Module 5: Memory Organization

- Memory System: Basic Concepts,
- Memory hierarchy,
- Main Memory,
- Secondary storage,
- Cache memory mapping,
- Cache coherence.

Memory System: Basic Concepts

Memory System: Basic Concepts

- In a computer system, the **CPU (Processor) is like the Hare,** extremely fast and capable of executing millions of instructions per second.
- Meanwhile, **memory (RAM, Cache, Hard Drive) is like the Tortoise**, slower but essential for storing and retrieving data.
- The CPU (Hare) runs fast and wants to execute instructions quickly.
- However, it needs data from memory (Tortoise), which takes time to fetch.
- The Hare runs quickly but then stops and sleeps, wasting time.
- Similarly, a processor can be extremely fast, but if it has to wait for data from slow memory (RAM or even a hard drive), it idles, leading to performance bottlenecks.

Memory System: Basic Concepts

- Modern computers use a memory hierarchy, keeping frequently used data in cache (L1, L2, L3) to minimize waiting time.
- The CPU (Hare) should not always wait for slow memory (Tortoise) but instead anticipate data needs using prefetching and caching mechanisms.

Fast processor alone does not guarantee high performance—efficient memory management is also crucial factor.

Memory System: Basic Concepts

- A memory system in a computer is responsible for storing and retrieving data.
- It consists of various types of memory units organized in a hierarchy based on speed, cost, and capacity.

Key Characteristics of Memory:

- Volatility – Determines whether memory retains data after power loss (RAM vs ROM).
- Access Time – Time required to access data.
- Capacity – Amount of data a memory unit can store.
- Cost per Bit – Expense of storing a single bit of data.
- Read/Write Operations – Whether the memory is read-only (ROM) or read/write (RAM).

Memory hierarchy

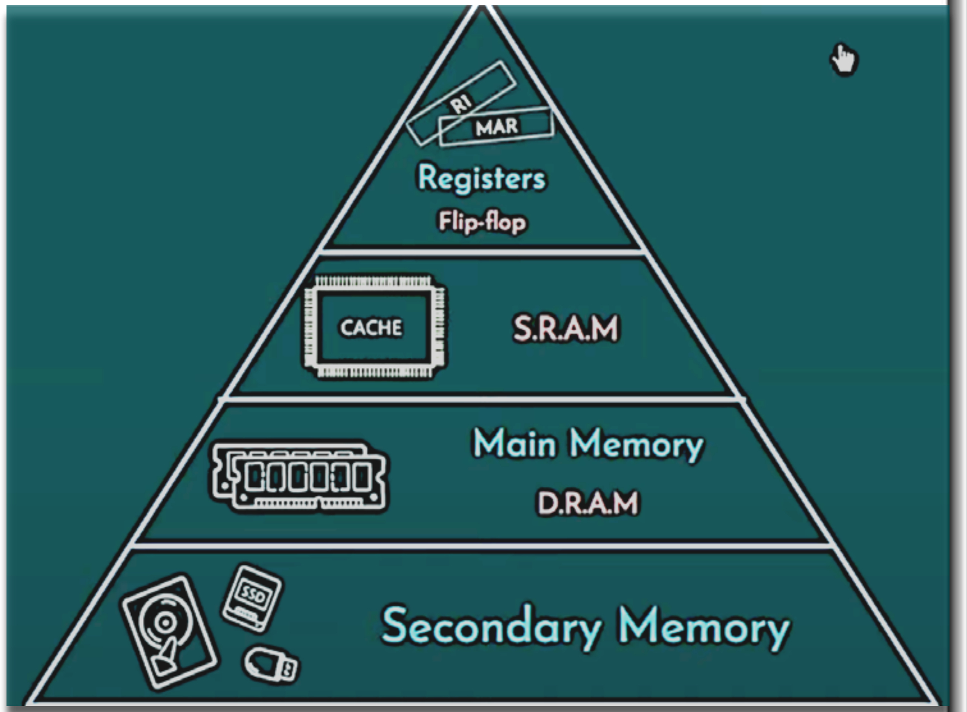
- Memory hierarchy is designed to balance speed, cost, and size to optimize performance.

•Internal Memory (Primary Memory).

Registers, main memory, and cache. Internal memory is directly accessible by the processor.

•External Memory (Secondary Memory).

Secondary storage (HDDs, SSDs) and tertiary storage (magnetic disk, magnetic tape, and optical disk. This is peripheral storage accessible by the processor via an I/O Module.



Memory hierarchy

Memory Speed vs Capacity Trade-off

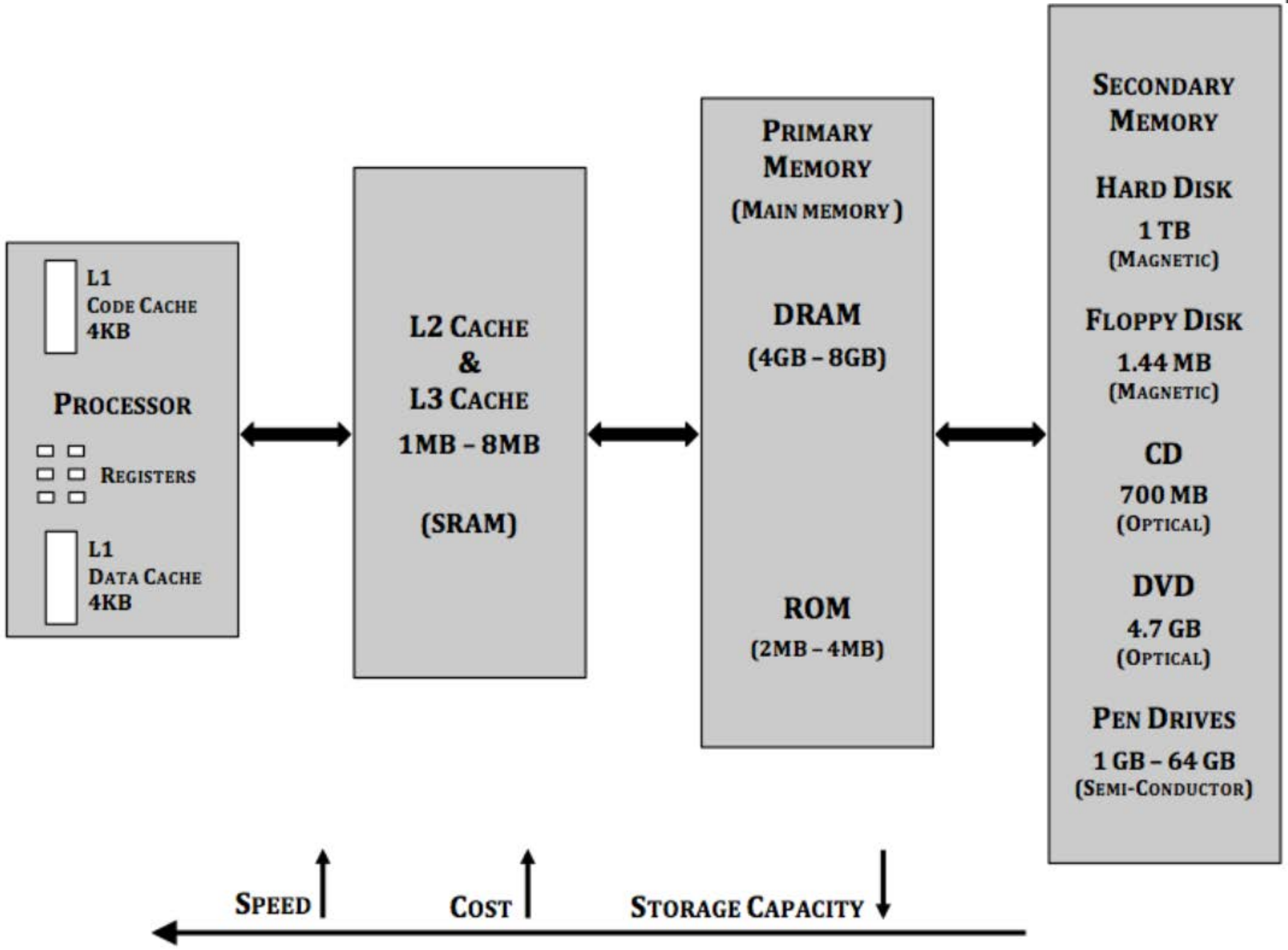
•**Registers** → Fastest but smallest in size.

•**Cache Memory** → Faster than RAM, used for frequently accessed data.

•**Main Memory (RAM)** → Medium speed and medium capacity.

•**Secondary Storage (HDD/SSD)** → Slower than RAM but higher capacity.

•**Tertiary Storage** → Slowest but largest in capacity.



