COMPUTER SCIENCE

Computer Organization and Architecture

Cache Memory

Lecture_05

Vijay Agarwal sir

01 Mapping Techniques

Cache Memory.

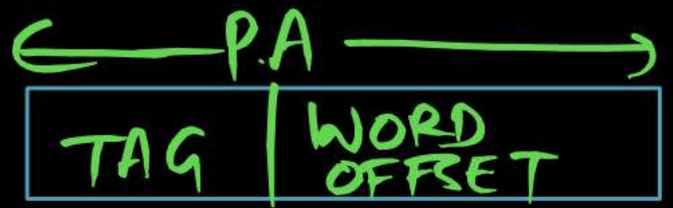
Cache Design

① Memory Org.

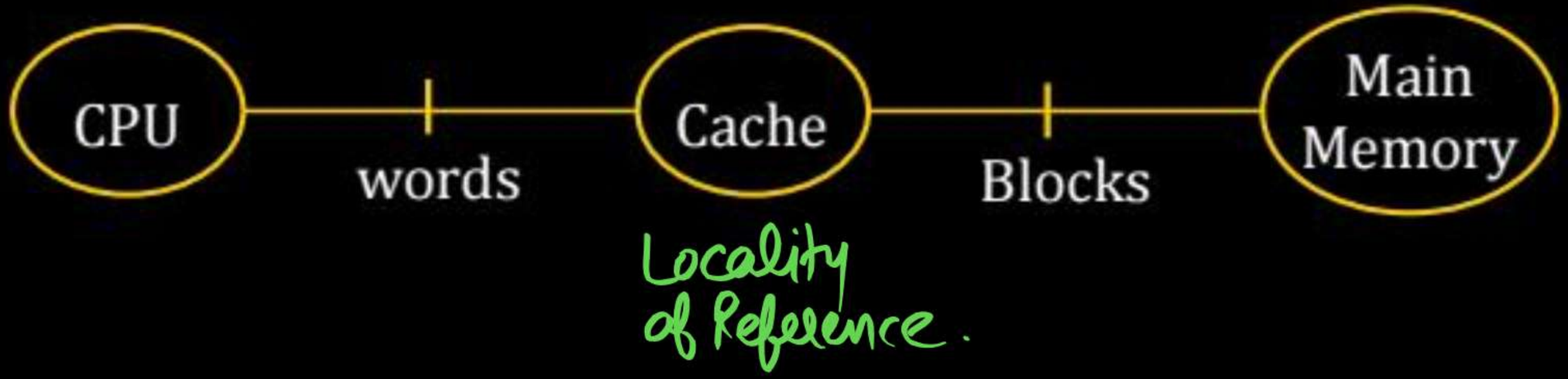
Cache memory

Main Memory.


Cache Index
LINE OFFSET
Block (WORD) OFFSET

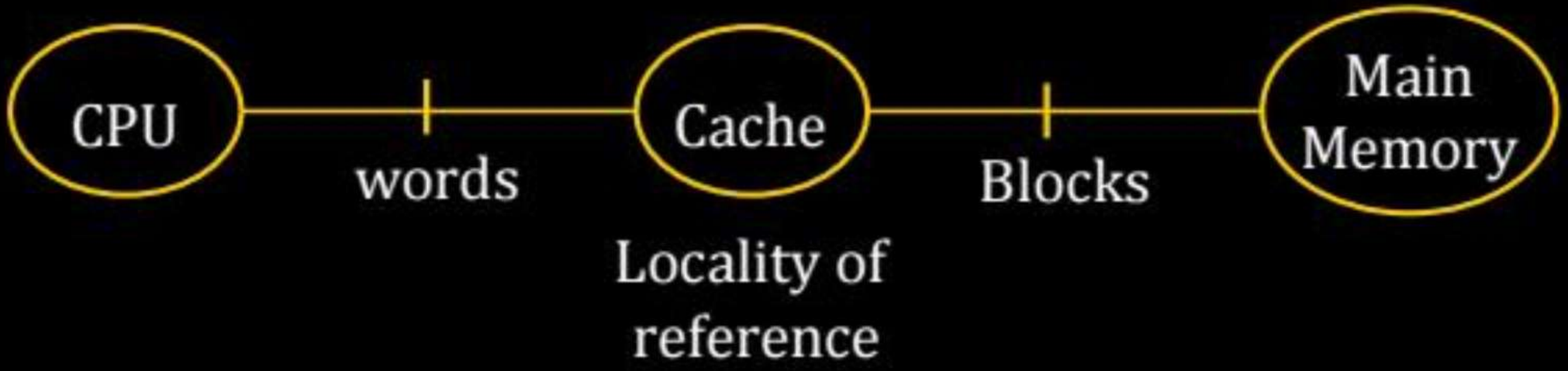

P.A
TAG
WORD OFFSET

Memory Organization



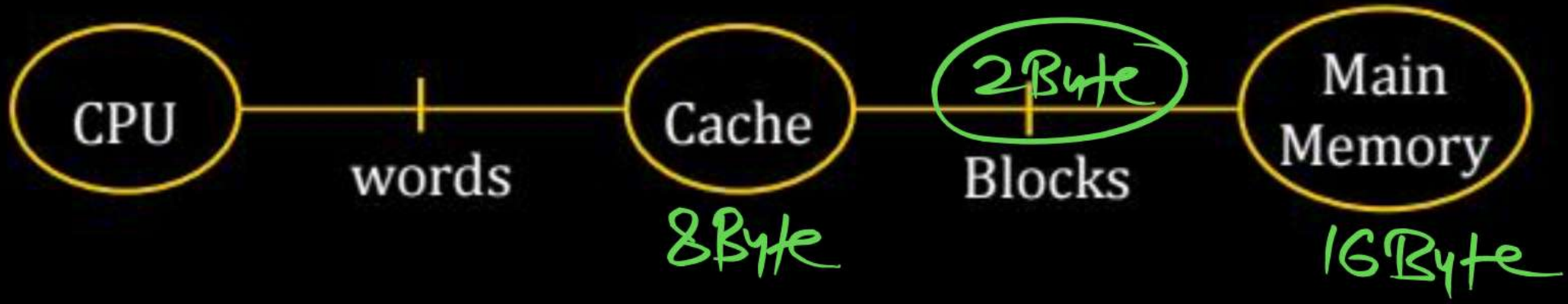
Locality of Reference.

Cache Memory

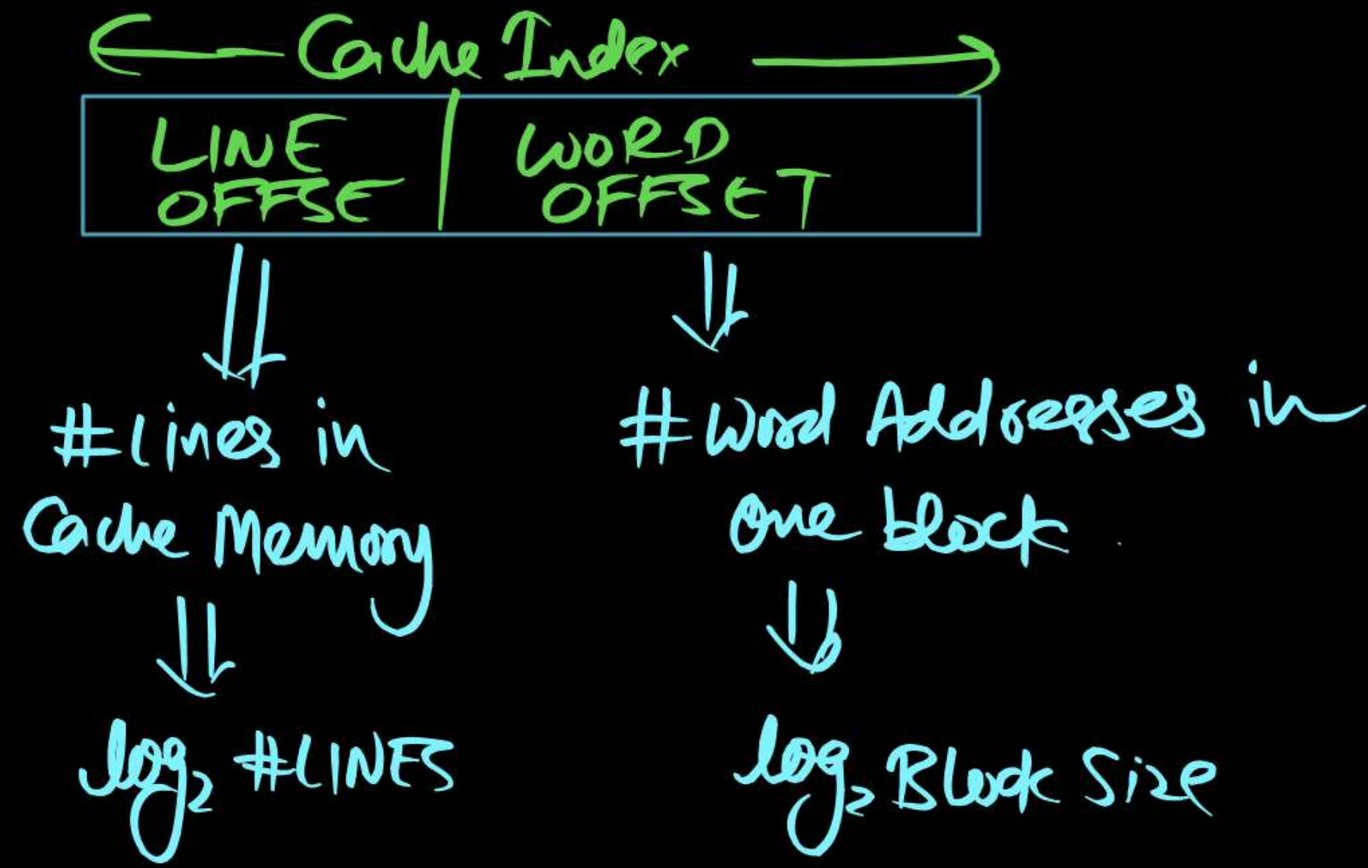


1)Memory Organization.
2) Mapping Technique.
3) Replacement Algorithm.
4) Updating Technique.
5) Multi level cache.

