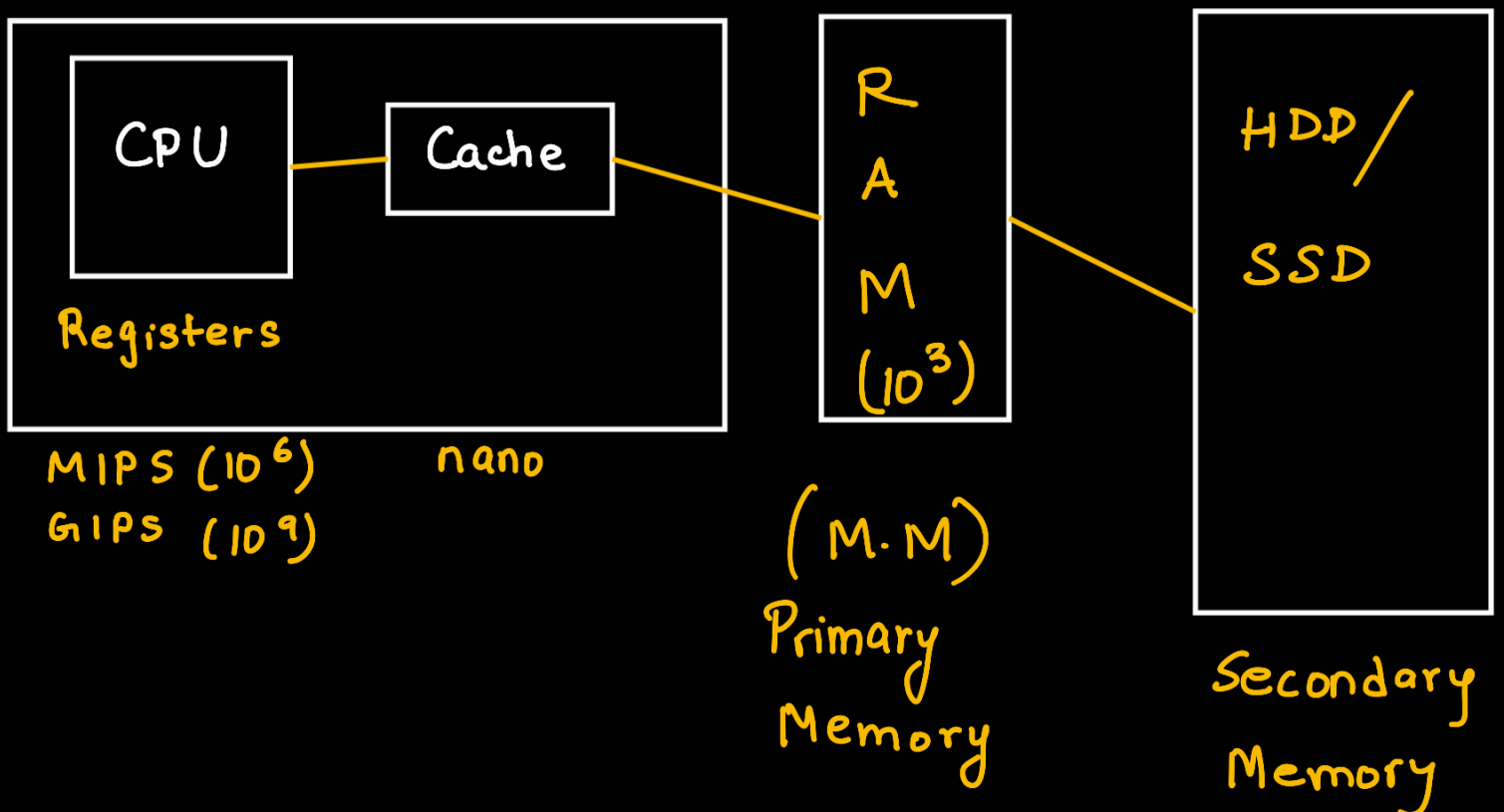
Cache Mapping

- Direct
- Fully Associative
- k-way set Associative



Flow of Data

Fetch from secondary memory to MM.