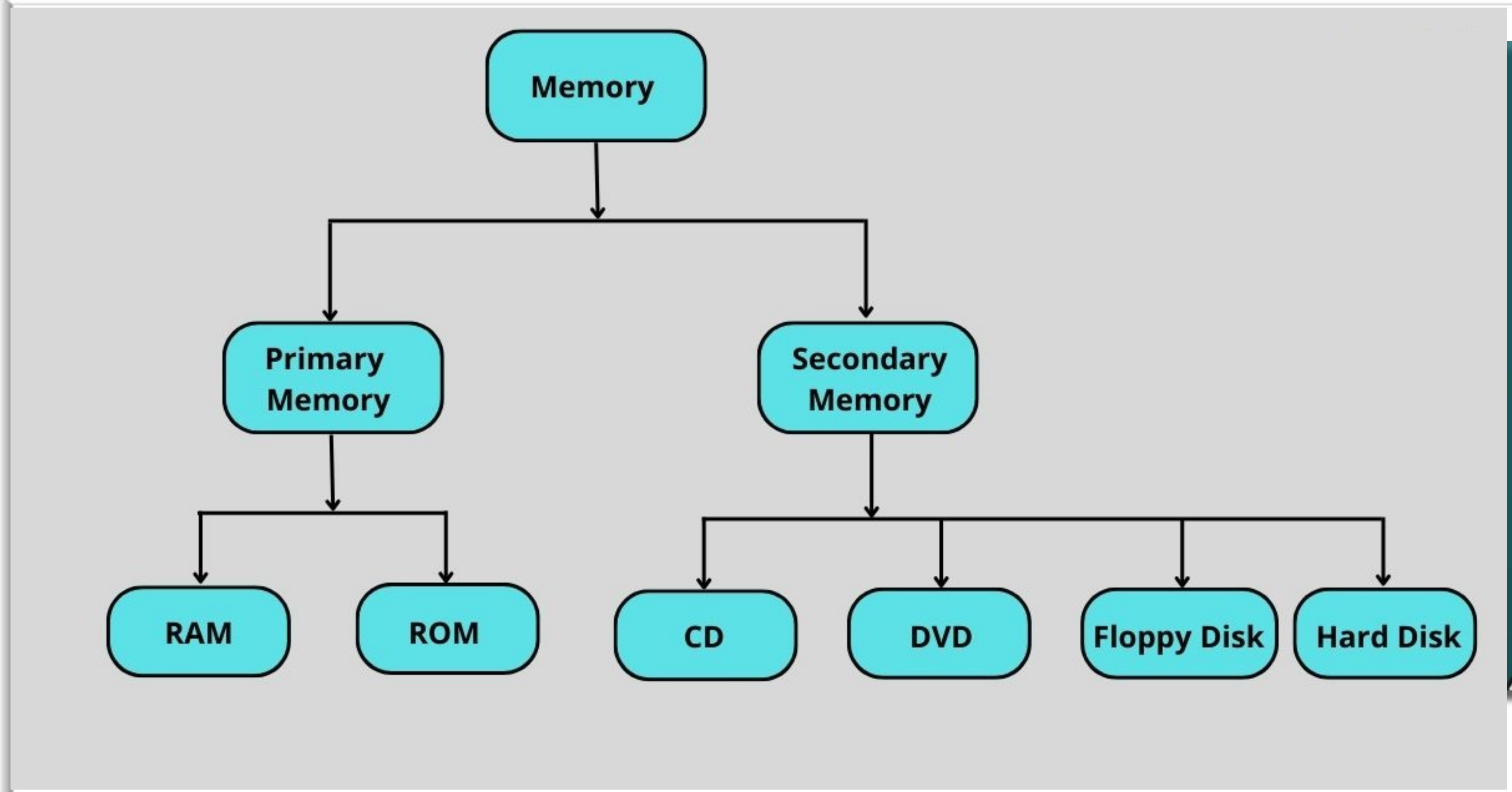
Memory hierarchy

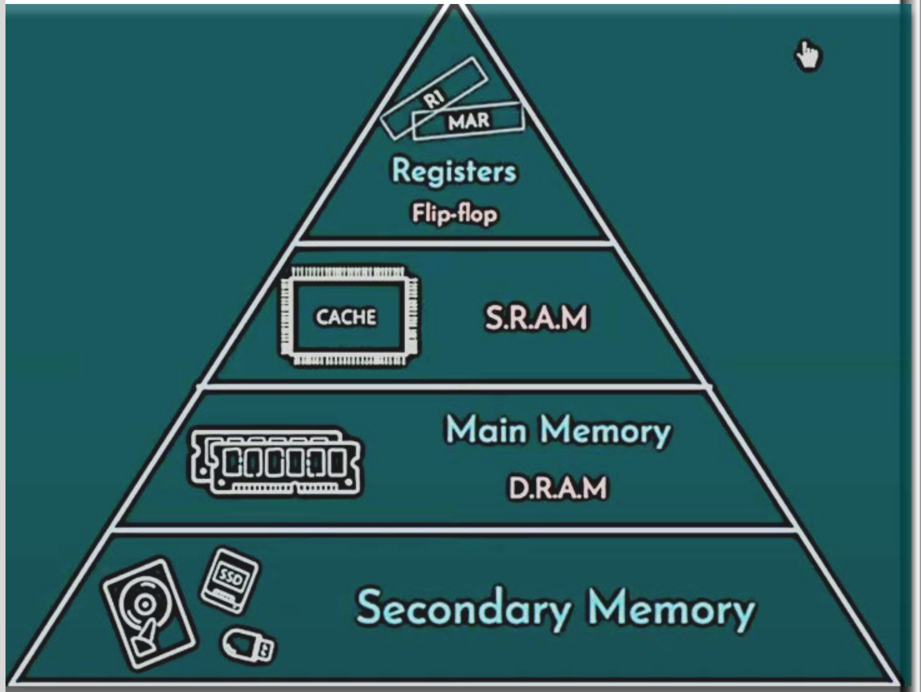
Memory Type	Access Time	Size	Cost per Bit
Registers	**Few ns**	Very small	High
Cache Memory	**Few ns**	Small	High
Main Memory (RAM)	**10-100 ns**	Large	Moderate
Secondary Storage (HDD, SSD)	**ms (HDD), µs (SSD)**	Very Large	Low

Registers

1) Registers are **present inside the processor**.
2) They are basically a **set of flip-flops**.
3) They store **data and addresses** and can **directly take part** in **arithmetic and logic operations**.
4) They are very small in size typically **just a few bytes**.

Main or Primary Memory





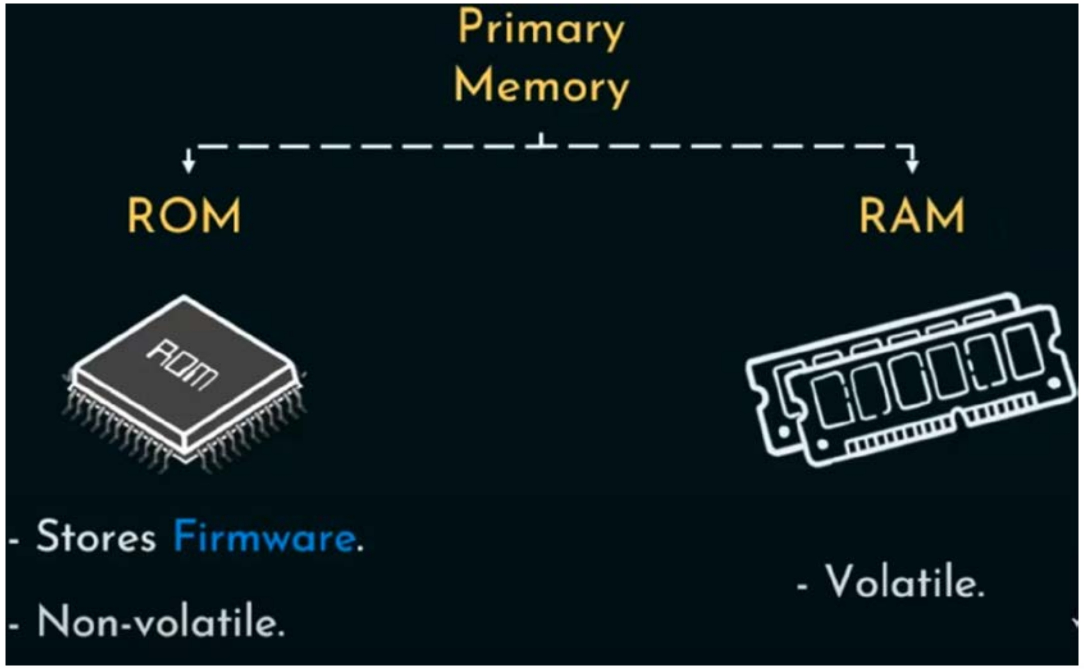
Main or Primary Memory

PRIMARY MEMORY

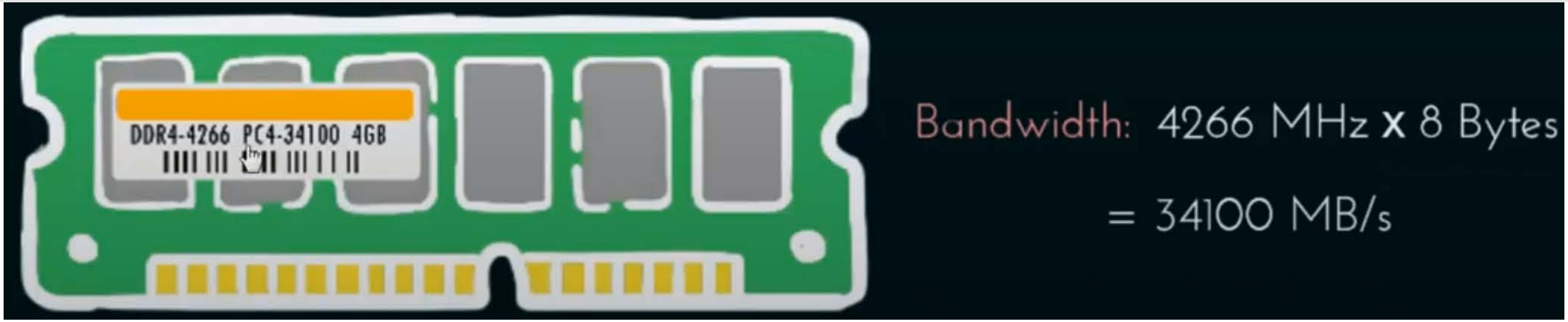
1) It is the original form of memory also called as **Main memory**.
2) It comprises of **RAM and ROM**, both are **Semi-Conductor** memories. (chip memories)
3) **ROM is non-volatile**.
   It is used is storing permanent information like the **BIOS program**.
   It is typically of **2 MB - 4 MB** in size.
4) **RAM is writable** and hence is used for **day-to-day operations**.
   Every file that we access from secondary memory, is **first loaded into RAM**.
   To provide large amount of working space RAM is **typically 4 GB - 8 GB**.

Main or Primary Memory

- Primary memory is a computer system's internal memory.
- It stores and retrieves data, instructions.
- Primary memory is volatile and loses data and instructions when the power turns off.