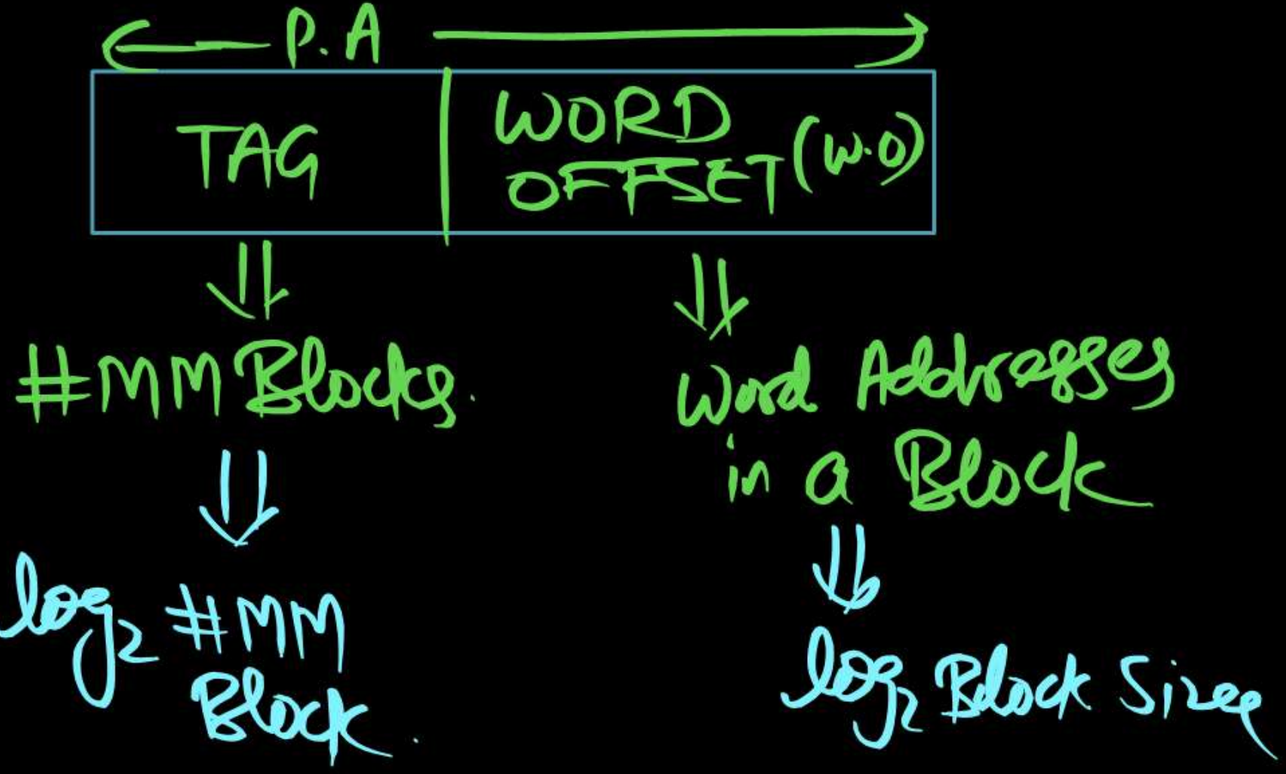
Memory Organization



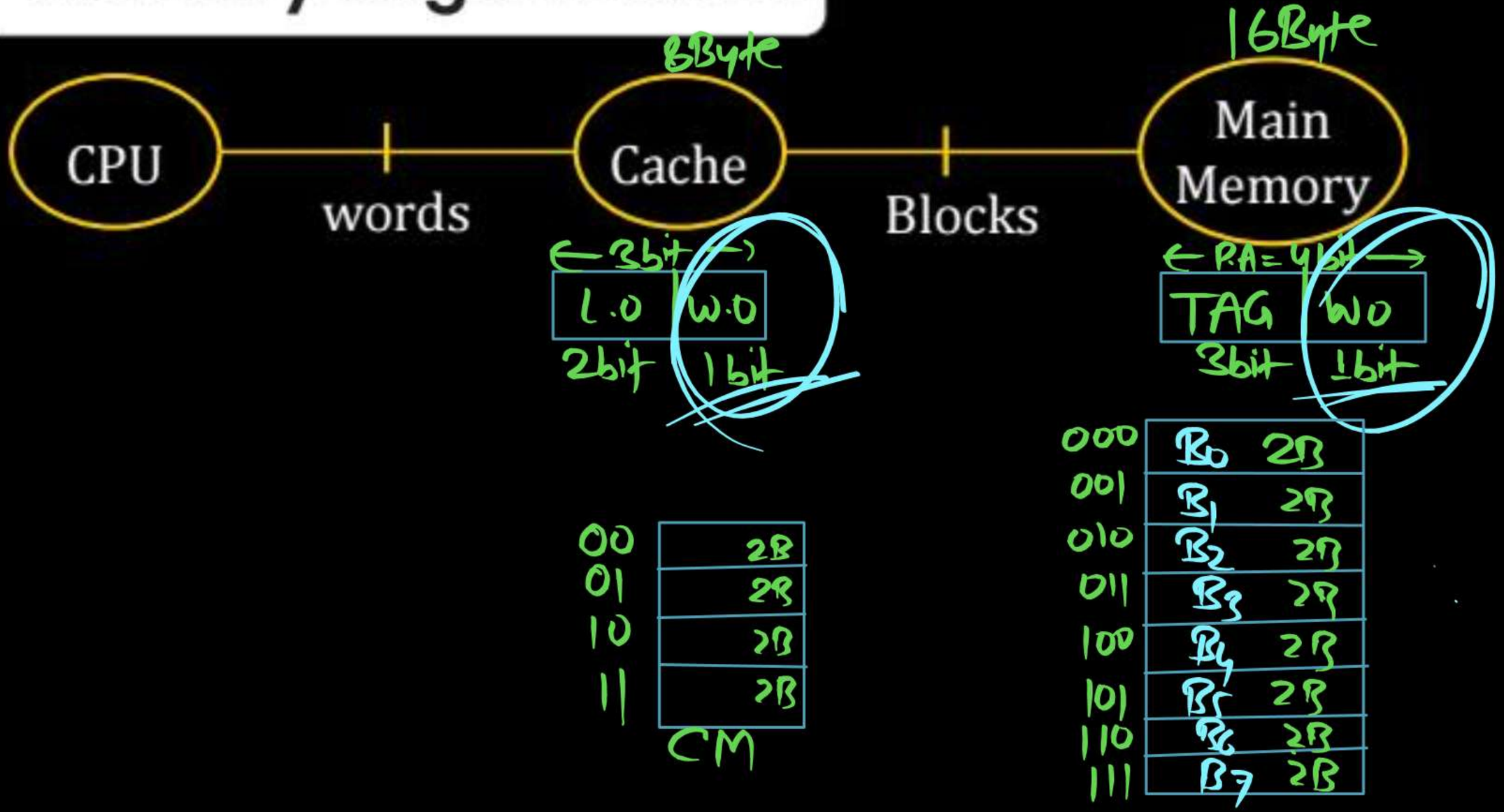
Cache Memory



Main Memory


P.A
TAG
WORD OFFSET (W.O)
#MM Blocks
log2 #MM Block
Word Addressable in a Block
log2 Block Size

Memory Organization



Mapping

The process of transfer the Data from Main Memory to Cache Memory is called mapping. There are 3 Type of Mapping Technique

1) Direct Mapping
2) Set Associative Mapping
3) Fully Associative Mapping

In this Process Complete Block is transfered from Main Memory to Cache Memory & Based On the CPU Request Addressed that Particular Data(word) given to CPU By Cache Memory.

Mapping Function

- Because there are fewer cache lines than main memory blocks, an algorithm, is needed for mapping main memory blocks into cache lines.
- Three techniques can be used:

Direct	Associative	Set Associative
• The simplest technique • Maps each block of main memory into only one possible cache lines.	• permits each main memory block to be loaded into any line of the cache. • The cache control logic interprets a memory address simply as a Tag a word field • To determine whether a block is in the cache, the cache control logic must simultaneously examine every line's Tag for a match	• A compromise that exhibits the strength of both the direct and associative approaches while reducing their disadvantage.

Direct Mapping

CM: Cache memory
MM: Main Memory

Cache Controller Interprets the Physical Address as:

$TAG = P.A - (L.O + W.O)$

OR

In Direct Mapping

$$TAG = \frac{MM\ Size}{CM\ Size}$$

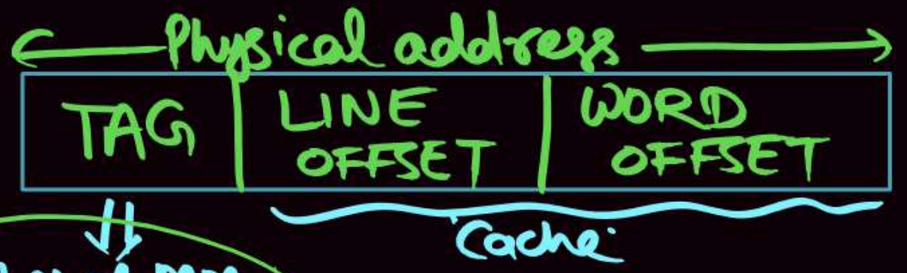
Number of CM Lines (Block)

Number of address in a block

$\log_2$ Block Size

Direct Mapping

Cache Controller Interprets the Physical Address as:



Number of MM Blocks are possible in One Cache Line But Not at the same time.

