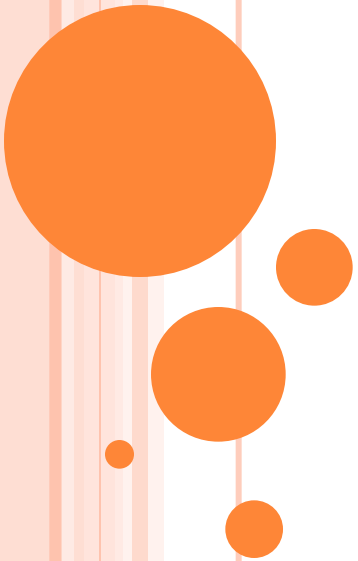
COMPUTER ORGANIZATION AND ARCHITECTURE

UNIT 4: MEMORY UNIT



Memory

- Memory is the computer's electronic scratchpad or local store in computer terminology.
- Used for temporary storage of calculations, data, and other work in progress.
- Two types: Primary and Secondary
- Primary memory or the main memory is part of the main computer system. The primary memory itself is of two types.
- The first is called random access memory (RAM) and the other is read only memory (ROM).

- Some reasons why memory play vital role in computers overall performance.

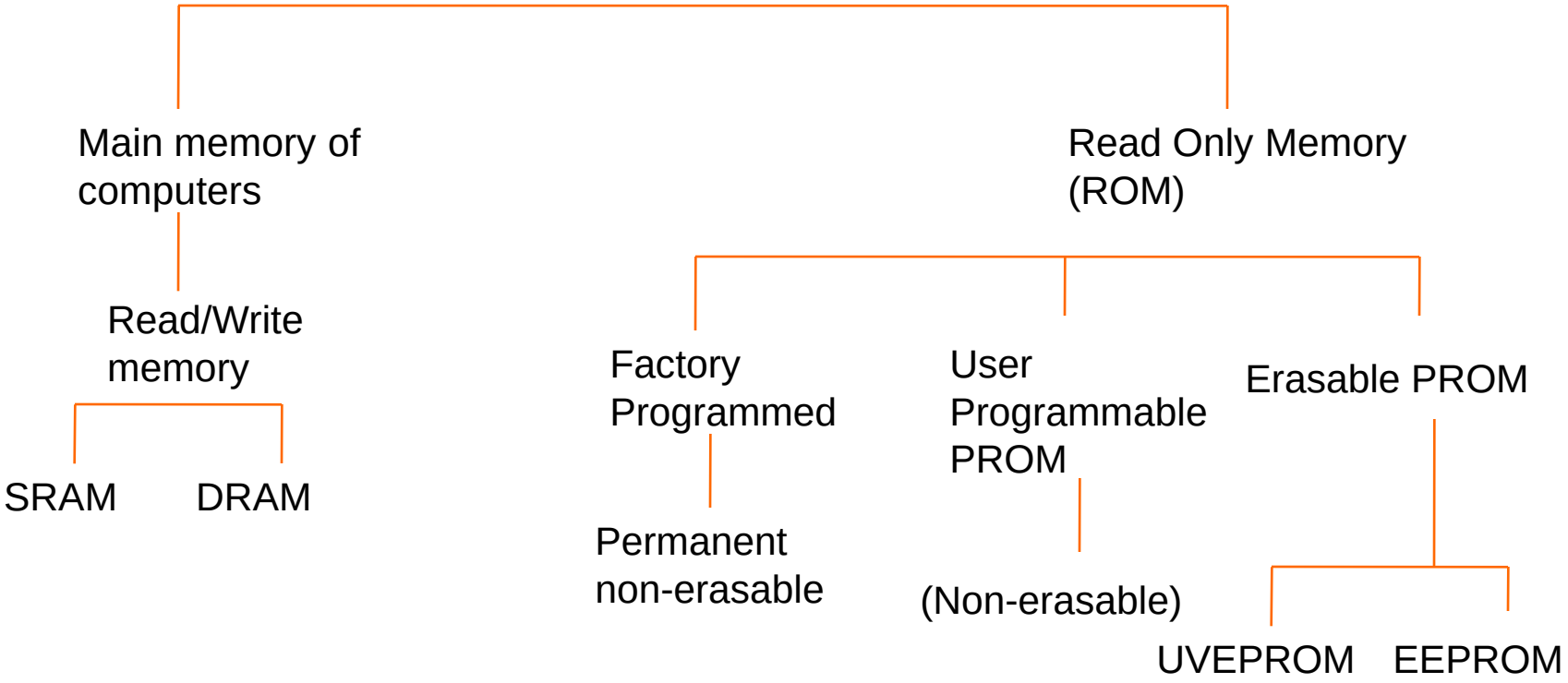
- Storage of Data: Allows the access of data quickly
- Multitasking: multiple tasks simultaneously
- Running Application: Applications require some memory to run
- Operating System: Also requires to run smoothly.