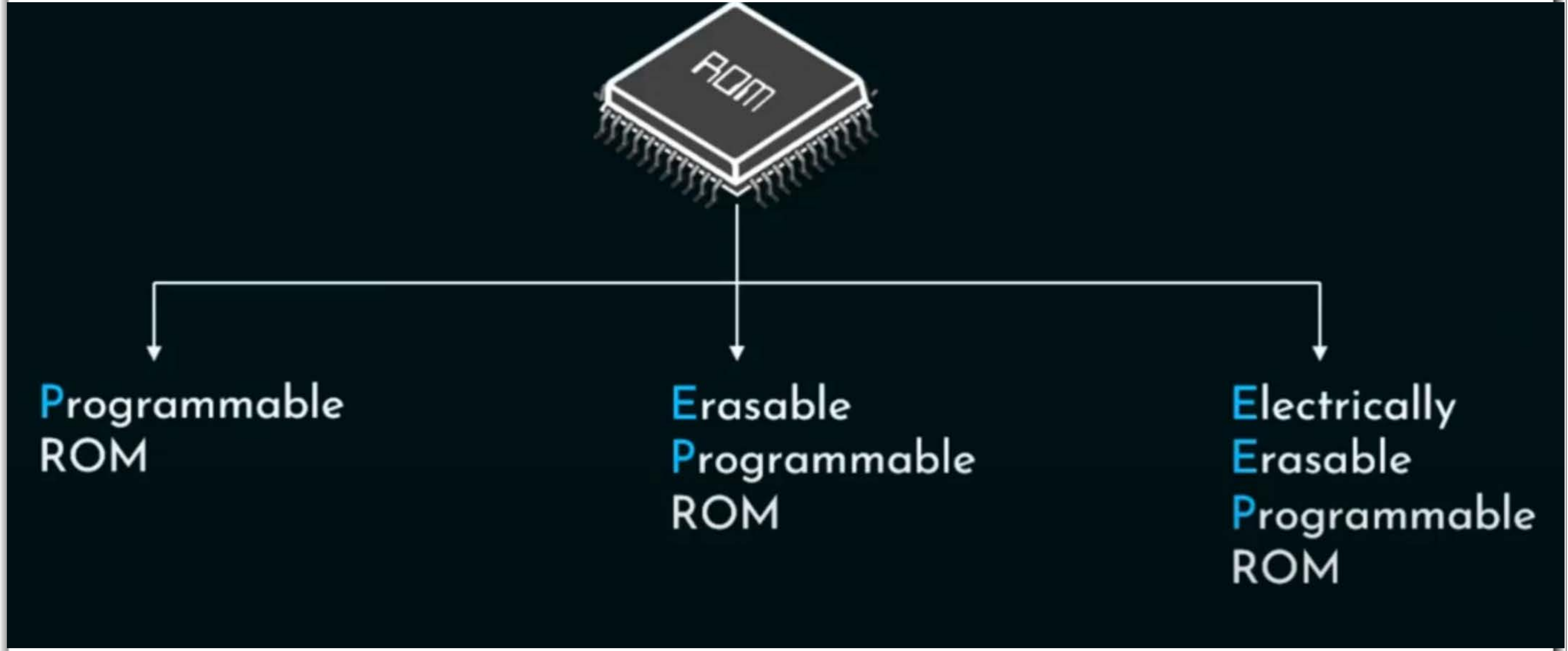
RAM



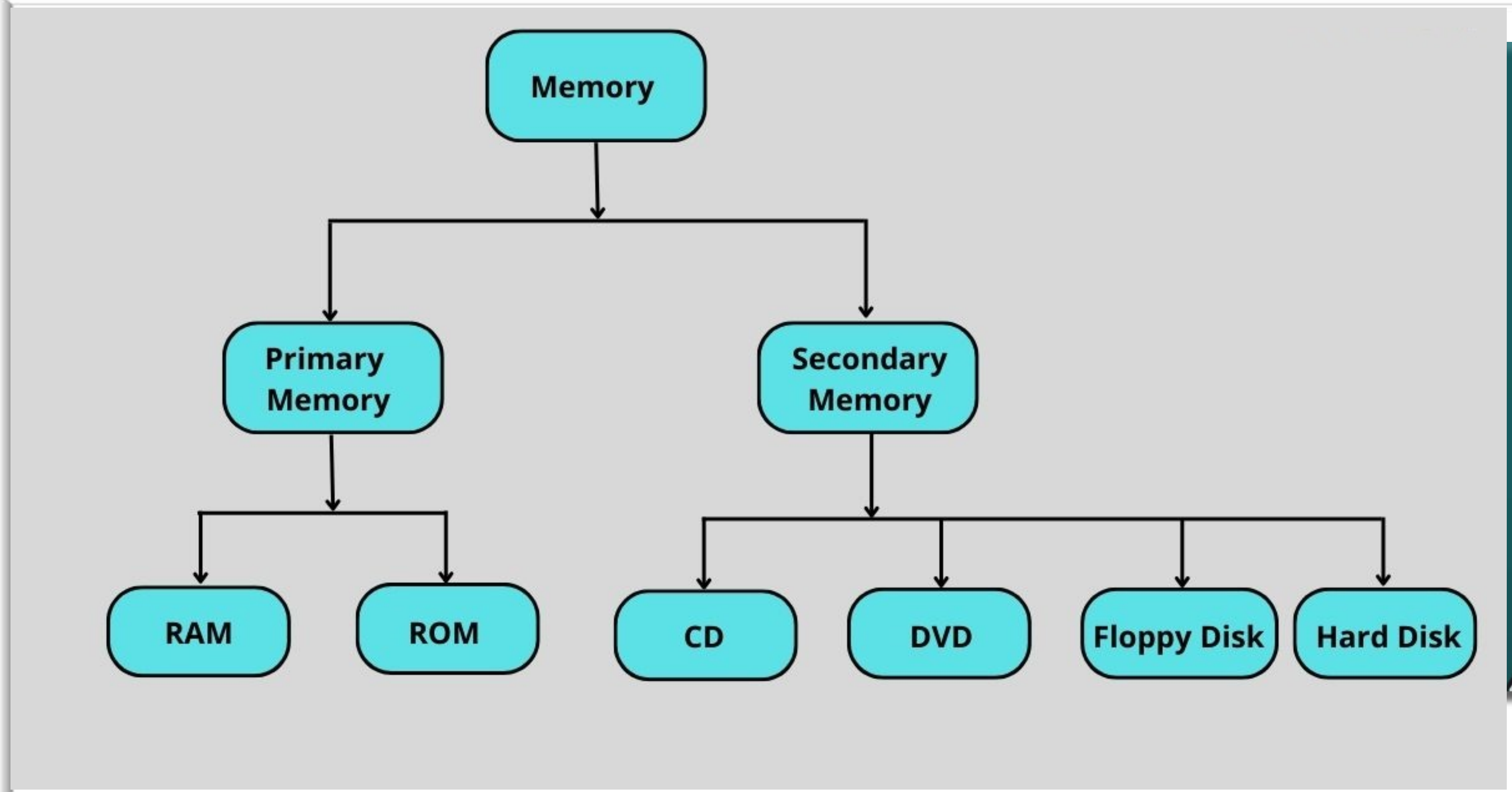
Bandwidth: 4266 MHz x 8 Bytes

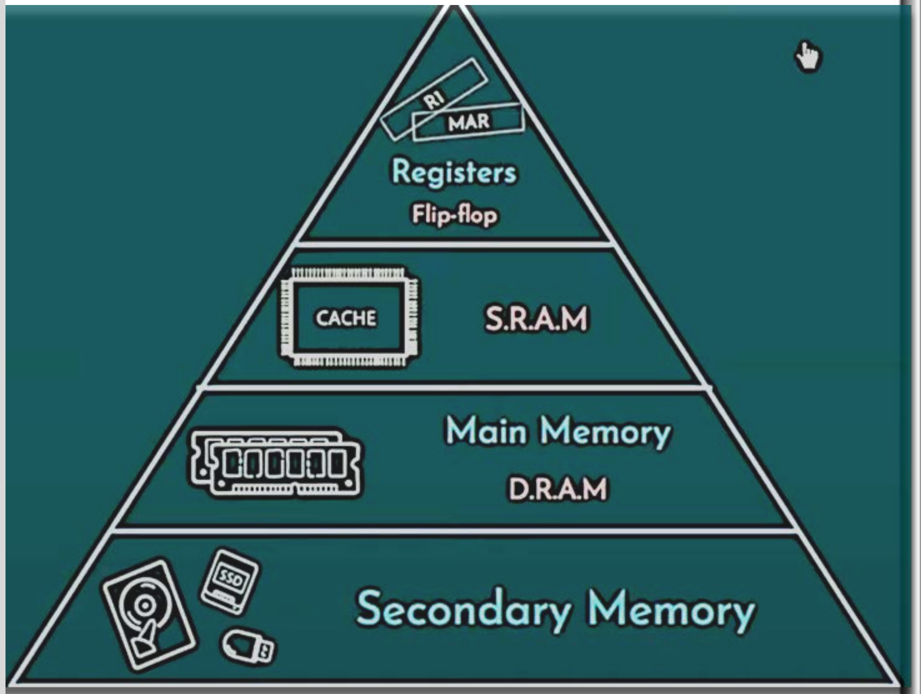
= 34100 MB/s

Classification of ROM


ROM
Programmable ROM
Erasable Programmable ROM
Electrically Erasable Programmable ROM

Secondary Memory





Secondary Memory

Secondary memory is a permanent storage device that the CPU cannot access directly.

•These are magnetic and optical memories.

•It is a type of **non-volatile** memory, i.e., data is permanently stored even when the computer is turned off

•It helps store data on a computer

•Slower than primary memory

•**Examples**: magnetic tapes, optical discs, floppy disks, USB drives, HDD, SSD etc.

Secondary Memory

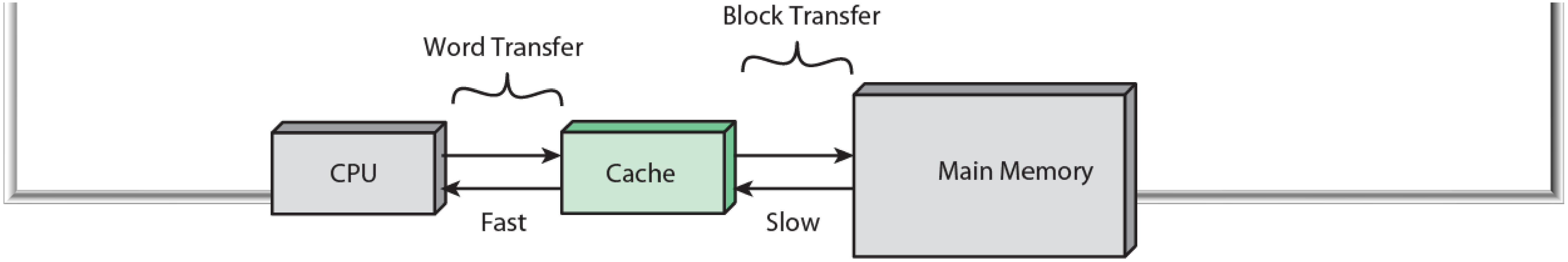
1) The main purpose of Secondary Memory is to **increase the storage capacity, at low cost**.
2) Its biggest component is the **Hard Disk**.
   This is where all the files inside a computer **are stored**.
3) It is **writeable as well as non-volatile**.
4) Typical size of a **HD is 1 TB**.
5) Disk memories are much **slower than chip memories** but are also **much cheaper**.