(eg) If Tag = 2 bit then

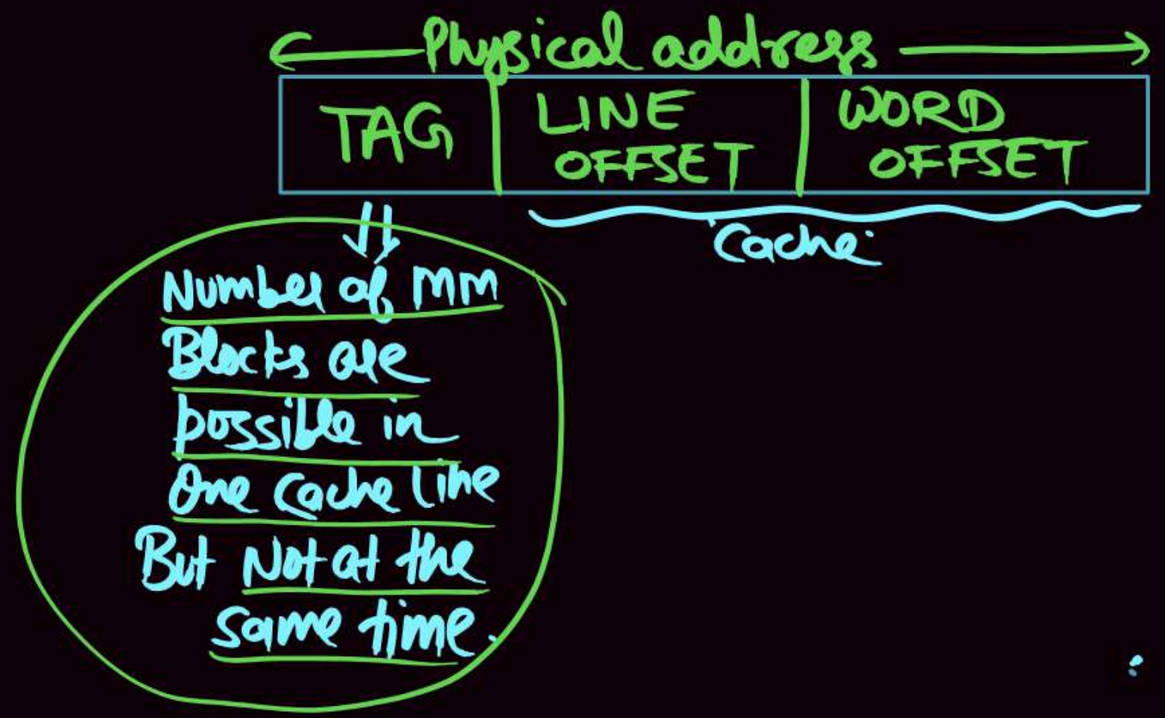
$$\#\text{MM Block in One Cache Lines} = 2^2$$

$$= 4 \text{ Block}$$

are Competing for One Cache Line.

Direct Mapping

Cache Controller Interprets the Physical Address as:



Direct Mapping

In this Direct Cache Controller interprets the CPU generated Request as follows:

$$\#\ \text{LINE} = \frac{\text{CM Size}}{\text{BLOCK Size}}$$

$$\text{Word Offset} = \log_2(\text{Block Size})$$

$$\text{LINE Offset} = \log_2(\#\text{LINE})$$

TAG = Physical Address - (Line offset + Word offset)

TAG Memory Size = #LINE's × Tag bits (Depend on the mapping technique)

Q.

Consider a Direct Mapping if the size of Cache memory is 512KB & Main Memory 512 KB & Cache line size (Block) is 64KB the calculate the number of bit required for

(i) P.A (ii) TAG (iii) B.O (iv) W.O

(v) #LINES (vi) TAG Memory Size.

Q.

Consider a Direct Mapping, Cache size = 64 byte, Line Size = 8 Byte. MM = 256 Byte then #bits for P.A, TAG, L.O, W.O Tag memory size?

Q. Consider a Direct Mapping, Cache size = 128 KB, Line Size = 64 Byte. Main Memory is 1MB then what is the line number of physical address $(ABCDE)_{16}$?

Q.

Consider a machine with a byte addressable main memory of 2^{20} bytes, block size pf 16 bytes and a direct mapped cache having 2^{12} cache lines. Let the addresses of two consecutive bytes in main memory be $(E201F)_{16}$ and $(E2020)_{16}$. What are the tag and cache line address (in hex) for main memory address $(E201F)_{16}$?

(a) E, 201 (b) F, 201 (c) E, E20 (d) 2, 01F

[GATE-2015]

Q.

Consider a machine with a byte addressable main memory of 2^{32} bytes divided into blocks of size 32 bytes. Assume that a direct mapped cache having 512 cache lines is used with this machine. The size of the tag field in bits is _____.

[GATE - 2017]

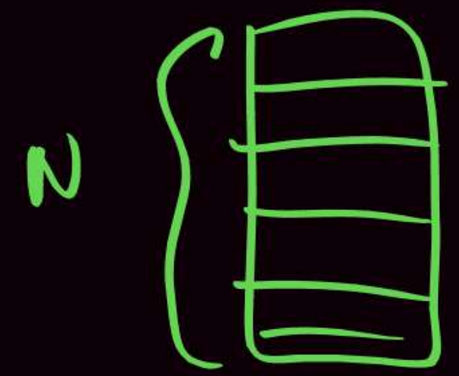
NAT

Consider a computer system with a byte-addressable primary memory of size 2^{32} bytes. Assume the computer system has a direct-mapped cache of size 32 KB (1 KB = 2^{10} bytes), and each cache block is of size 64 bytes.

The size of the tag field is _____bits.

[GATE-2021(Set-1)-CS: 1M]

Mapping



Direct Mapping

in DSA & Algo

Hash function

x MOD N

Direct Mapping

In the Direct Mapping we have a Mapping function to transfer [copying] the MM Block from MM to Cache Memory.

Mapping function :

MM Block No (MM Request) **MOD** Number of Cache Line $\Rightarrow$ Cache Address.

$K \text{ MOD } N = i$

K: MM Block Number
N: #Cache Lines (Block)
i: Cache Line Number (0 to N-1)

K MOD N means K^{th} MM Block (MM Request) are Load (transferred) into (K MOD N) Cache Line Number.

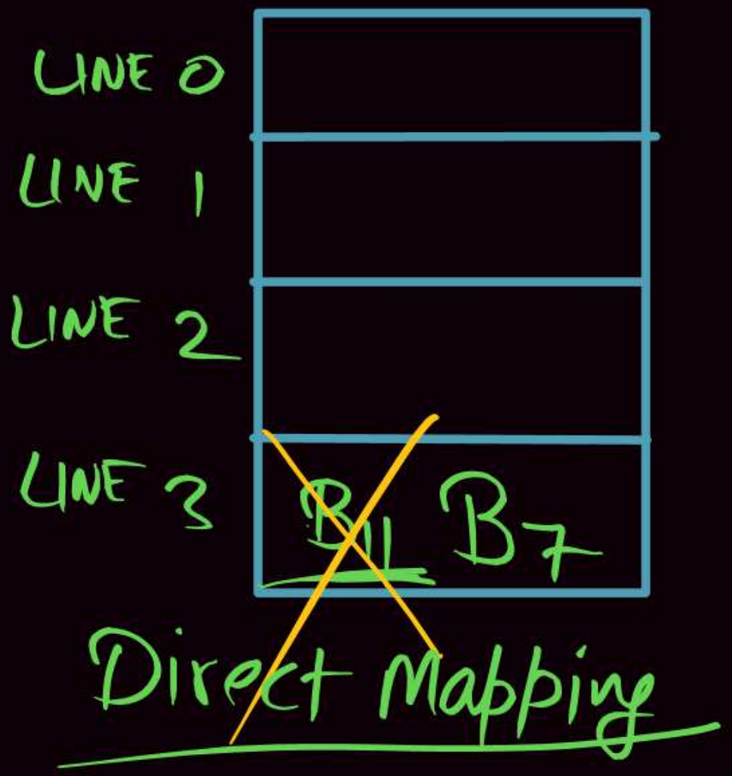
(eg) If we have 4 Cache Line & MM Request MM Block No is 11 (B_{11})

then MM Block No 11 are transferred (11 MOD 4) ⇒ CM LINE No 3.

(eg) If MM Block No. B_7 then this MM Block No B_7 Placed into Cache (7 MOD 4) = CM LINE NO = 3.

This means B_{11} & B_7 are transferred (Mapping) into Cache Line Number 3.

But out of B_{11} (or) B_7 Only One (Any One) Present in Cache at a time.


LINE 0
LINE 1
LINE 2
LINE 3
B7
Direct Mapping

1) Direct Mapping

In this Technique mapping function is used to transfer the data from Main Memory to Cache Memory. The Mapping Function is

Cache address = Main Memory request MOD # CM LINES

(Or)

K MOD N = i

K: MM Block No.
N: # of Cache Line
i: CM Line Number

eg1

Assume 50 MM Block.