Importance of Memory:

- Speed
- Capacity
- Stability
- Cost
- Scalability

Sample Configuration: 1TB, 16 GB, 768 KB, 4 MB, 16 MB

Varieties of Semiconductor Random Access Memories



Random Access Memory (RAM)

- The processor directly stores and retrieves information from it.
- Memory is organized into locations. Each memory location is identified by a unique address. The access time is same for all location.
- It is volatile: when turned off, everything in RAM disappears.
- Two types:

Types of RAM

- Dynamic Random Access Memory (DRAM):

  This type RAM retain the content of any location only for a few milliseconds. Within that period, each location must be written again with the same contents. This is known as refreshing.

- Static Random Access Memory (SRAM):

  This type of RAM preserves the contents of all the locations as long as the power supply is present. SRAM is generally included in a computer system by the name of cache.

Semiconductor RAM

- It allows data to be read and written in almost the same amount of time irrespective of the physical location of data inside the memory. It uses semiconductor technology to store the data.

Feature	DRAM	SRAM
Storage	Uses Capacitors	Uses bistable latches (Flipflops)
Speed	Slower	Faster
Refresh	Required	Not Required
Power	Lower power consumption	Higher power consumption
Cost	Cheaper	More expensive
Usage	Main System M/Y	Cache M/Y

Read Only Memory (ROM)

- Data stored in ROM cannot be modified, or can be modified only slowly or with difficulty, so it is mainly used to distribute.
- The instructions in ROM are built into the electronic circuits of the chip which is called firmware.
- Random access in nature and non-volatile.

Types of ROM

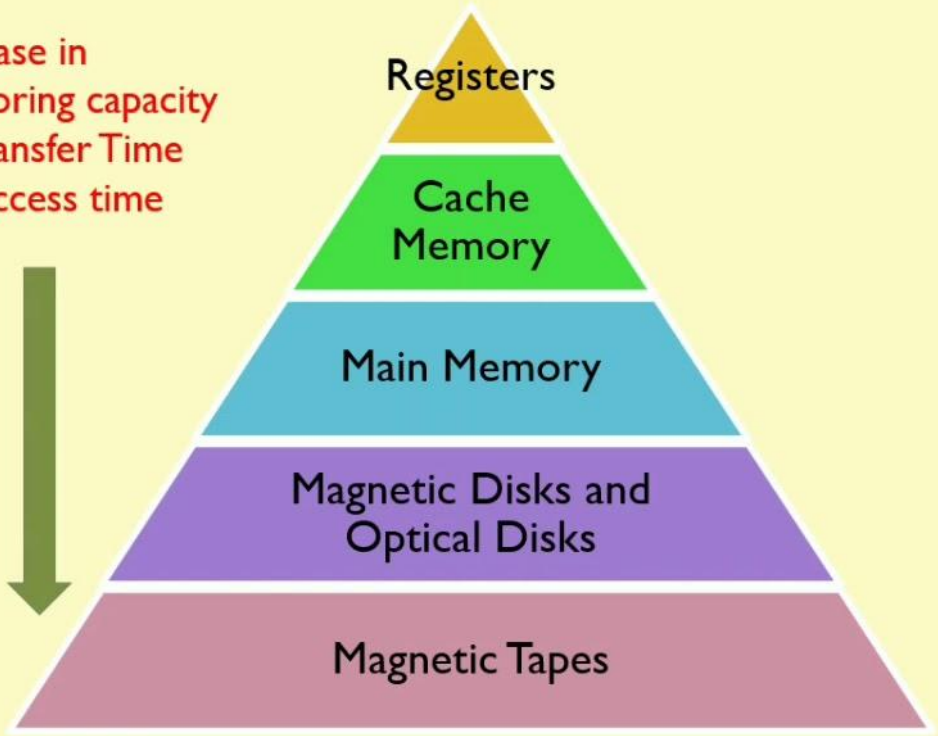
- Programmable read-only memory (PROM), or one-time programmable ROM can be written to or programmed via a special device called a PROM programmer.
- Erasable programmable read-only memory (EPROM) can be erased by exposure to strong ultraviolet light then rewritten with a process that again needs higher than usual voltage applied.
- Electrically erasable programmable read-only memory (EEPROM) is based on a similar semiconductor structure to EPROM, but allows its entire contents (or selected banks) to be electrically erased, then rewritten electrically, so that they need not be removed from the computer