Fetch from MM to cache

Fetch from cache to CPU

1 Hr

10 Lac Hr

Capacity of

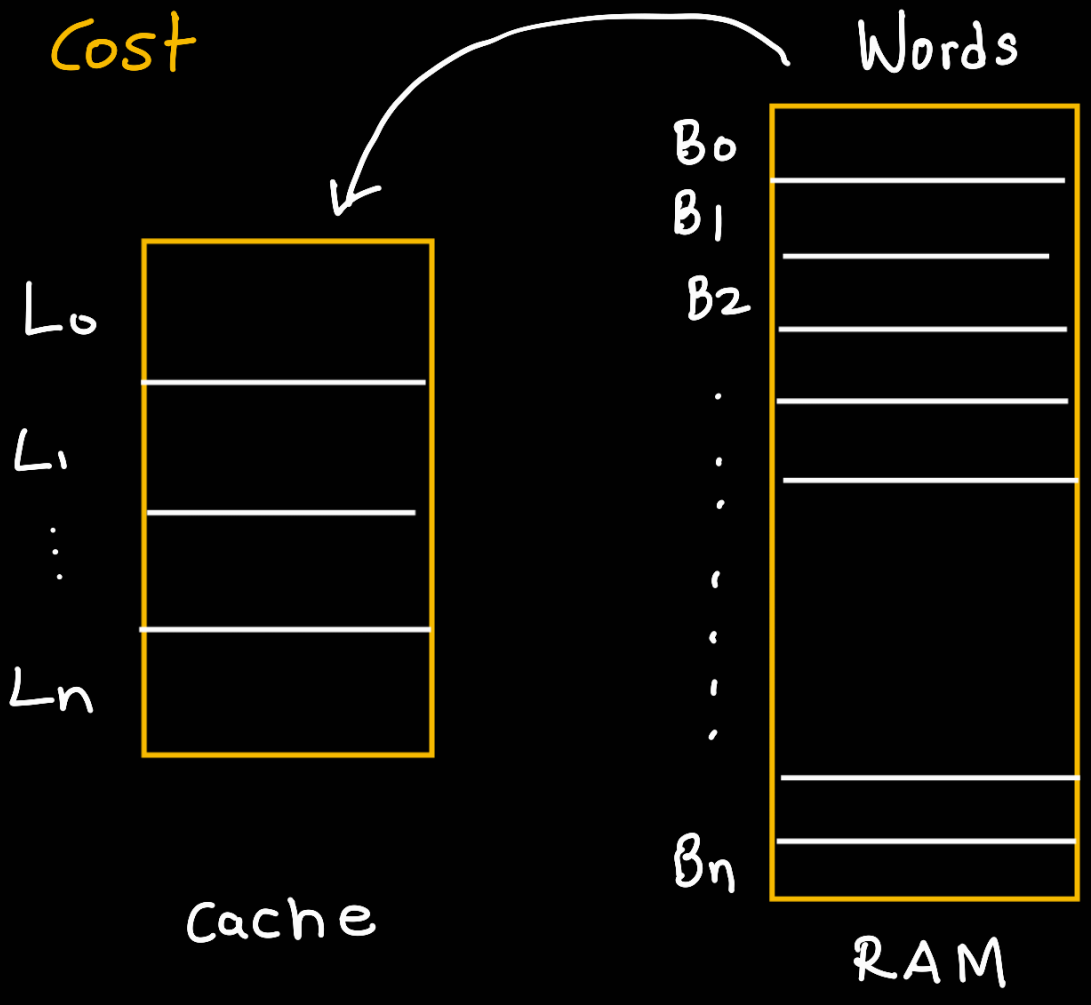
HDD → TB

RAM → GB

Cache → MB

So there has to be a trade off between speed & cost.