10 Cache Lines.

K MOD N = i

Cache Address = MM Block No (MOD) #CM Lines

5 MM Blocks ⇒ 1 CM LINE

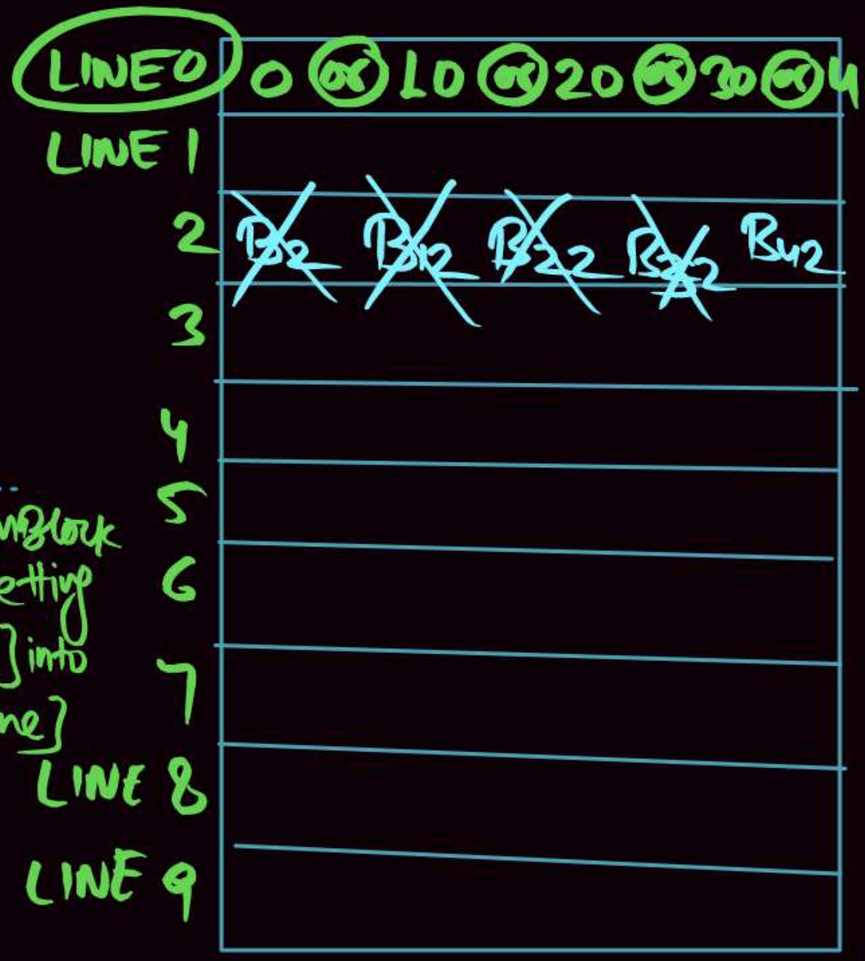
K MOD 10

0, 10, 20, 30, 40	MOD 10 Competing →	Cache Line '0'
1, 11, 21, 31, 41	MOD 10 →	LINE 1
2, 12, 22, 32, 42	→	LINE 2
8, 18, 28, 38, 48	MOD 10 →	LINE '8'
9, 19, 29, 39, 49	MOD 10 →	LINE '9'

Here 5 MM Block are competing [Residing] into 1 Cache Line]

5:1

Out of 5 MM Block, Only One MM Block Present in Particular Cache Line.



Cache

5 MM Block : LCM Line

5 : 1

$$TAG = \frac{MM\ Size}{CM\ Size}$$

$$= \frac{50}{10}$$

TAG = 5

5 MM Blocks are fighting for each Cache Line.
But Only One Block Present in Cache at a time.

es2

MM Size = 16 Block (B_0 to B_{15})

Cache = 4 Block (LINE)

k MOD N = i [k MOD 4 = i]

4 : 1

0 MOD 4 = 0

1 MOD 4 = 1

2 MOD 4 = 2

3 MOD 4 = 3

4 MOD 4 = 0

5 MOD 4 = 1

6 MOD 4 = 2

7 MOD 4 = 3

8 MOD 4 = 0

9 MOD 4 = 1

10 MOD 4 = 2

11 MOD 4 = 3

12 MOD 4 = 0

13 MOD 4 = 1

14 MOD 4 = 2

15 MOD 4 = 3

LINE 0	B_0 or B_4 or B_8 or B_{12}
LINE 1	B_1 or B_5 or B_9 or B_{13}
LINE 2	B_2 or B_6 or B_{10} or B_{14}
LINE 3	B_3 or B_7 or B_{11} or B_{15}

$$TAG = \frac{MM\,Size}{CM\,Size}$$

$$= \frac{16}{4}$$

TAG = 4 ie 4 MM Blocks are fighting for each Cache Line But Only One Block at a time.

B_0 B_4 B_8 B_{12}

Block No [0, 4, 8, 12] → are mapped → Cache LINE '0'

But only one block at a time.

B_1 B_5 B_9 B_{13}

Block No [1, 5, 9, 13] → are fighting (Residing) → Cache LINE '1'

But only one MM block at a time

B_2 B_6 B_{10} B_{14}

Block No [2, 6, 10, 14] → are Residing (transferred into) are put into → Cache LINE '2'

But only one MM block at a time

B_3 B_7 B_{11} B_{15}

Block No [3, 7, 11, 15] → are put into → Cache LINE '3'

But only one MM Block at a time.

eg 3

MM Block = 8 Block [B_0 to B_7)

CM Block = 4 LINE (Block)

Mapping Function = $k \bmod N = i$

MM Block NO $\xrightarrow{\text{MOD}}$ # CM LINE = Cache Address [CM LINE NO]

2 : 1

k MOD 4

(B_0) 0 MOD 4 = 0

(B_1) 1 MOD 4 = 1

(B_2) 2 MOD 4 = 2

(B_3) 3 MOD 4 = 3

(B_4) 4 MOD 4 = 0

(B_5) 5 MOD 4 = 1

(B_6) 6 MOD 4 = 2

(B_7) 7 MOD 4 = 3

Block 0 → LINE '0'

Block 4 → LINE '0'

B_1 → LINE 1

B_5 → LINE 1

B_2 → LINE '2'

B_6 → LINE '2'

B_3 → LINE '3'

B_7 → LINE '3'

LINE 0	B_0 or B_4
LINE 1	B_1 or B_5
LINE 2	B_2 or B_6
LINE 3	B_3 or B_7

$$\text{TAG} = \frac{\text{MM Size}}{\text{CM Size}}$$

$$= \frac{8}{4} = 2$$

Tag = 2 ie 2 MM Blocks are fighting for each Cache Line

But Only 1 MM Block present in Cache at a time.

$K \bmod N = i$ (CM LINE NO.)

$K \bmod 4 = i$

B_0 & B_4 are Mapped into CM LINE '0'

B_1 & B_5 are Mapped into CM LINE '1'

B_2 & B_6 are Mapped into CM LINE '2'

B_3 & B_7 are Mapped into CM LINE '3'

But only one MM Block are Mapped on each Cache Line According Mapping function.

Q) Before Mapping Why we do Memory Org for Both MM & CM ?

Q) MM Size = 16 Byte & CM Size = 8 Byte & Block Size = 2 Byte

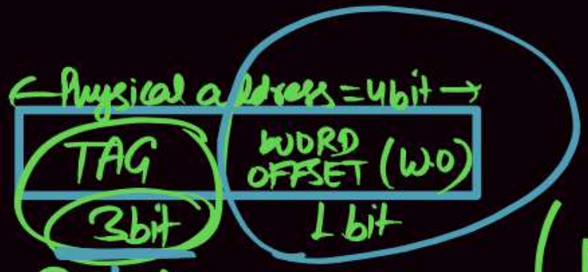
(i) Before Mapping Show the format of Cache & MM ?

(ii) In Direct Mapping Show MM format ?

Before Mapping

Cache

Before Mapping

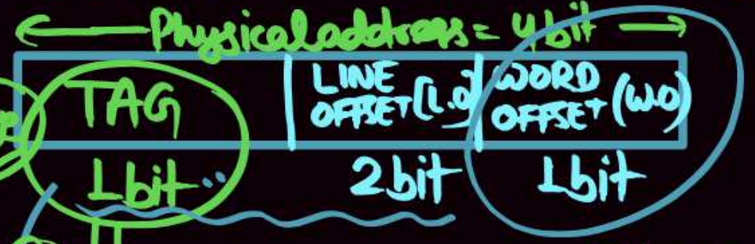


(MM Blocks)

Here TAG bit Indicates the Number of MM Blocks.

Cache Controller Interpret the physical address as.

In Direct Mapping



$TAG = \frac{MM\ Size}{CM\ Size}$; $\frac{16B}{8B} = 2$ Tag.

Tag bit = 1bit

TAG = PA − (L.O + W.O)
= 4 − (2+1)
= 1bit

① Indicate #MM Blocks are Possible (Fighting) in each Cache Line.
But Not at the same time.

② Here Tag bit = 1bit ⇒ $2^1 = 2$ Tag → 2 MM Blocks are Possible (Fighting) in each Cache Line But one at a time Block.

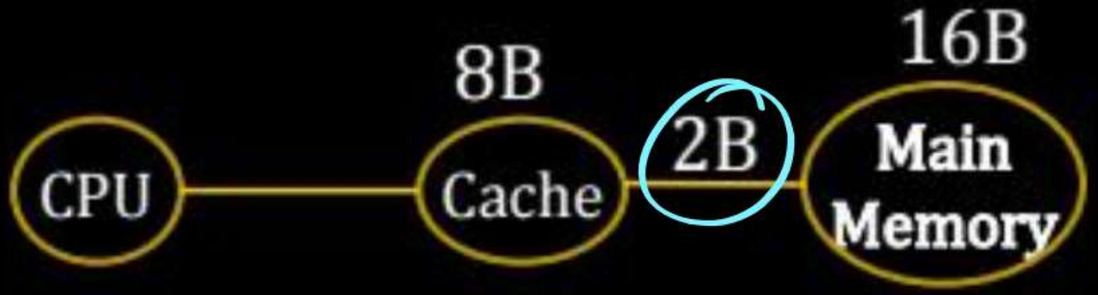
Tag = 2

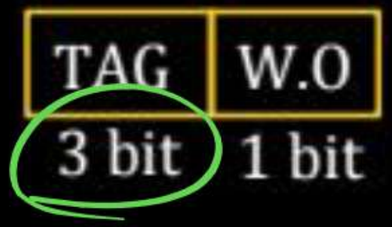
Note: With the Help of this Tag bit are used to Identify [checking] is there is a Cache Hit or Miss.

Q If Number of Tag bits for Cache block x is 4 bits what is the meaning of it ?

Soln Tag = 4 bit

means 2^4 MM Blocks are completing for Cache Line Number x (Position x).