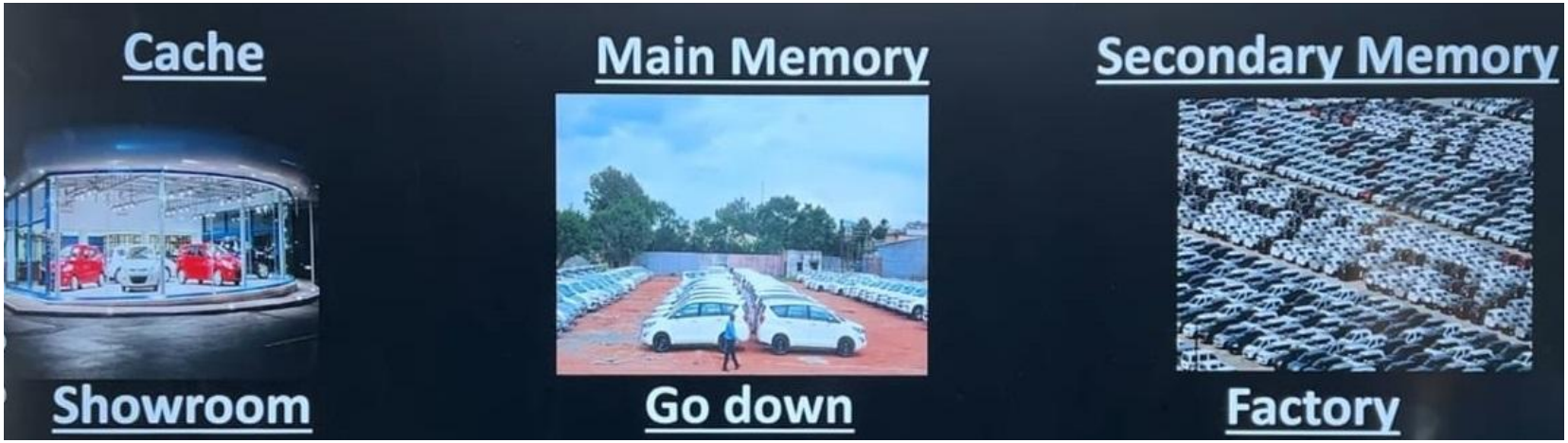
Flash Memory

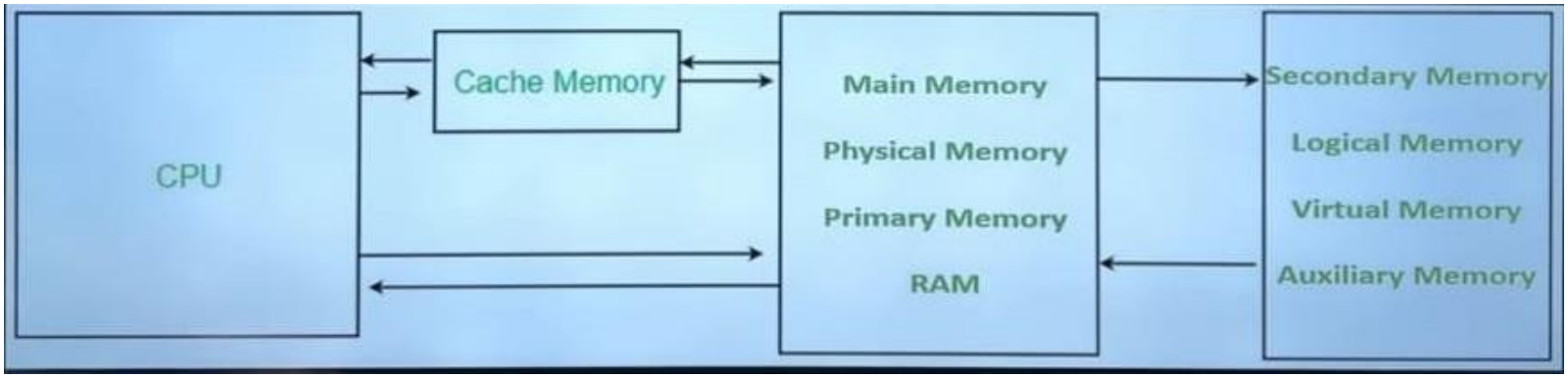
- Modern type of EEPROM invented in 1984.
- Faster than regular EEPROM
- Use one transistor per memory cell and come in different capacities
- The read time is much smaller (tens of nanoseconds) compared write time (tens of microseconds).
- Commonly used for USB drives, SSDs.

