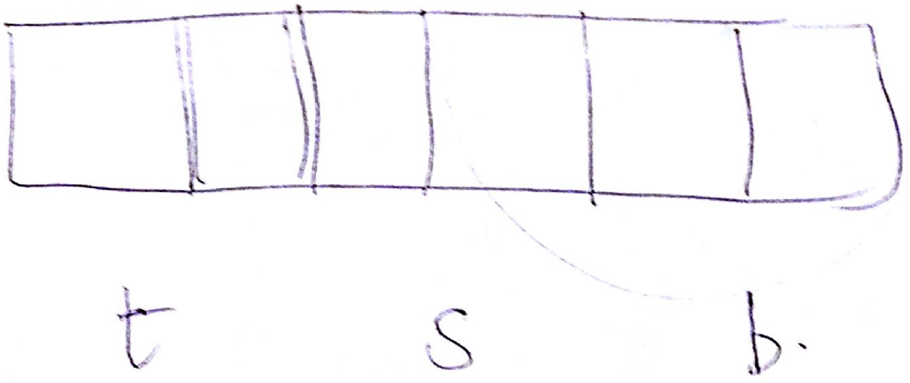
If E = 1 (Direct Mapped Cache)



$(\boxed{S}, \underline{E}, \boxed{B}, \boxed{m})$

Extra

$m = 6$



$E = 1$

$\underline{\underline{E}} = 100$ => Bigger the E value, the fewer the misses you'll get.

But it makes no sense to have $E > 2^t$. Because you'll only be able to reference 2^t lines at max

$C = S \times \boxed{E} \times B$