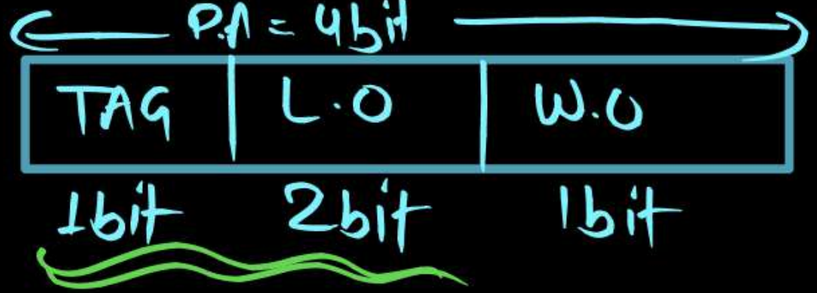
CM Line = 4

mm Blocks = 8

K MOD N = i

K MOD 4 = i

If Direct Mapping Show the Physical address by Cache Controller:



$\#TAG = \frac{mm\ size}{CM\ size} = \frac{16B}{8B} = 2\ Tags$

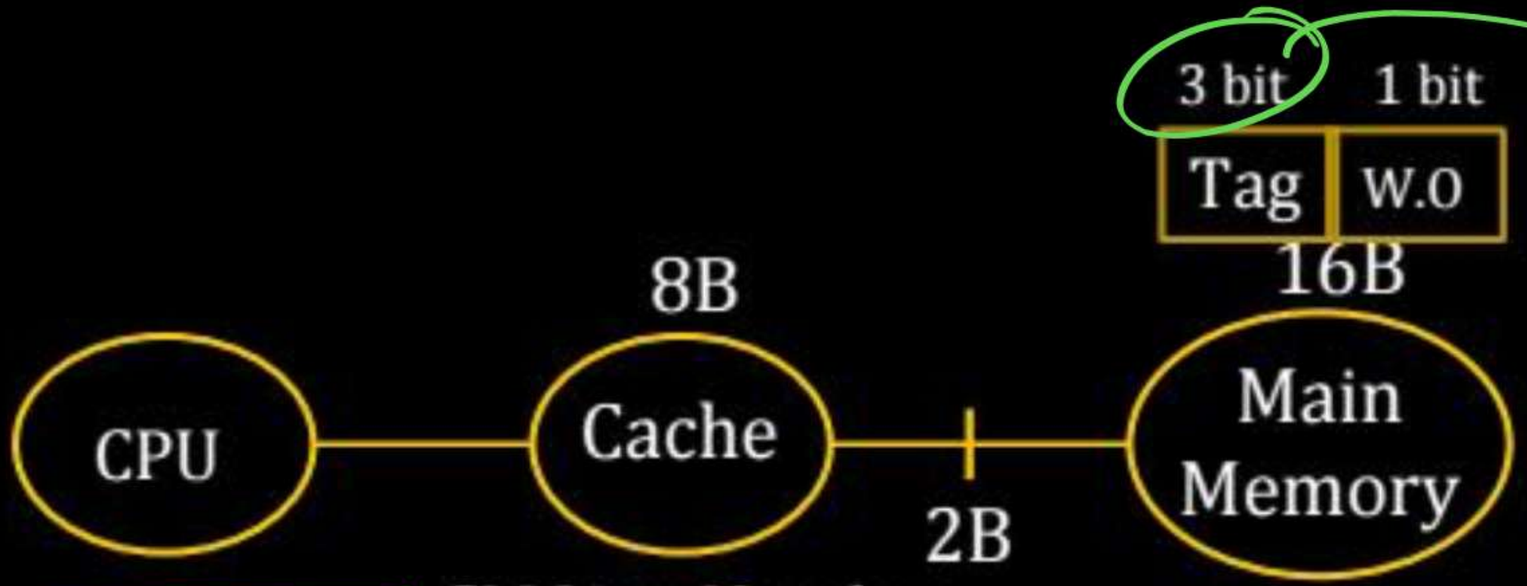
Tag bit = 1bit

$\#LINE = \frac{CM\ size}{Block\ size} = \frac{8B}{2B} = 4\ LINE$

L.O = 2bit

TAG = P.A - [L.O + W.O]

= 4 - [2+1] = 1bit



8 MM Block

4 CM LINES

CM Line Number

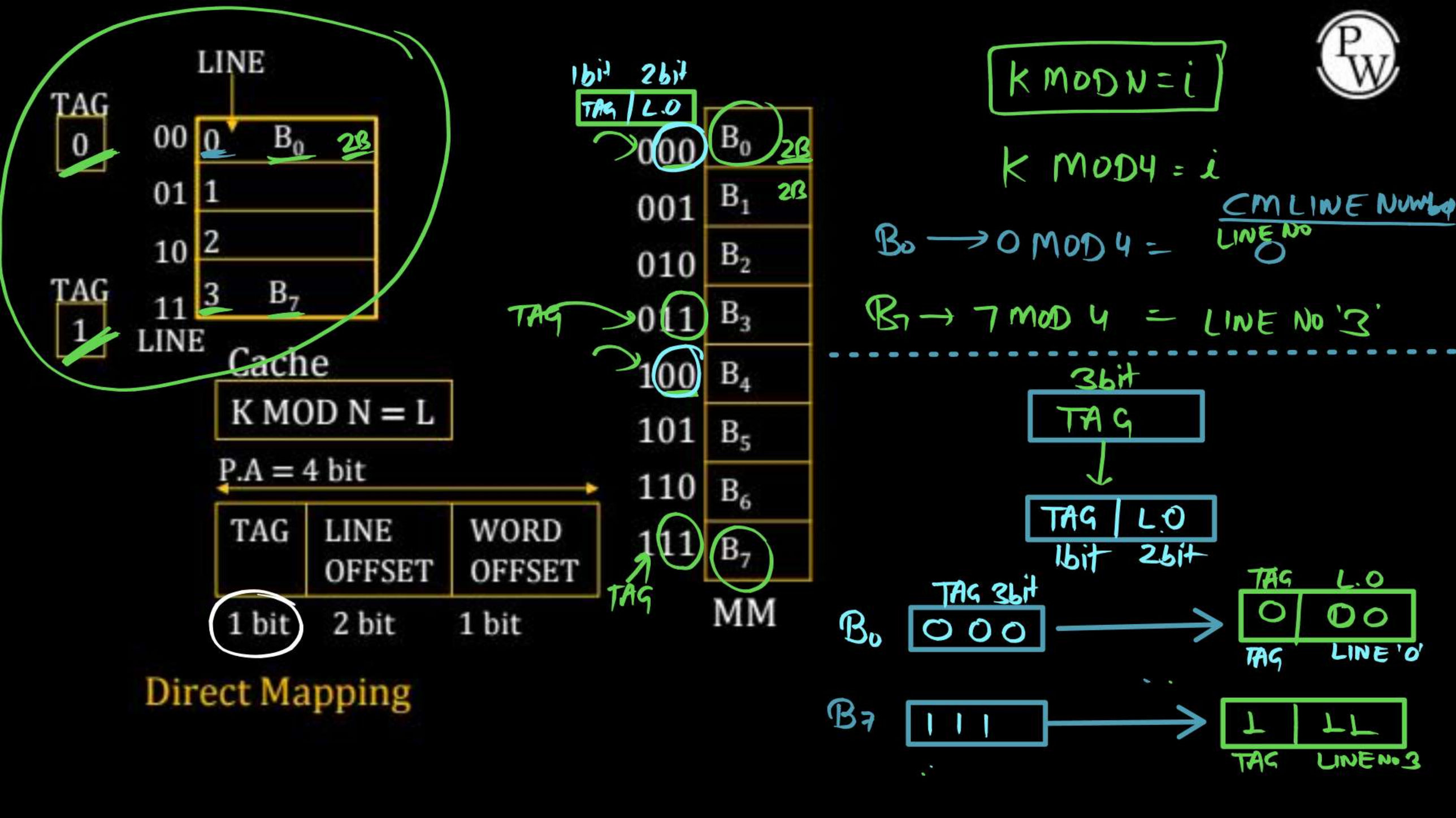
K MOD N = i

0 MOD 4 = 0
1 MOD 4 = 1
2 MOD 4 = 2
3 MOD 4 = 3
4 MOD 4 = 0
5 MOD 4 = 1
6 MOD 4 = 2
7 MOD 4 = 3

CM

B_0 & B_4: LINE 0
B_1 & B_5: LINE 1
B_2 & B_6: LINE 2
B_3 & B_7: LINE 3

→ But Only One MM Block Present in CM Line at a time.


LINE
TAG
0
00
0
B0
2B
01
1
10
2
TAG
1
11
3
B7
LINE
Cache
K MOD N = L
P.A = 4 bit
TAG
LINE OFFSET
WORD OFFSET
1 bit
2 bit
1 bit
1bit 2bit
TAG | L.O
000
B0
001
B1
010
B2
011
B3
100
B4
101
B5
110
B6
111
B7
TAG
MM
Direct Mapping
K MOD N = i
K MOD 4 = i
CM LINE Number
LINE No
B0 → 0 MOD 4 = LINE No '0'
B7 → 7 MOD 4 = LINE No '3'
3bit
TAG
TAG | L.O
1bit 2bit
TAG 3bit
B0
000
TAG L.O
0 00
TAG
LINE '0'
B7
111
1 11
TAG
LINE No 3

MM Block	Direct Mapping	CM LINE
TAG LINE 0 00 $B_{0_{[000]}}$	K MOD N = i 0 MOD 4 = '0'	LINE '0'
TAG LINE 1 11 $B_{7_{[111]}}$	K MOD N = i 7 MOD 4 = '3'	LINE '3'

---

1 00 $B_4[100]$ — K MOD N = i, 4 MOD 4 = '0' → LINE '0'

TAG L.O

0 11 $B_3(011)$ — 3 MOD 4 → LINE 3

TAG

Tag Memory Size = #LINE's × Tag bits [Depends On the Mapping technique]

In the above example: # LINE = 4
Tag bit = 1 bit
(Direct Mapping)

Tag Memory Size = 4 × 1 = 4 bits

Q. How to check Cache Hit or Cache Miss ?

Practically ?