MEMORY HIERARCHY

Increase in
- Storing capacity
- Transfer Time
- Access time

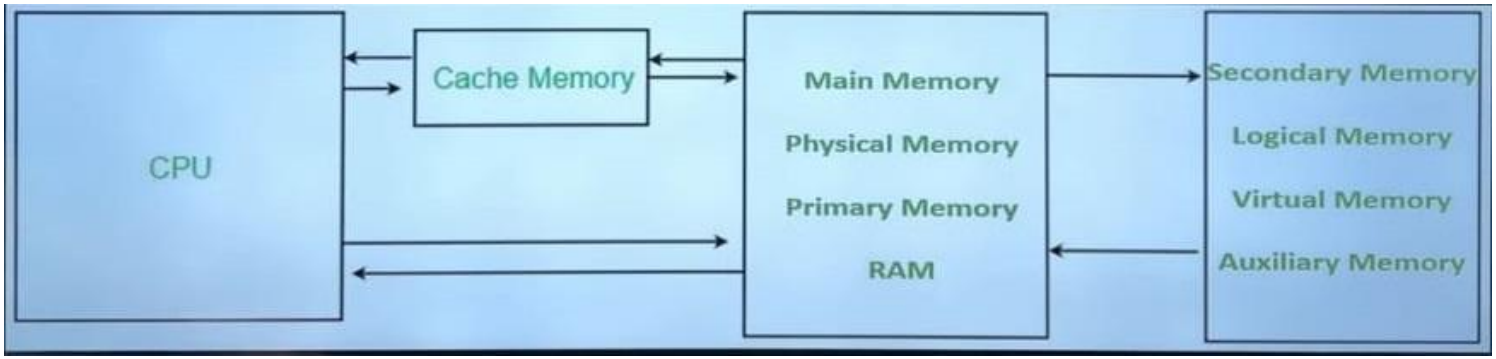



Cache
Showroom
Main Memory
Go down
Secondary Memory
Factory


CPU
Cache Memory
Main Memory
Physical Memory
Primary Memory
RAM
Secondary Memory
Logical Memory
Virtual Memory
Auxiliary Memory

How CPU uses M/Y hierarchy

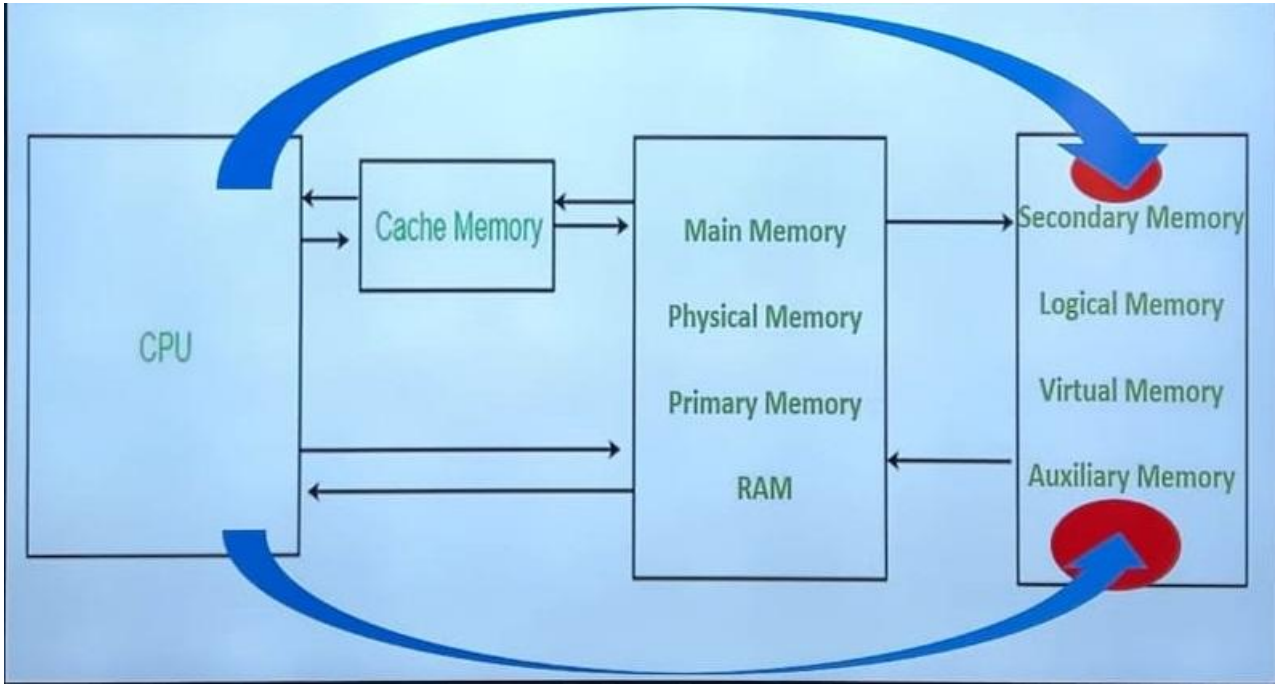
- Cache memory boosts processing speed. It bridges the speed gap between the processor and main memory.
- Main m/y holds data for immediate access by the CPU.
- For efficient processing, data sought by the CPU should ideally be in Cache.