Cost



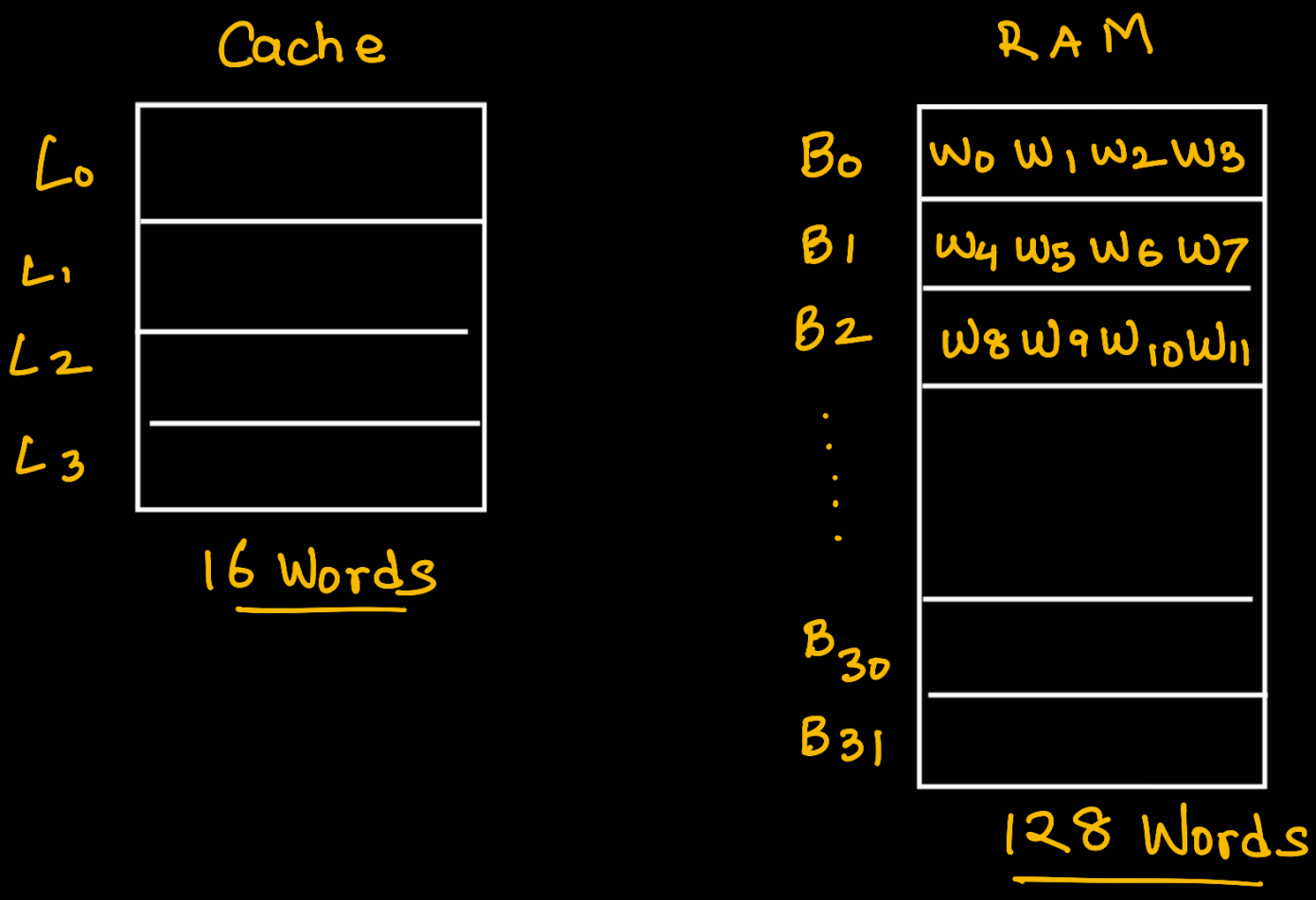
Whenever the CPU encounters a Cache Miss, it fetches the blocks of words from the main memory.

From Blocks to Lines

Word / Byte : Minimum addressable unit of memory.

Cache Mapping : It is the process which instructs the CPU, exactly where inside cache, the words fetched from the RAM are stored into the cache lines.

Example



1 Word → 2/4/8 Bytes

1 Block can store 4 Words

∴ 32 Blocks are there in the main memory.

Similarly 1 Line ⟶ 4 Words

∴ Total 4 Lines are required.

Direct Mapping

Follow the formula $K \bmod n$

- K → Block No.
- n → No. of lines

For ex. Block 0

$\therefore$ $K \bmod n = 0 \bmod 4 = 0$

$\therefore$ Block 0 goes into Line 0

$\therefore$ Block 1 goes into L_1 $(1 \bmod 4) = 1$

Block 2 goes into L_2

Block 3 goes into L_3

Block 4 goes into L_0 $(4 \bmod 4) = 0$

∴ L_0 consists of B_0 ∴ $w_0 w_1 w_2 w_3$

∴ L_1 " " B_1 ∴ $w_4 w_5 w_6 w_7$

Also, L_0 may contain $B_4, B_8, B_{12} \cdots B_{28}$

L_1	B_0 B_4 .... B_{28}
L_2	B_1 B_5 .... B_{29}
L_3	B_2 B_6 .... B_{30}
L_4	B_3 B_7 .... B_{31}

It is to be noted that at a time, only 1 block may occupy the cache line.