Portable Secondary Memory

1) These are required to **physically transfer files** between computers.
2) **Floppy Disk**: It is a **magnetic form** of storage. Typical **Size is 1.44 MB**.
3) **CD**: It is an **optical form** of storage. Typical **Size is 700 MB**.
4) **DVD**: It is an **optical form** of storage. Typical **Size is 4.7 MB**.
5) **Pen Drives & Memory Cards**: It is a **semi-conductor form** of storage.
   It is composed of FLASH ROM.
   It's a special type of ROM that's writable as well as non-volatile.
   Typical **Size ranges form 1 GB - 64 GB** depending upon the cost.

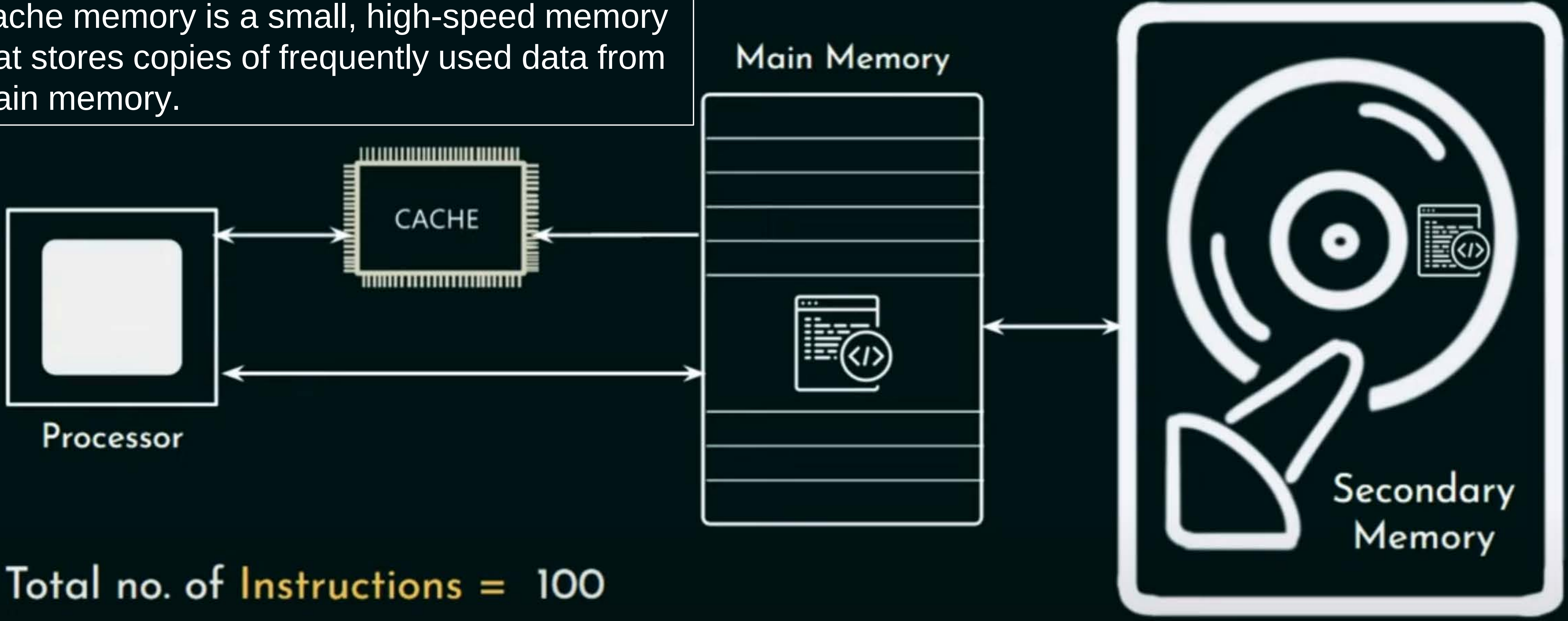
Cache Memories

1) It is the **fastest form of memory** as it uses SRAM (Static RAM).
2) The Main Memory uses DRAM (Dynamic RAM).
3) **SRAM uses flip-flops and hence is much faster than DRAM which uses capacitors.**
4) But SRAM is also **very expensive** as compared to DRAM.
5) Hence **only the current portion of the file** we need to access is copied from Main Memory (DRAM) to Cache memory (SRAM), to be directly accessed by the processor.
6) This gives **maximum performance and yet keeps the cost low**.
7) Typical size of Cache is around **2 MB – 8MB**.
8) If **code and data** are in the **same cache** then it is **unified cache** else its **called split cache**.
9) Depending upon the location of cache, it is of three types: L1, L2 and L3.
10) **L1** cache is **present inside the processor** and is a **split cache** typically **4-8 KB**.
11) L2 is present on the **same die as the processor** and is a **unified cache** typically **1 MB**.
12) L3 is present **outside the processor**. It is also **unified** and is typically of **2-8 MB**.



Cache Memory Basics

Cache memory is a small, high-speed memory that stores copies of frequently used data from main memory.



Total no. of Instructions = 100

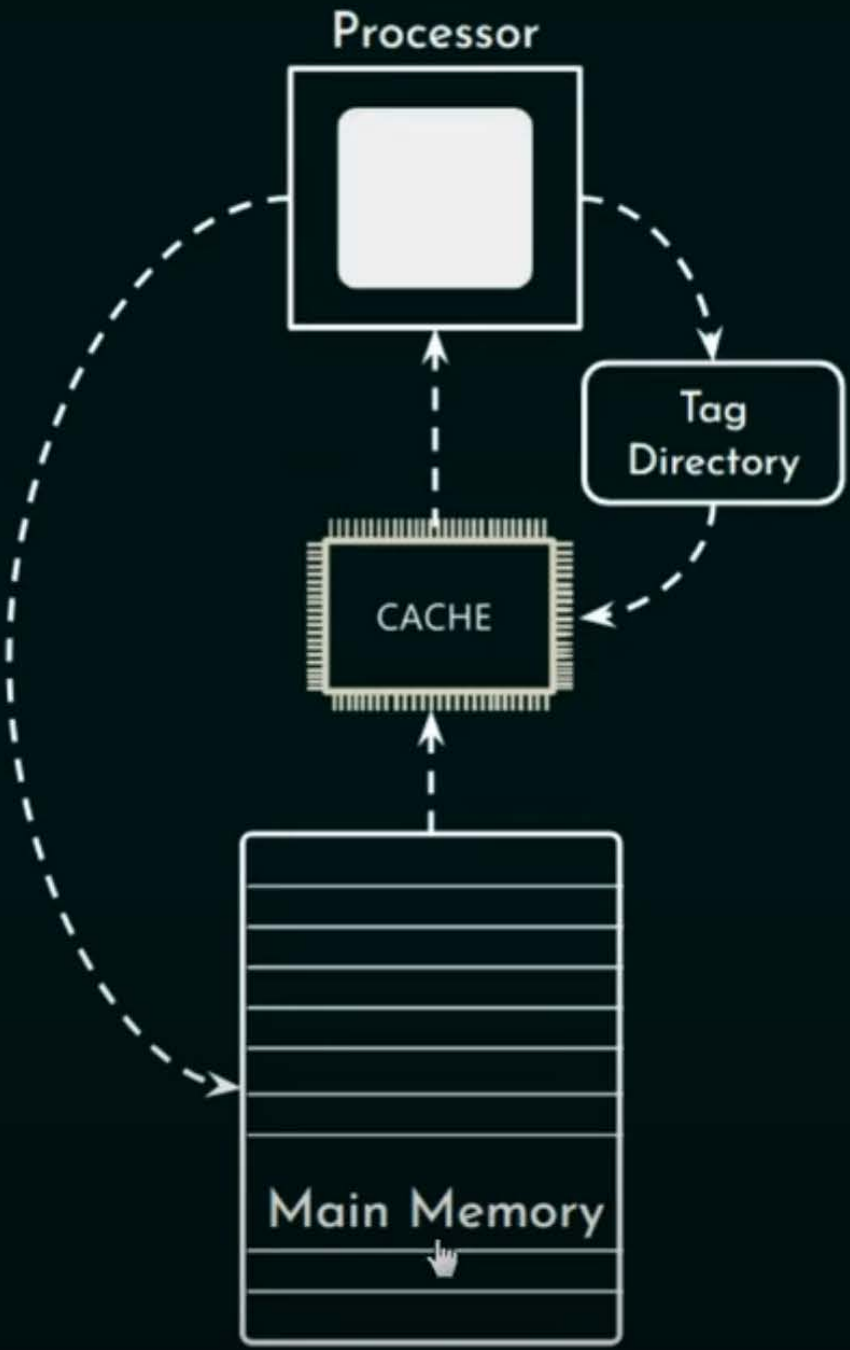
No. of Instructions in MM = 80

MM's Hit Ratio = 80/100 = 0.8 or 80%

Cache related terms:

- Cache Hit "Hit Latency"
- Tag Directory (Data Structure)
- Cache Miss "Miss Latency"

- CPU requests contents of memory location
- Check cache for this data
- If present, get from cache (fast)
- If not present, read required block from main memory to cache
- Then deliver from cache to CPU
- Cache includes tags to identify which block of main memory is in each cache slot