Sample Configuration: 1TB, 16 GB, 768 KB, 4 MB, 16 MB

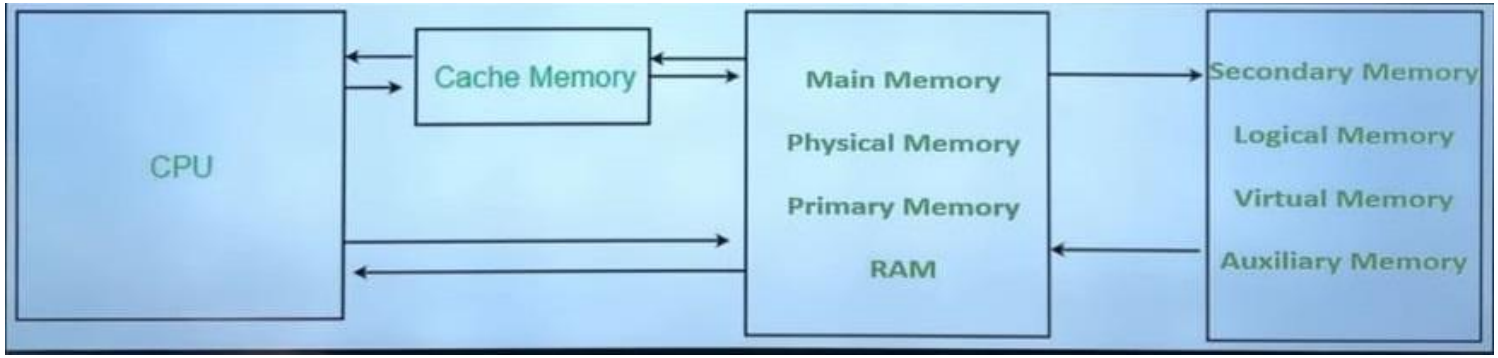
Locality of Reference

- It refers to the tendency of a program to access a small portion of its memory at a given time. While the majority of the m/y is unused. There are two types of locality of reference: temporal and spatial locality.