No 2 blocks can co-exist inside the same cache line at a time.

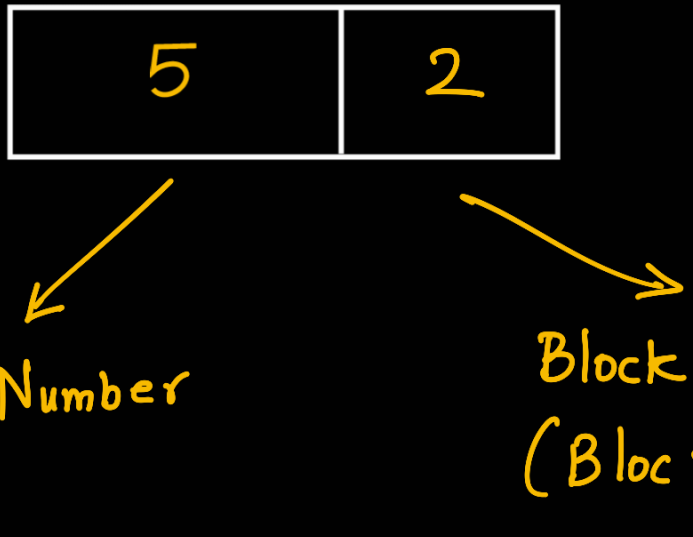
Also, these blocks will always have a fixed location

Even if a line is free, it cannot be allocated to another block. (Conflict Miss).

How the CPU access the WORD ??

128 words ($W_0 - W_{127}$) can be accessed using 7 bits, i.e 2^7.

These 7 bits are further broken down as:-



To represent the block offset i.e No. of words in each block, we need 2 bits, i.e $2^2 = 4$

∴ Each Block can store 4 words. These words can be accessed using the Block Offset.

The remaining 5 bits are used to represent the block number.