Cache memory mapping

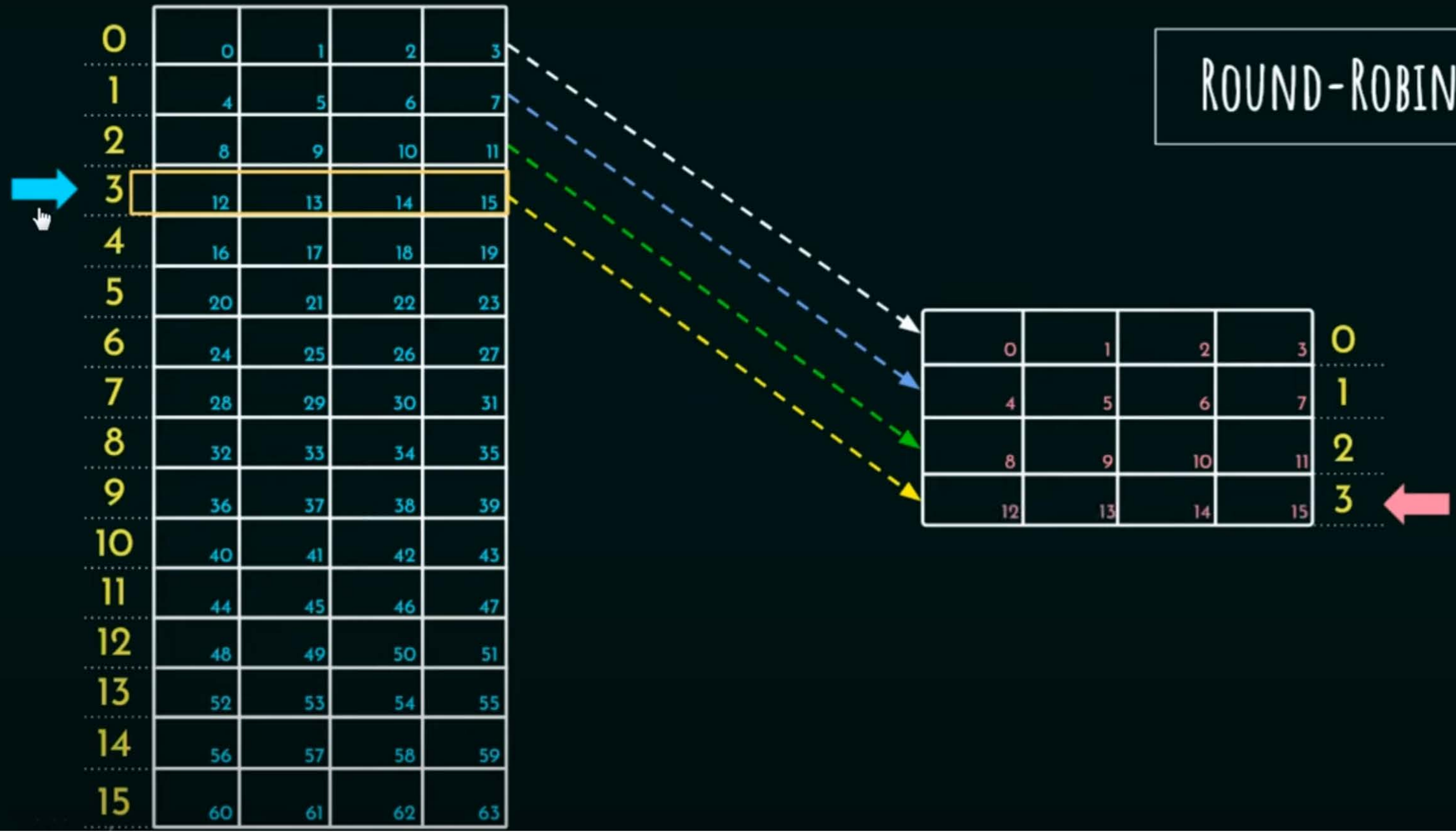
DIRECT MAPPING (ONE WAY SET ASSOCIATIVE MAPPING)

Direct Mapping technique states:
Any block of Main Memory can only be mapped at ONE block of Cache Memory.
Since there is only one way of Mapping, its also called One Way Set Associative Mapping.
We treat the entire Cache as One Set.
The Main Memory is divided into Sets which are then subdivided into Blocks.
A Block of Main Mem. (of any set), can only be mapped into the same Block No. in Cache Mem.
This means, Block 0 of Main Memory (of any set), can only be mapped into Block 0 of Cache Memory.
In other words, Block 0 of Cache Memory can only contain Block 0 of Main Memory but of any Set.

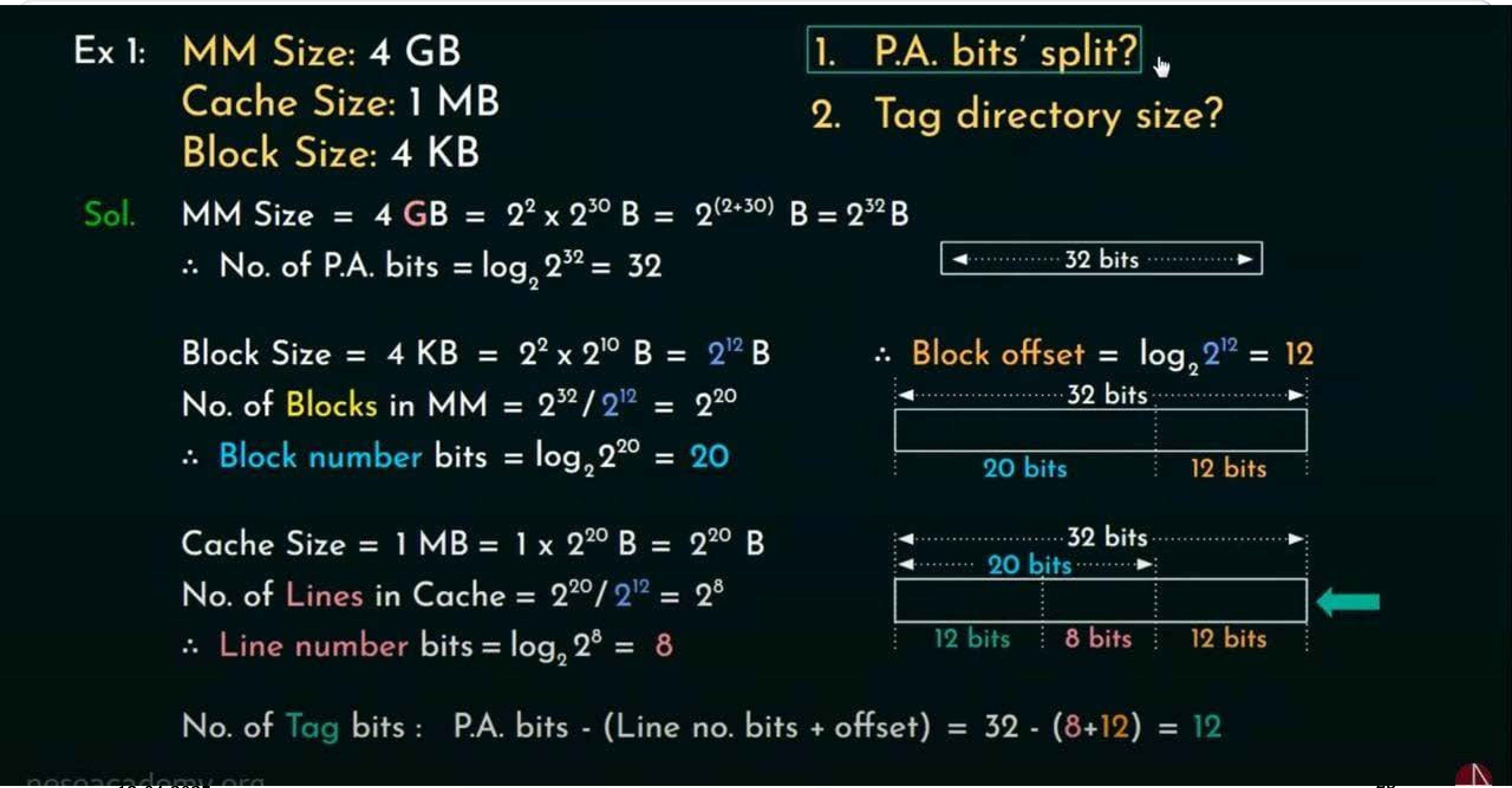
Tag Size:
Since, Block 0 of Cache Memory can only contain Block 0 of Main Memory but of any Set, the Tag has to only indicate the Set No. of Main Memory, from which the block is present.
As Main Memory has 2^{19} sets, the **Tag** size is **19 bits**.

Searches:
If we need Block 0 if Main Memory, we only need to search Block 0 of Cache Memory.
Hence, we need to do only **1 search** in Cache Memory to know if it is a Hit or a Miss.

ROUND-ROBIN



0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15
16 17 18 19
20 21 22 23
24 25 26 27
28 29 30 31
32 33 34 35
36 37 38 39
40 41 42 43
44 45 46 47
48 49 50 51
52 53 54 55
56 57 58 59
60 61 62 63
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15
0
1
2
3



Ex 1: MM Size: 4 GB
Cache Size: 1 MB
Block Size: 4 KB

1. P.A. bits' split?
2. Tag directory size?

Sol. MM Size $= 4\ GB = 2^2 \times 2^{30}\ B = 2^{(2+30)}\ B = 2^{32}\ B$

$\therefore$ No. of P.A. bits $= \log_2 2^{32} = 32$

32 bits

Block Size $= 4\ KB = 2^2 \times 2^{10}\ B = 2^{12}\ B$

No. of Blocks in MM $= 2^{32} / 2^{12} = 2^{20}$

$\therefore$ Block number bits $= \log_2 2^{20} = 20$

$\therefore$ Block offset $= \log_2 2^{12} = 12$

32 bits
20 bits
12 bits

Cache Size $= 1\ MB = 1 \times 2^{20}\ B = 2^{20}\ B$

No. of Lines in Cache $= 2^{20} / 2^{12} = 2^8$

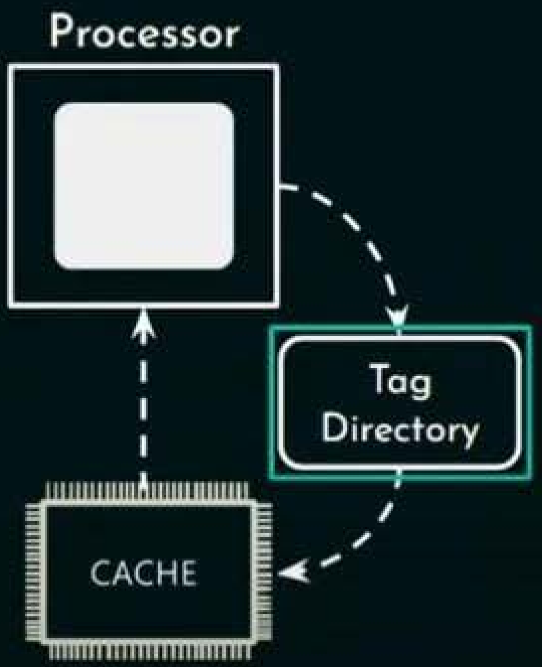
$\therefore$ Line number bits $= \log_2 2^8 = 8$

No. of Tag bits : P.A. bits - (Line no. bits + offset) $= 32 - (8+12) = 12$

Ex 1: MM Size: 4 GB
Cache Size: 1 MB
Block Size: 4 KB

1. P.A. bits' split?
2. Tag directory size?

Sol.