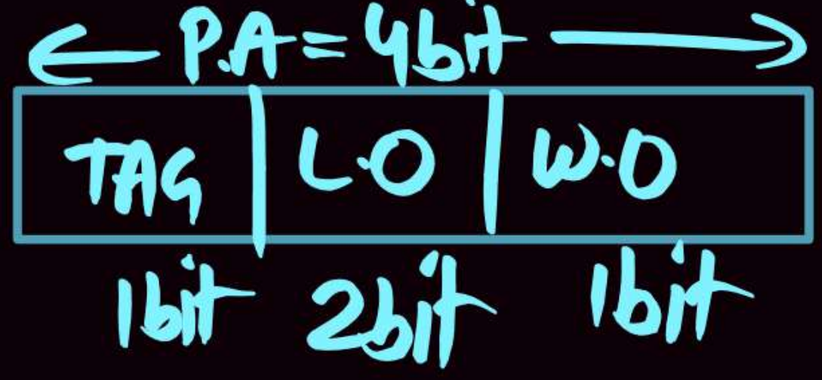
Sol^n

MM = 16B

CM = 8B

Block Size = 2B

Direct Mapping Cache



Consider the following program

I_1: MOV r_0 [0000] → B_0

I_2: MOV r_1 [1000] → B_4

I_3: ADD $r_0 r_1$

CPU generate Memory Request Go to Cache.

I_1: 0000 RD (Memory Read)

Direct Cache.

TAG	L.O	W.O
1bit	2bit	1bit

Known Format

TAG	L.O	W.O
0	00	0

L.O

TAG		
0	00	B_0
	01	
	10	
1	11	B_7

2B

TAG is 1bit [0/1] means 2 tags

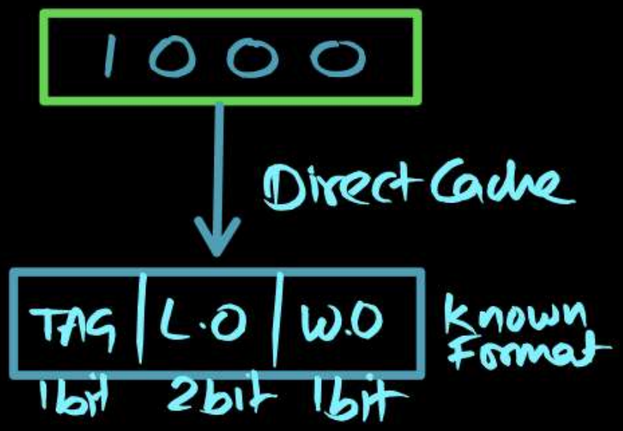
⇓

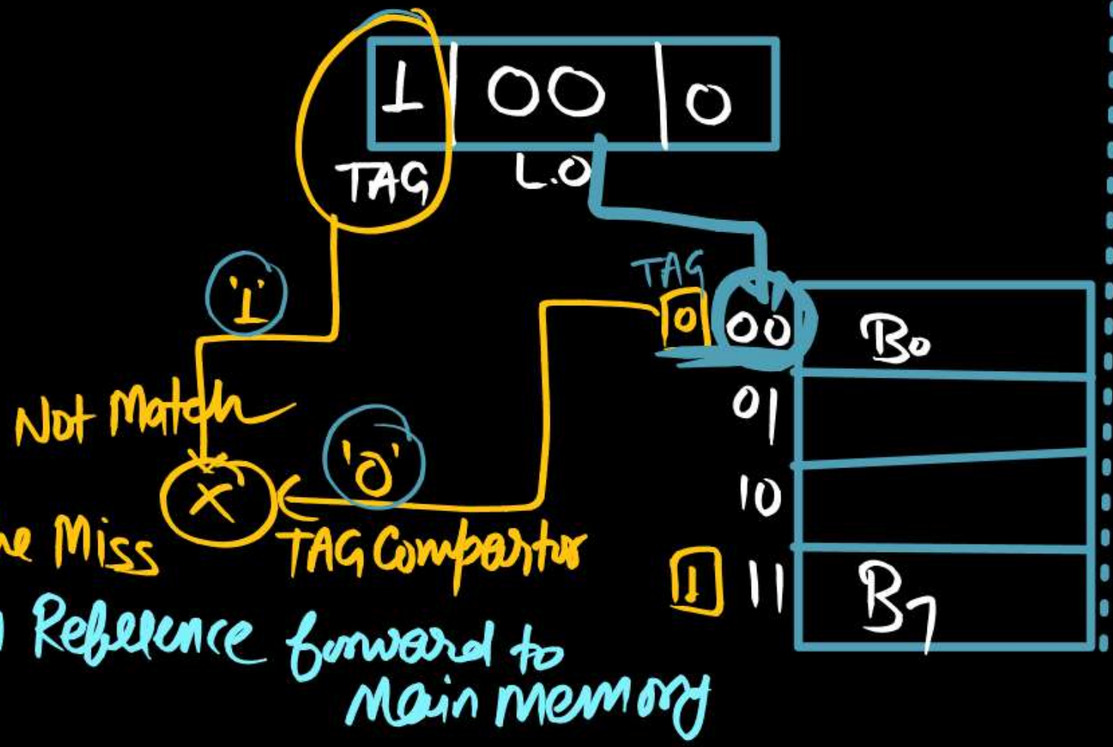
2MM Blocks are fighting for each Cache Line.

Match then Cache Hit. Comparator

Data given from Cache to CPU.

I_2 : MOV R1, 1000 CPU generate the Memory Request

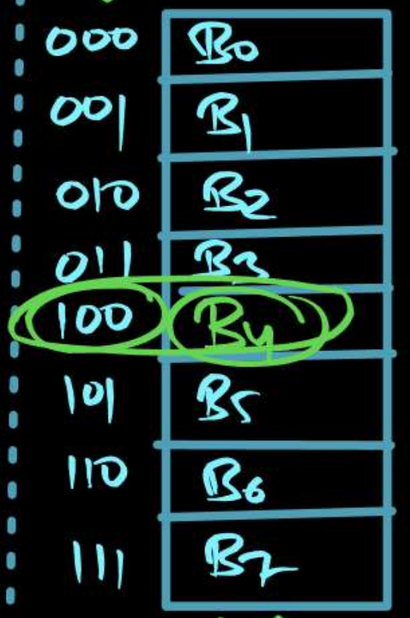


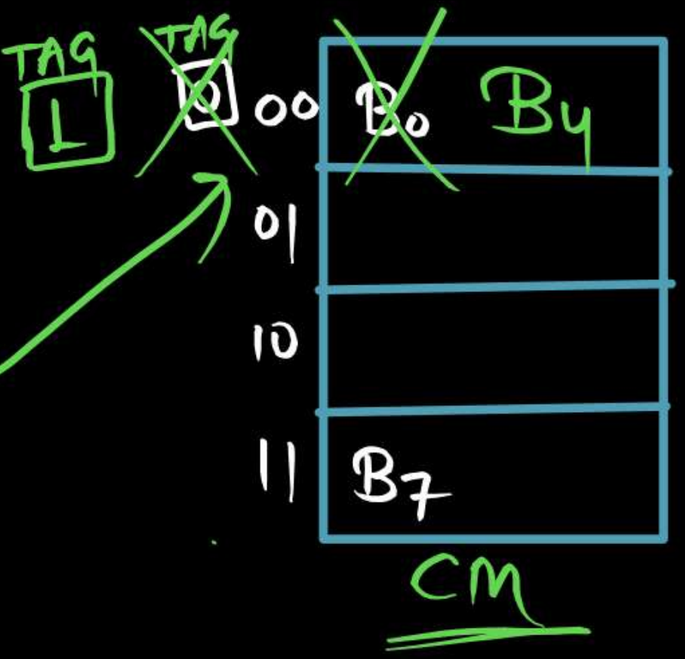


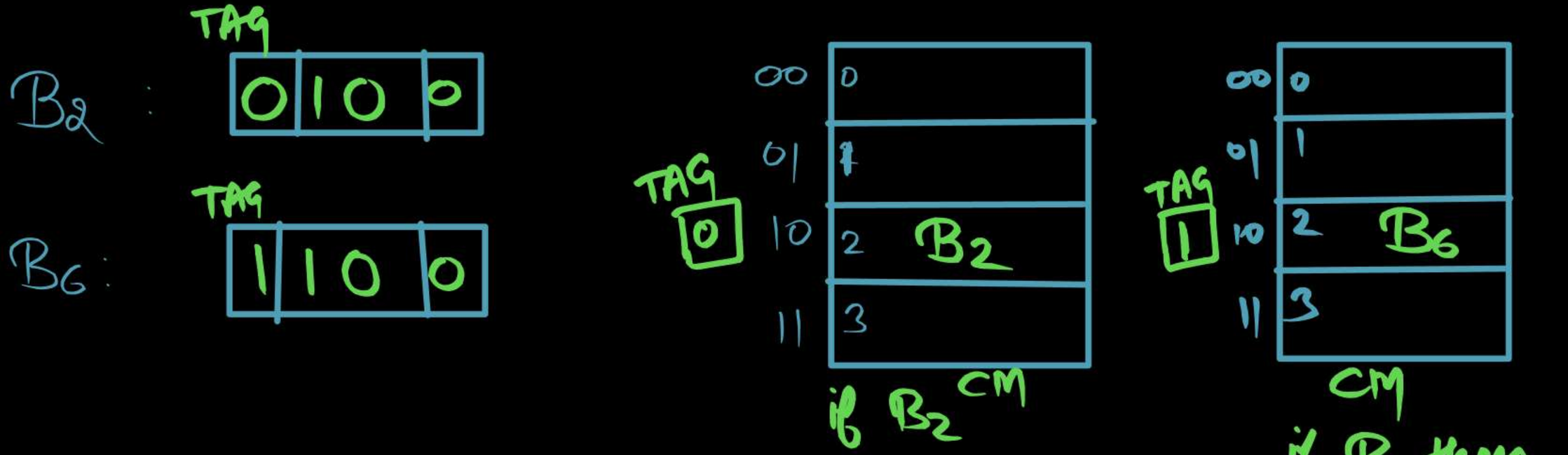
If Cache Miss then from mm to CPU Block Transferred

k MOD N = i

4 MOD 4 = CM LINE '0'






TAG
B_2 : 0 1 0 0
TAG
B_6 : 1 1 0 0
TAG
0
00 0
01 1
10 2 B_2
11 3
if B_2 CM
TAG
1
00 0
01 1
10 2 B_6
11 3
CM
if B_6 then

Complete Working

- CPU generated Request firstly go the the Cache Memory.
- [Direct Mapped Cache] Cache Controller interpreted the CPU generated Request [Physical address) as:

- Line offset field is Directly Connected with address logic of the Cache Memory, so Respective Cache Line will be enabled
- The existing Tag in enabled line is Compared with CPU generated TAG with the Help of TAG Comparator.