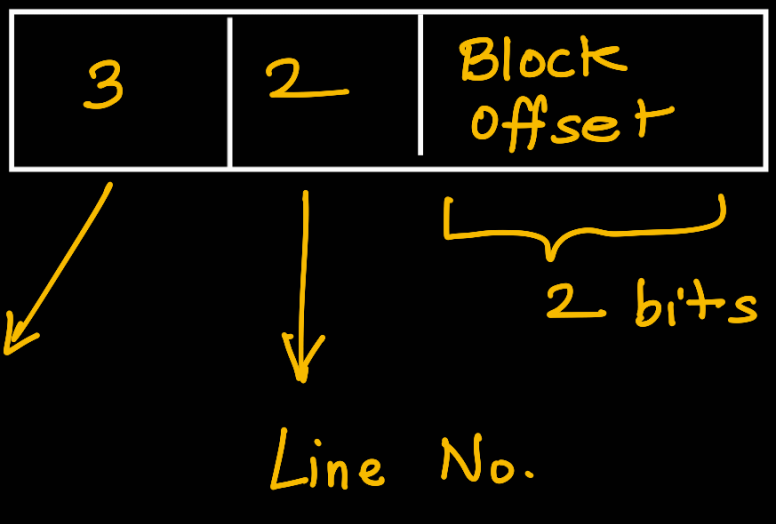
Ex: The CPU generates an address

00010 10

Block 2 | Word 2

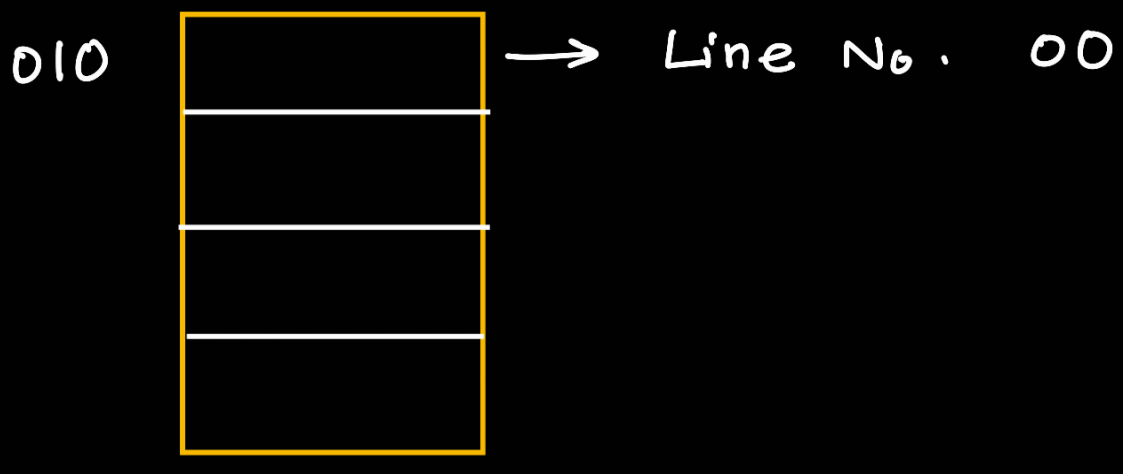
This is how the CPU generates the physical address for a Word.

How to resolve the data present in the cache?



The Tag represents which blocks can be accomodated in a cache line.

For ex. Tag = 010



∴ The locations are as follows:

010 00 00 → 32
010 00 01 → 33
010 00 10 → 34
010 00 11 → 35

} Block 8 will be selected

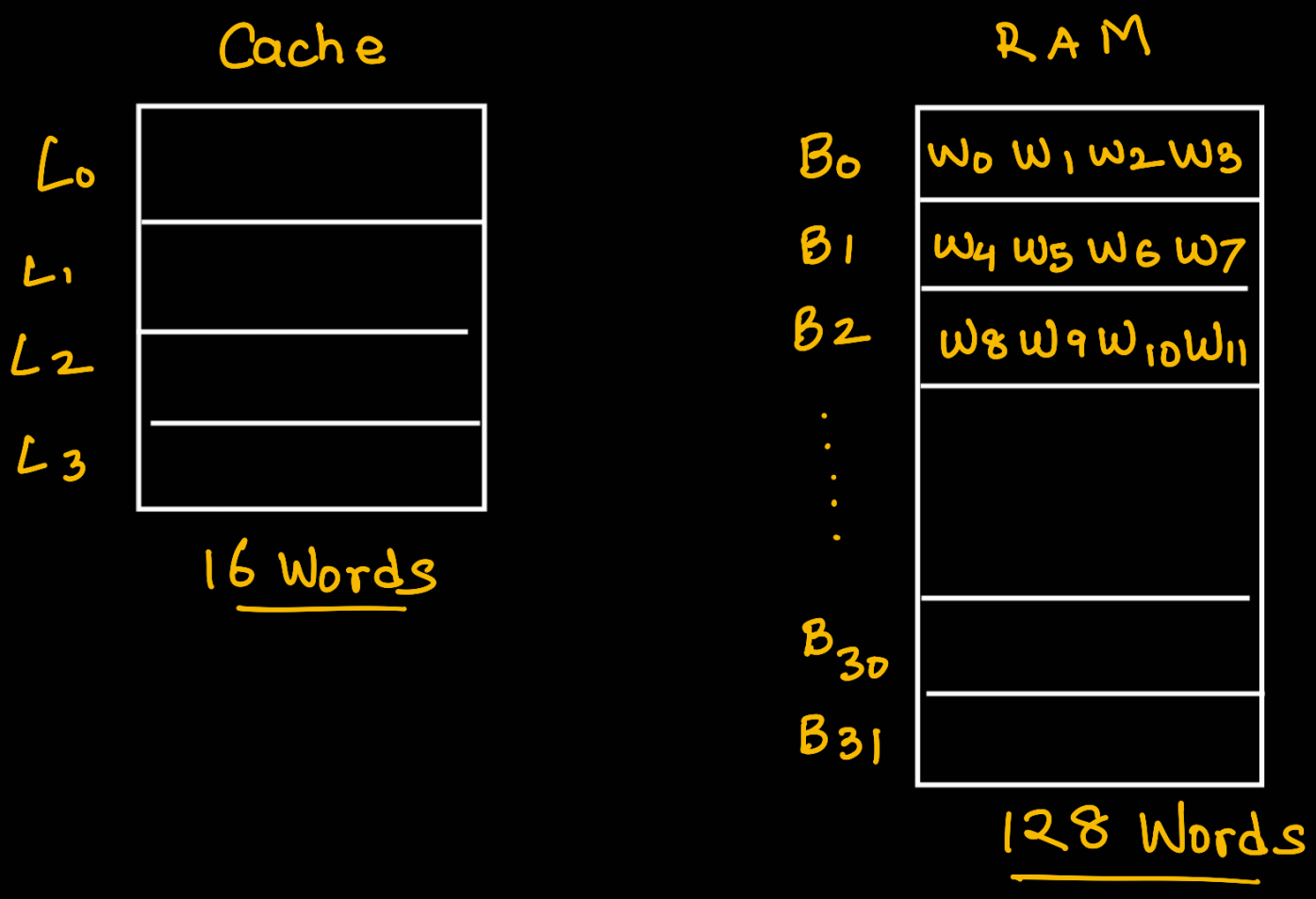
010 ⇒ 2nd Block i.e

$B_0, B_4, (B_8) \ldots$

Example. Tag → 001
line → 01

} Which block will be fetched.

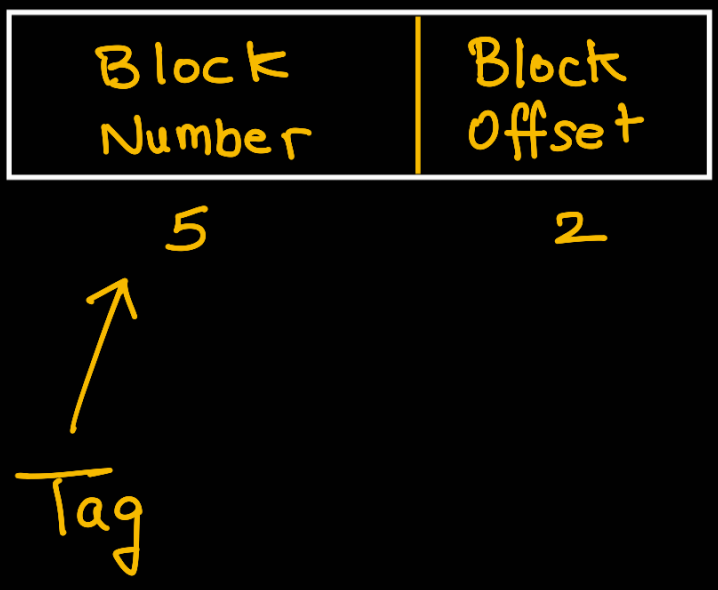
Fully Associative



In fully associative mapping, any block can be accomodated in any free line.

This resolves the issue of conflict miss.

Physical Address:



There is no need to fix a line.

Any of the 2^5 blocks can be present in the cache.

Example:

Tag → 00100

Then Block to be selected is 4.

∴ 00100 00
00 100 01
00 100 10
00100 11

} Words $W_{16} - W_{19}$ can be fetched from the M.M and brought into the cache.

It can be put into any free cache line.

No. of hits increases in this case

Tag Size Increases

K-Way Set Associative Mapping

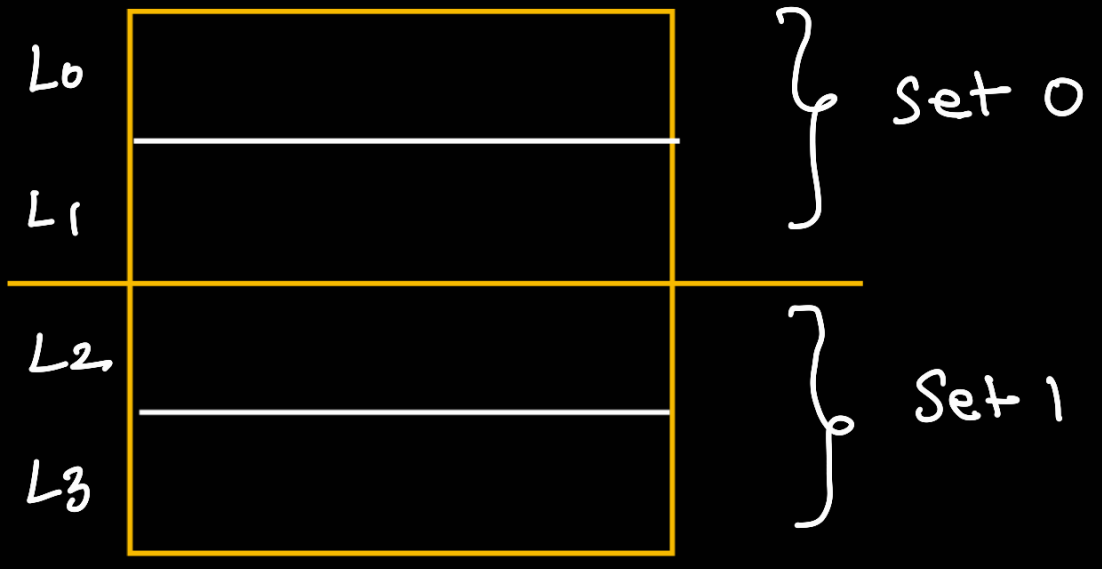
It is a combination of both Direct & fully associative cache mapping.