Tag directory:

- Keeps primarily the record of Tag bits, cache line-wise.
- No. of entries = No. of Cache lines.

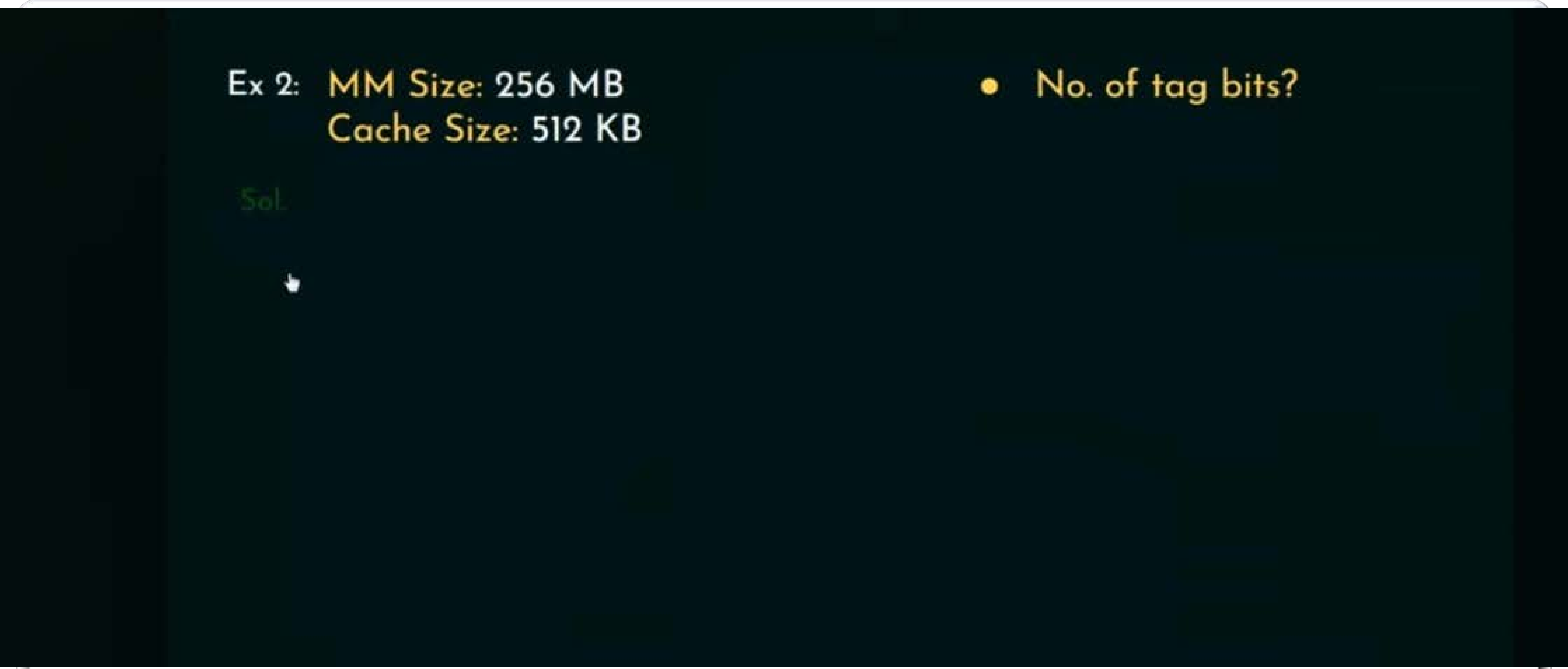
32 bits
20 bits

Tag	Line	B/L offset
12 bits	8 bits	12 bits

No. of Lines in Cache = 2^8

No. of Tag bits = 12

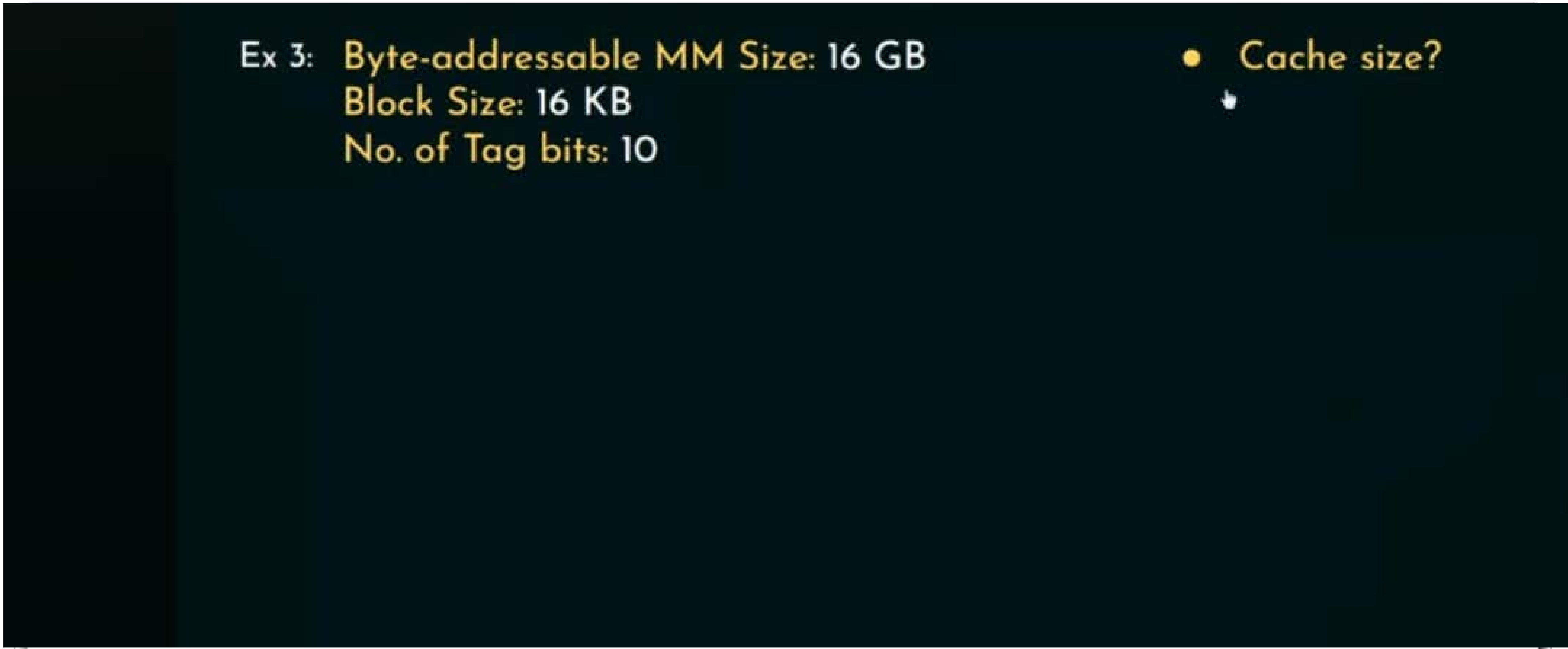
Tag directory size = $2^8 \times 12$ bits



Ex 2: MM Size: 256 MB
Cache Size: 512 KB

- No. of tag bits?

Sol.



Ex 3: Byte-addressable MM Size: 16 GB
Block Size: 16 KB
No. of Tag bits: 10

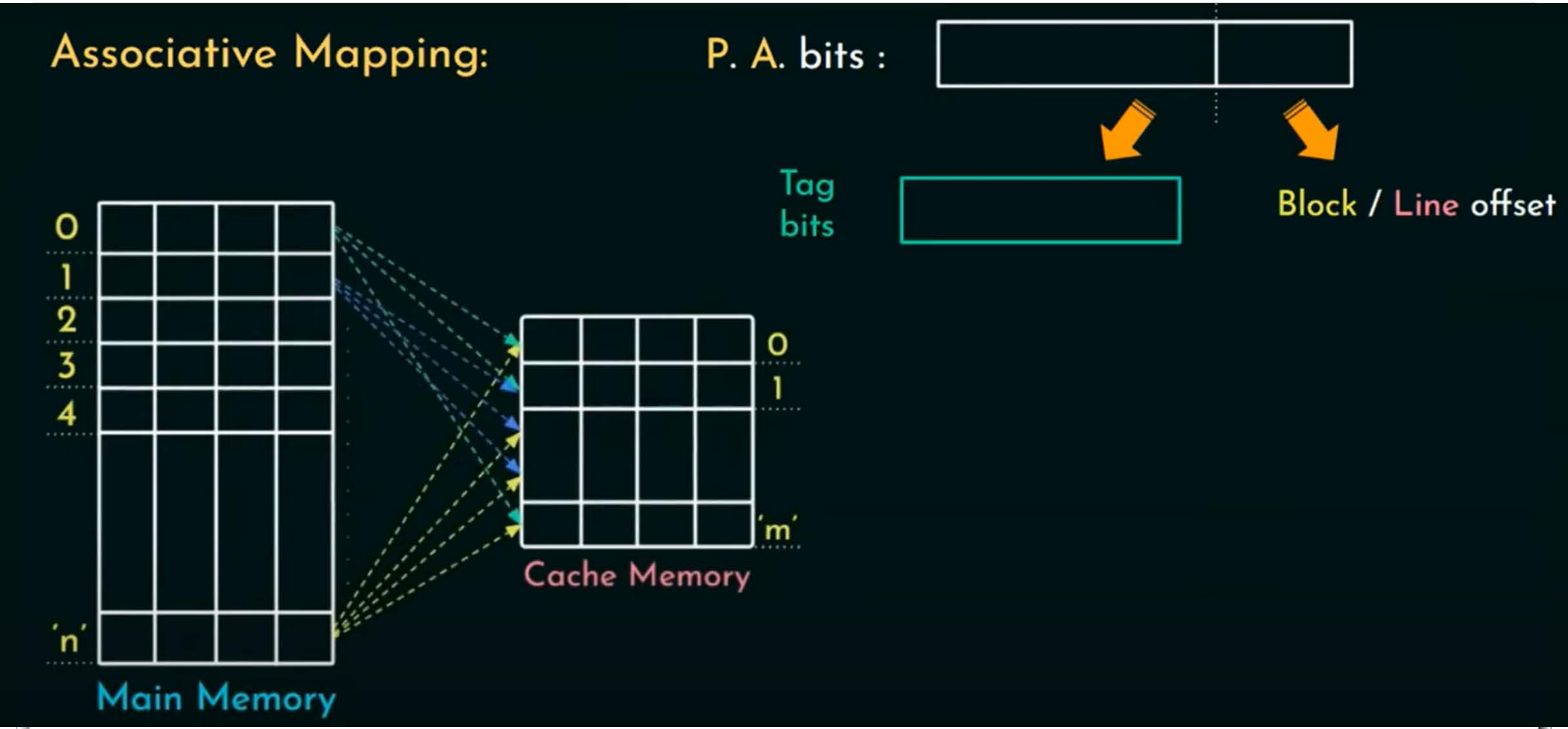
- Cache size?