When Both Tag are Match [CPU generated Tag & enabled Cache Line Tag] then its called Cache Hit [operation is Hit (ie Data Present in Cache)]

Note So Based on the Word offset [0th Byte or 1st Byte in last example], Respective Word or Byte (from that Block) is given from Cache to CPU.

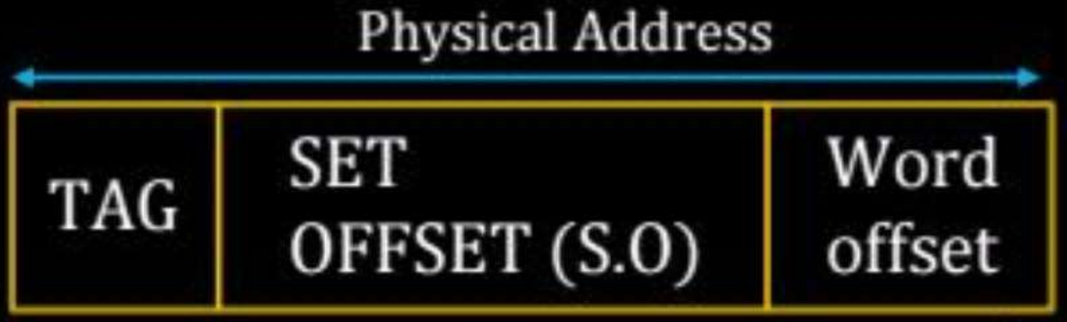
- If Both Tag are Not Matching then its Cache Miss, then Reference will Forward to Main Memory.
- According to mm format, Respective MM Block will be enabled & By Using Mapping function [K MOD N] [eg By in Previous eg] Complete Block transfered from MM to Cache, then Respective Word/Byte given from Cache to CPU.

Here Cache Controller Maintain Some extra bits for each Cache Line & Stored in Tag Directory | Tag Array.

Tag Memory Size = Number of LINES X Tag bits
(Tag + Extra bit if given)

2) Set Associative Cache

SET associative cache controller, Interpreter the CPU generated request as follows:



Word Offset = $\log_2$ Block Size

$$\#\text{lines} = \frac{\text{CM Size}}{\text{Block Size}}$$

$$\#\text{SETS} = \frac{\#\text{Lines}}{\text{N-way}}$$

SET OFFSET = $\log_2$ #SETS

TAG = Physical address - (S.O + W.O)

Q.

Consider a 2-way set associative if the size of cache memory is 512KB & Main Memory 512MB & Cache line size is 64KB then calculate the Number of bit Required for

1. (i) P.A (ii) TAG (iii) L.O (iv) W.O

(2) #lines (3) TAG memory size

Q. Consider a 2-way set associative Cache Size = 256 KB, Line size = 32 Byte, MM = 1MB, then what is the set number of Physical address $(ABCDE)_{16}$?

Q. The main memory of a computer has 2 cm blocks while the cache has 2 c blocks. If the cache uses the set associative mapping scheme with 2 blocks per set, then block k of the main memory maps to the set. [GATE - 1999]

(a) (k mod m) of the cache
(b) (k mod c) of the cache
(c) (k mod 2 c) of the cache
(d) (k mod 2 cm) of the cache

Q.

Consider a 4-way set associative cache consisting of 128 lines with a line size of 64 words. The CPU generates a 20-bit address of a word in main memory. The number of bits in the TAG, SET and WORD fields are respectively.

(a) 9, 6, 5 (b) 7, 7, 6 (c) 7, 5, 8 (d) 9, 5, 6

[GATE - 2007]

Q.

[Common Data for this and next question]
Consider a computer with a 4-way set-associative mapped cache of the following characteristics; a total of 1 MB of main memory, a word size of 1 Byte; a block size of 128 words and a cache size of 8 KB.

The number of bits in the TAG, SET and WORD fields, respectively are:

(a) 7, 6, 7 (b) 8, 5, 7 (c) 8, 6, 6 (d) 9, 4, 7

Q.

[Common Data]
Consider a computer with a 4-way set-associative mapped cache of the following characteristics; a total of 1 MB of main memory, a word size of 1 Byte; a block size of 128 words and a cache size of 8 KB.

While accessing the memory location 0C795H by the CPU, the contents of the TAG field of the corresponding cache line is

(a) 000011000 (b) 110001111
(c) 00011000 (d) 110010101 [GATE – 2008]

Q.

A 4-way set-associative cache memory unit with a capacity of 16KB is built using a block size of 8 words. The word length is 32 bits. The size of the physical address space is 4 GB. The number of bits for the TAG filed is

[GATE - 2014]

Q. The size of the physical address space of a processor is 2^P bytes. The word length is 2^w bytes. The capacity of cache memory is 2^N bytes, the size of each cache block is 2^M words. For a k - way set - associative cache memory, the length (in number of bits)of the tag field is

(a) $P - N - \log_2 K$

(b) $P - N + \log_2 K$

(c) $P - N - M - W - \log_2 K$

(d) $P - N - M - W + \log_2 K$

[GATE - 2018]

Set Associative Mapping function

Cache Set Address = MM Request MOD #SET is in Cache

(OR)

K MOD S = i

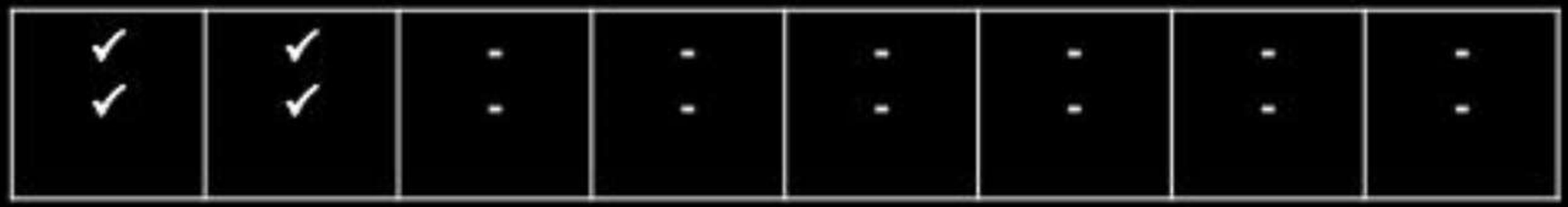
k: MM Block Number
s: # Cache Set
i: Cache Set Number

Example1: # Line's = 16 & 2 way set Associative

$$\#\,SET = \frac{\#LINES}{N\text{-way}} = \frac{16}{2} = 8$$

#SET = 8 S = 8

K MOD 8 = i



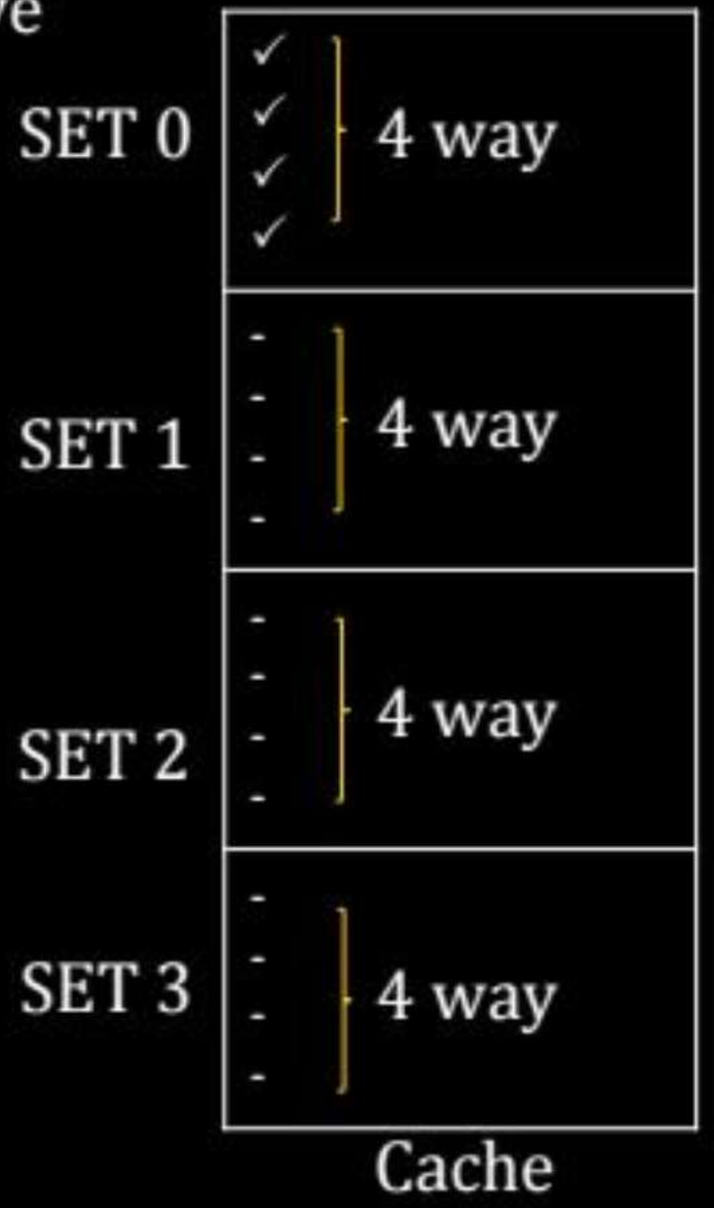
K MOD 8 = i 0 - 7

Example2:

#LINE = 16 & 4 way set Associative

$\#\ SET = \frac{16}{4} \Rightarrow 4$ **S = 4**

K MOD 4 = i



Example3:

#LINE = 16 & 8 way set Associative

$\#\ SET = \frac{16}{8} \Rightarrow 2$ S = 2

K MOD 2 = i

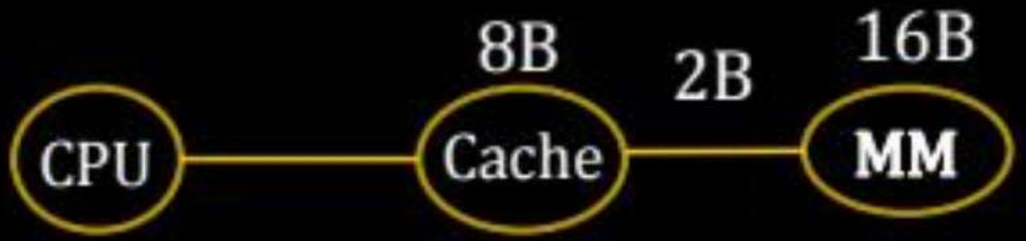
Example4:

#LINE = 16 & 16 way set Associative

$\#\ SET = \frac{16}{16} \Rightarrow 1$ S = 1

K MOD 1 = i

Fully
Associative
(Whole cache as a set)

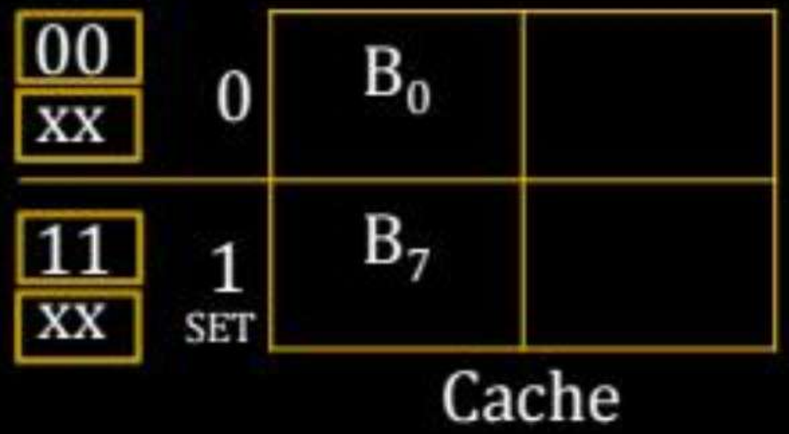


2 way set associative

K MOD S = i

P.A

TAG	S.O	W.O
2 bit	1 bit	1 bit



000	B_0
001	B_1
010	B_2
011	B_3
100	B_4
101	B_5
110	B_6
111	B_7

$B_{0[000]}$ →

TAG	S.O
00	0
2 bit	1 bit

→ SET '0'

$B_{7[111]}$ →

TAG	S.O
11	1

→ SET '1'

MM BLOCK Mapping Tech. CM SET

$B_{0[000]}$ $\xrightarrow[0 \text{ MOD } 2 = \text{'0'}]{K \text{ MOD } S = i}$ SET '0'

$B_{7[111]}$ $\xrightarrow[7 \text{ MOD } 2 = \text{'1'}]{K \text{ MOD } S = i}$ SET '1'

Tag Memory Size = #SETS × #LINES In a each set × Tag bits

Example: # SET = 2 # LINE in Each set =2 TAG = 2 bit

Tag Memory size = 2 × 2 × 2 = 8 bits

1) In direct Mapping

Hit Latency = Latency of Tag comparator

2) In Set Associative Mapping

Hit Latency = Latency of Tag comparator + Latency of Multiplexer

Q.

[Common Data for this and next question]
A computer has a 256 Kbyte, 4-way set associative. Write back data cache with block size of 32 Bytes. The processor sends 32 bit address to the cache controller. Each cache tag directory entry contains, in addition to address tag, 2 valid bits, 1 modified bit and 2 replacement bit.

The number of bits in the tag field of am address is
(a) 11 (b) 14 (c) 16 (d) 27

[GATE - 2012: 2 Marks]

Q.

[Common Data from previous question]
A computer has a 256 Kbyte, 4-way set associative. Write back data cache with block size of 32 Bytes. The processor sends 32 bit address to the cache controller. Each cache tag directory entry contains, in addition to address tag, 2 valid bits, 1 modified bit and 2 replacement bit.

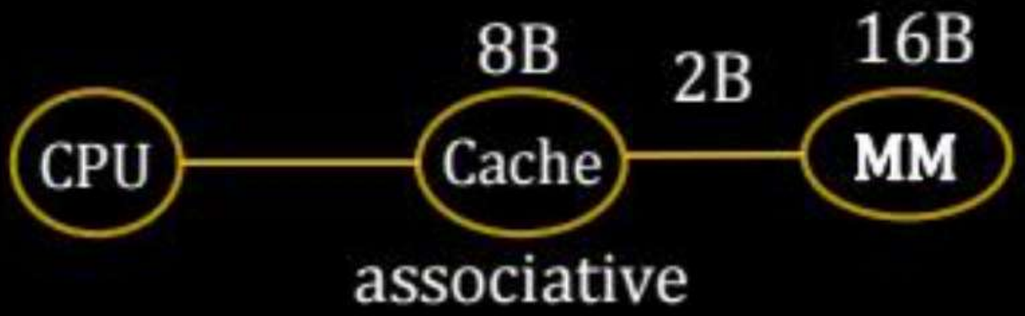
[GATE - 2012: 2 Marks]

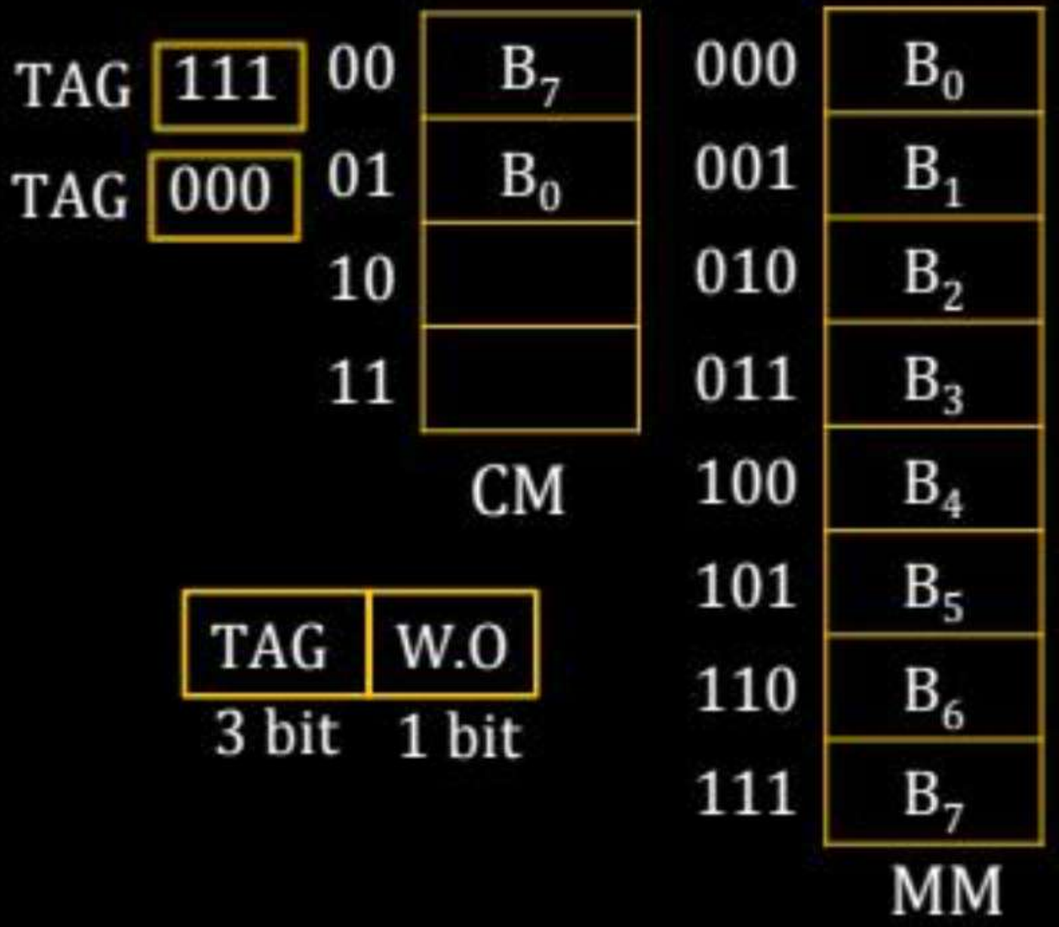
The size of the cache tag directory is

(a) 160 Kbits
(b) 136 Kbits
(c) 40 Kbits
(d) 32 Kbits

3) Associative Mapping

- In this no mapping function is used to transfer the data from MM to CM.
- No Conflict Miss





MM Block	Associative Mapping	CM Line
$B_{7[111]}$	No mapping Function →	Any line
$B_{0[000]}$	No mapping Function →	Any line

Tag Memory Size = # LINES × Tag bits

Example: #LINE =4 & Tag bits =3

Tag Memory Size = $4 \times 3 = 12$ bits

Q.

Consider fully associative cache consists 8 Block & MM contain 128 Block & Request made by the CPU:
119, 84, 37, 0, 16, 0, 84, 120, 121, 93, 37, 0, 43, 39, 47, 48.
Calculate # of compulsory & Capacity miss?

0	119 43
1	84 39
2	37 47
3	0 48
4	16
5	120
6	121
7	93

Hit: 0, 84, 37, 0

#Hits: 4

Compulsory Miss = 8

Capacity Miss = 4

Total Miss = 12

THANK YOU!