K → No. of lines in each set

∵ No. of sets = $\frac{\text{No. of lines}}{\text{Value of K}}$

∴ No. of set in our example

⇒ $\frac{4}{2}$ = 2 sets



First we fetch the block from the Main Memory using Direct Mapping

But where should we put the block of Words

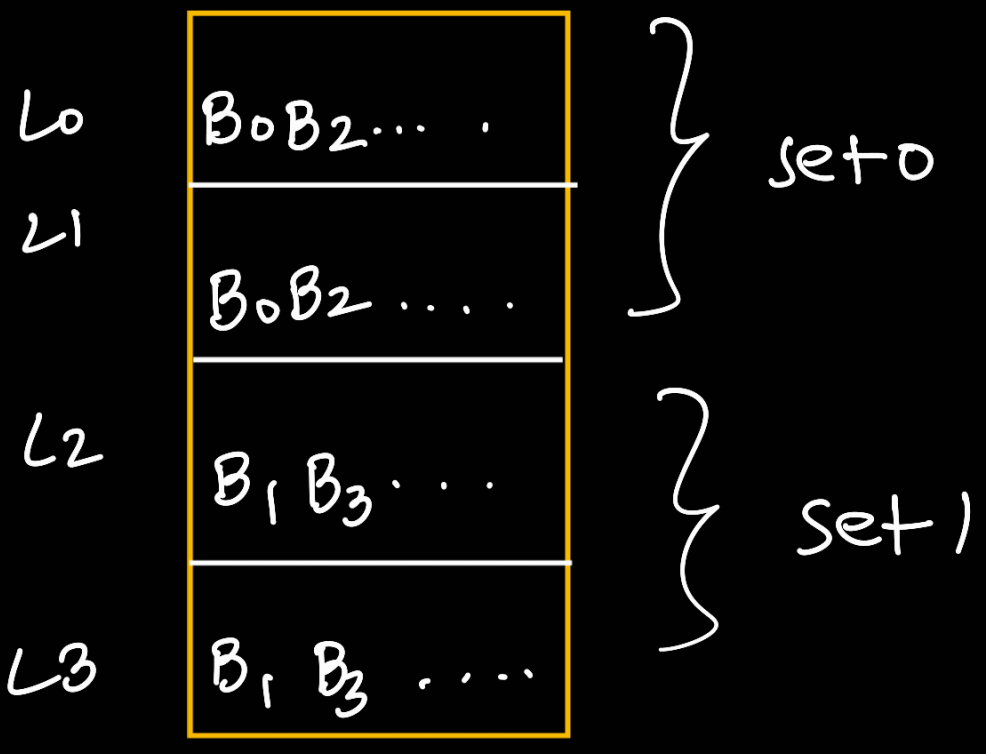
For. ex. Suppose Block 1.

k mod n

$\Rightarrow$ 1 mod 2 $\Rightarrow$ 1

$\therefore$ Hence Set 1 is selected

We can then accomodate the block of words in any of the lines of set 1



Physical Address

Block Number	Block offset
5	2

Cache memory Mapping

Tag	Set No.	Block offset
4	1	2

$\therefore$ Tag $\rightarrow 2^4 \rightarrow 16$

So any possible 16 blocks can occupy the lines.

$\therefore$ Even & Odd sets

Ex