Cache memory mapping

Associative Mapping (Fully Associative Mapping)

During memory operations, Blocks are loaded from Main Memory to Cache Memory.
Cache Mapping decides which block of Main Memory comes into which block of Cache Memory.
Fully Associative Mapping technique states:
Any block of Main Memory can be mapped at Any available block of Cache Memory.
There are no rules restricting the mapping at all.
This means the Full Cache is available for mapping hence the name Fully Associative.

Associative Mapping:



Ex 1: MM Size: 4 GB
Cache Size: 1 MB
Block Size: 4 KB

1. P.A. bits' split?

2. Tag directory size?

Sol. MM Size = 4 GB = $2^2 \times 2^{30}$ B = $2^{(2+30)}$ B = 2^{32} B

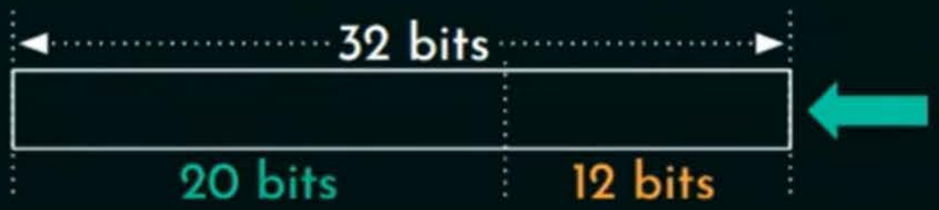
$\therefore$ No. of P.A. bits = $\log_2 2^{32} = 32$

Block Size = 4 KB = $2^2 \times 2^{10}$ B = 2^{12} B

$\therefore$ Block offset = $\log_2 2^{12} = 12$

No. of Blocks in MM = $2^{32} / 2^{12} = 2^{20}$

$\therefore$ No. of Tag bits = $\log_2 2^{20} = 20$



Cache Size = 1 MB = 1×2^{20} B = 2^{20} B

No. of Lines in Cache = $2^{20} / 2^{12} = 2^8$

$\therefore$ Tag directory size = $2^8 \times 20$ bits
= 5120 bits

Ex 2: MM Size: 16 GB
Block Size: 16 KB

1. P.A. bits' split?
2. Tag directory size?

Sol. MM Size = 16 GB = $2^4 \times 2^{30}$ B = 2^{34} B

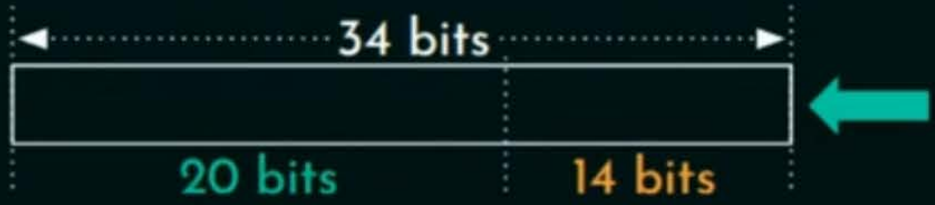
∴ No. of P.A. bits = 34

Block Size = 16 KB = $2^4 \times 2^{10}$ B = 2^{14} B

∴ Block offset = 14

No. of Blocks in MM = $2^{34} / 2^{14} = 2^{20}$

∴ No. of Tag bits = 20



Cache Size = ?

No. of Lines in Cache = ?

∴ Tag directory size = --

Q: A certain processor uses a fully associative cache of size 16 kB, The cache block size is 16 bytes. Assume that the main memory is byte addressable and uses a 32-bit address. How many bits are required for the Tag and the Index fields respectively in the addresses generated by the processor?

(A) 24 bits and 0 bits (B) 28 bits and 4 bits

(C) 24 bits and 4 bits [✓ (D) 28 bits and 0 bits]

GATE CS 2019

Sol. Cache Size $= 16\text{ KB} = 2^{(4+10)}\text{ B} = 2^{14}\text{ B}$

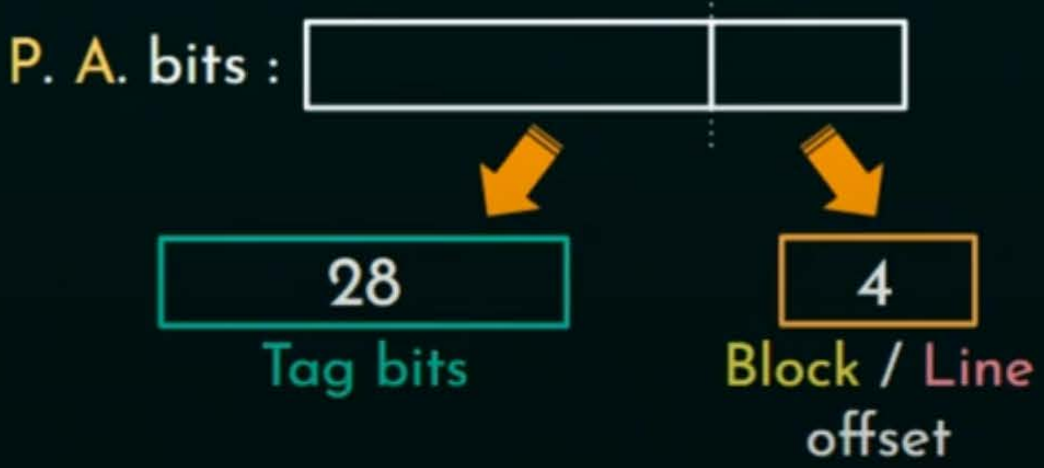
Block Size $= 16\text{ B} = 2^{4}\text{ B}$

$\therefore$ Block offset $= 4$

P. A. bits : 32

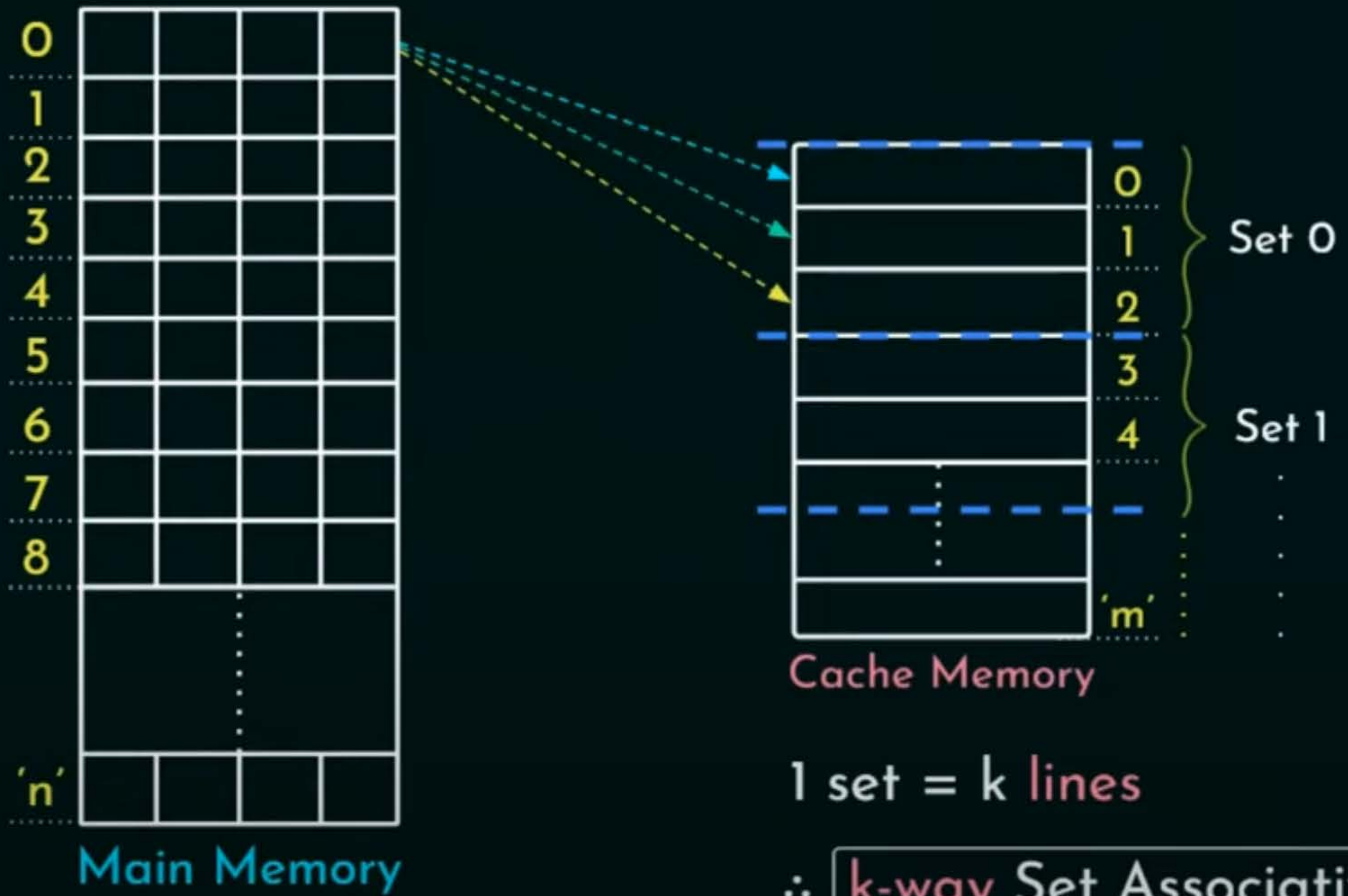
Tag bits $= 32 - 4 = 28$

$\therefore$ Index bits $= \log_2 2^{(14-4)} = 10$



12-04-2025

Set Associative Mapping:



1 set = k lines

∴ $\boxed{\text{k-way Set Associative}}$

Cache memory mapping

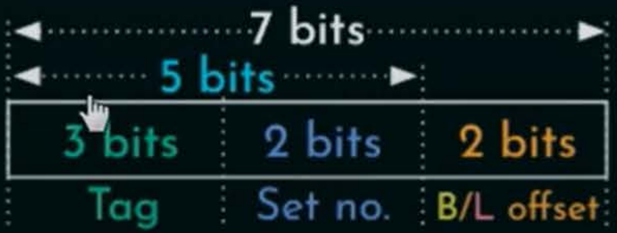
Set Associative Mapping (Two Way Set Associative Mapping)

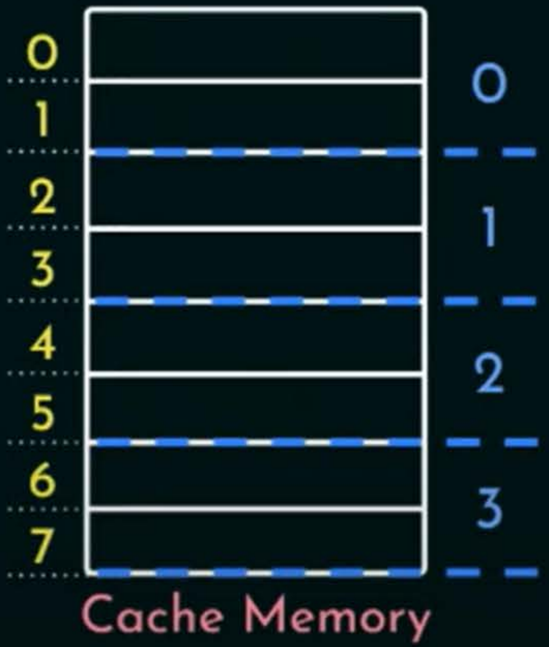
Two way Set Associative Mapping technique states:
A block of Main Memory can only be mapped into the same corresponding Block No. of Cache Memory, in any of the two sets.
Since there are two ways of Mapping, its called Two Way Set Associative Mapping.
We treat the entire Cache as **Two Sets**. The Main Memory is divided into Sets, subdivided into Blocks.
A Block of Main Mem. (of any set), can only be mapped into the same Block No. in Cache Memory again of any set. This means, Block 0 of Main Memory (of any set), can only be mapped into Block 0 of Cache Memory, into one of its two sets.
In other words, Block 0 of Cache Memory can only contain Block 0 of Main Memory but of any Set.

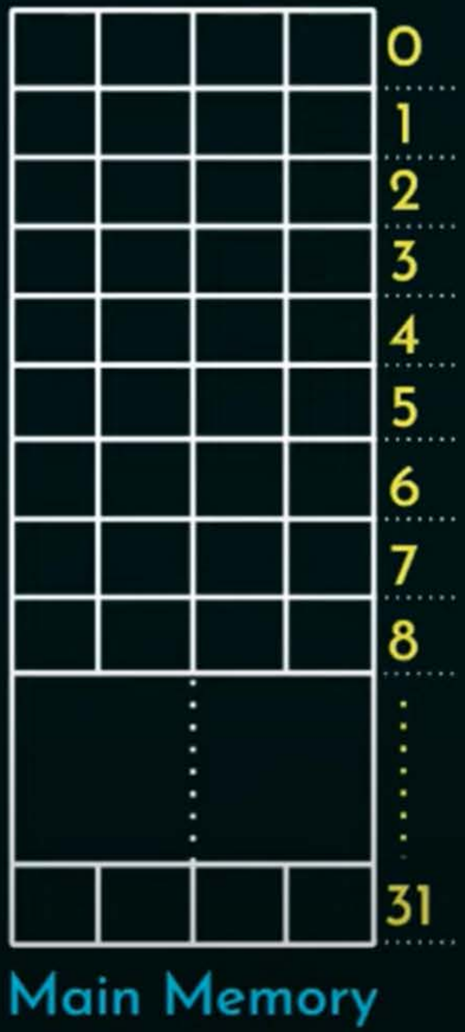
Ex : Byte addressable MM Size: 128 B
Cache Size: 32 B
Block Size: 4 B
2-way Set Associative $\boxed{1 \text{ set} = 2 \text{ lines}}$

P.A.S. = 128 B = 2^7 B ∴ P.A. bits = 7
Block Size = 2^2 B ∴ offset = 2
∴ No. of MM Blocks = $2^7 / 2^2$
$= 2^5$

Cache Size = 2^5 B
∴ No. of lines = $2^5 / 2^2 = 2^3$
∴ No. of sets = $2^3 / 2^1 = 2^2$

7 - (2+2) =






Cache memory mapping

A block-set associative cache memory consists of 128 blocks divided into four block sets. The main memory consists of 16,384 blocks and each block contains 256 eight-bit words.

a) How many bits are required for addressing the main memory?

b) How many bits are needed to represent the TAG, SET and offset fields?

c) Skitch the block diagram with specifications for the same. **(12M)**

Cache Memory Mapping Techniques || Summary

Since cache size is smaller than main memory, it needs a mapping technique to determine where each memory block should be stored in cache. The three common methods are:

1.Direct Mapping (One-to-One Mapping)

1. Each block from main memory is mapped to a specific cache line.
2. Simple but suffers from conflicts.

2.Fully Associative Mapping (Any Block Can Go Anywhere)

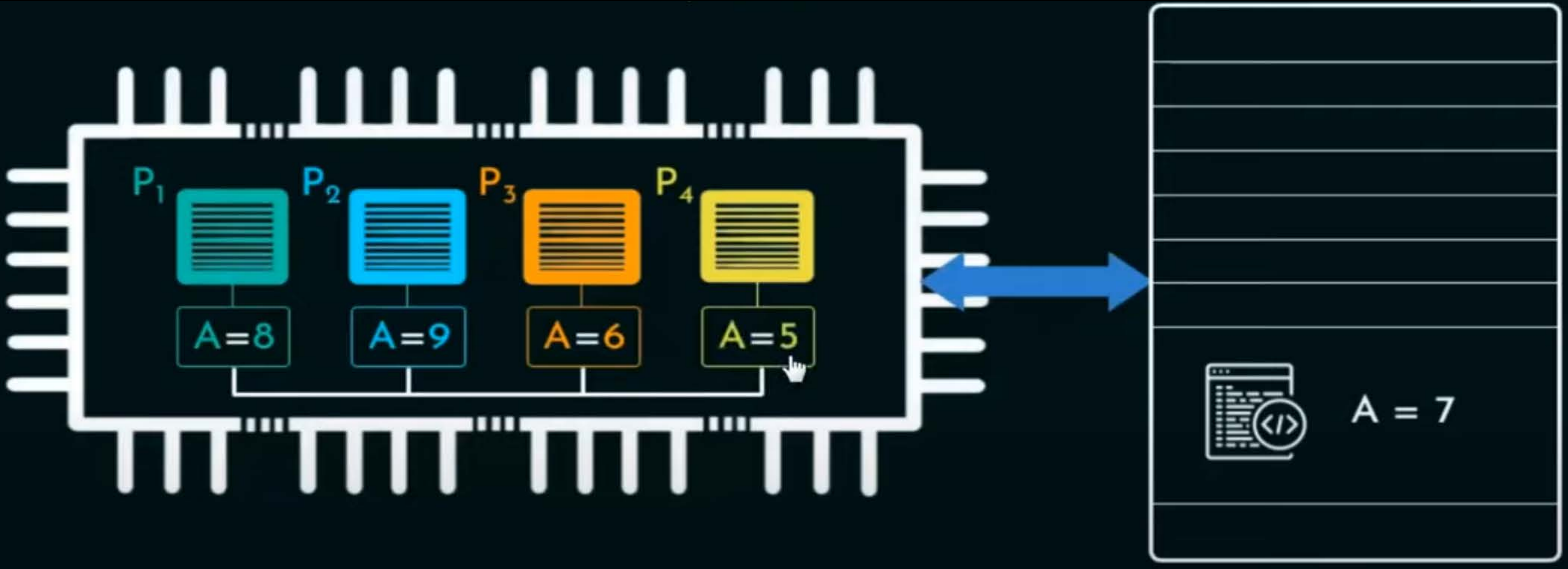
1. A block can be placed in any cache location.
2. More flexible but requires **complex hardware**.

3.Set-Associative Mapping (Compromise between the above two)

1. Blocks are divided into **sets**, and each set contains multiple blocks.
2. A memory block maps to **one specific set**, but can be stored in **any block within that set**.
3. Example: **4-way set associative** means each set contains **4 blocks**.

Cache Coherence

Cache coherence is the mechanism to ensure data consistency across caches by maintaining a consistent view of memory, using techniques like invalidation or updating cache lines.



Cache Coherence

- The cache coherence problem arises in multiprocessor systems where multiple caches store copies of the same data, potentially leading to inconsistencies if one cache's copy is updated without informing others.
- In a multiprocessor system, each processor (or core) typically has its own cache memory to speed up access to frequently used data.
- However, when multiple processors access and modify the same data, inconsistencies can arise if one processor updates its local cache copy without informing the others.

Cache Coherence

Cache coherence is the mechanism to ensure data consistency across caches by maintaining a consistent view of memory, using techniques like invalidation or updating cache lines.