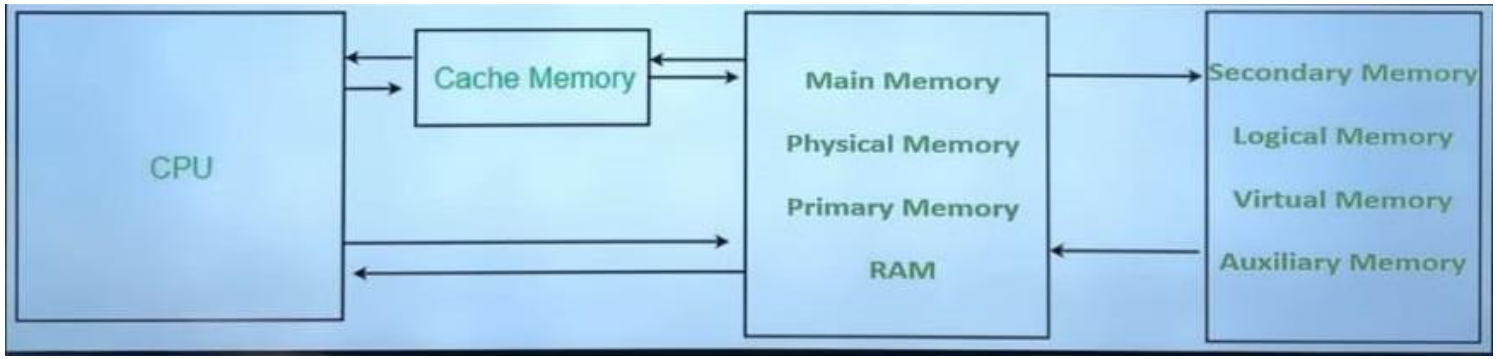


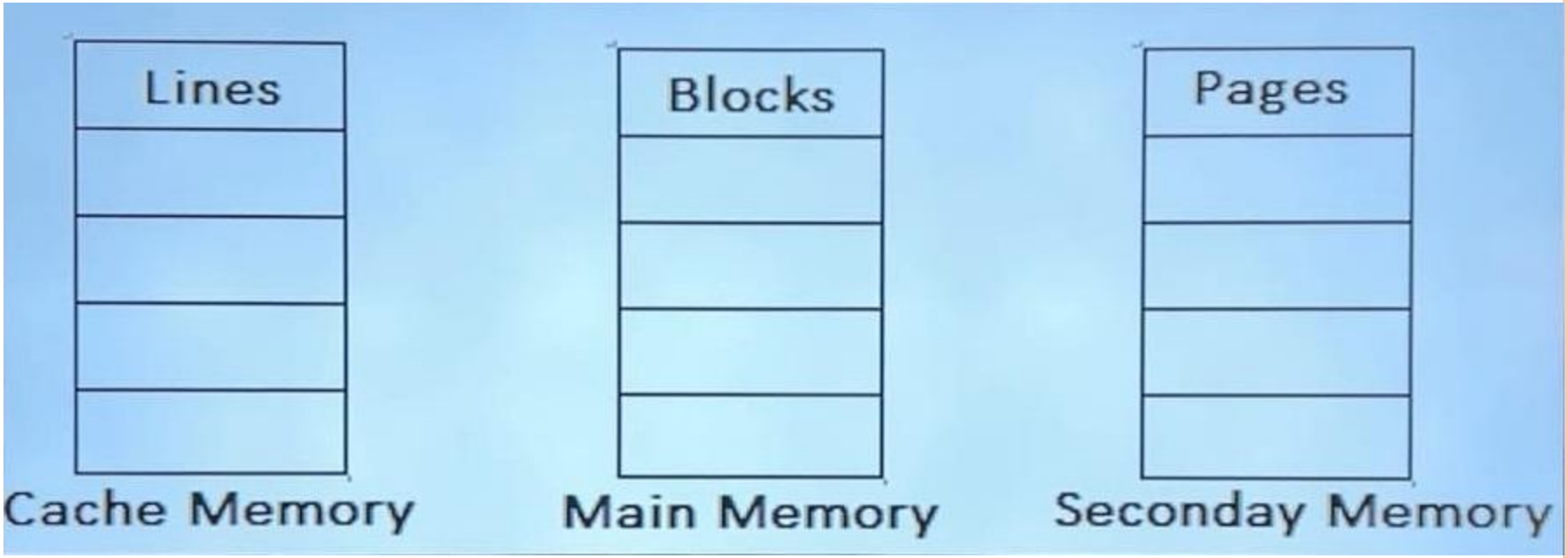
Spatial and Temporal Locality

- Spatial locality Refers to the tendency of a program to access m/y locations that are close to each other. This means that if a program accesses a m/y location, it is likely to access other nearby m/y locations soon after.
- Temporal refers to the tendency of a program to access recently used m/y locations repeatedly. This means that if a program accesses a m/y location, it is likely to access that same location again in the near future.



- Cache Hit:
- Cache Miss:
- Hit Ratio:
- Hit Latency:
- Cache Miss:
- Miss Latency:




Lines
Blocks
Pages
Cache Memory
Main Memory
Seconday Memory

10^3	1 Kilo
10^6	1 Mega
10^9	1 Giga
10^12	1 Tera
10^15	1 Peta

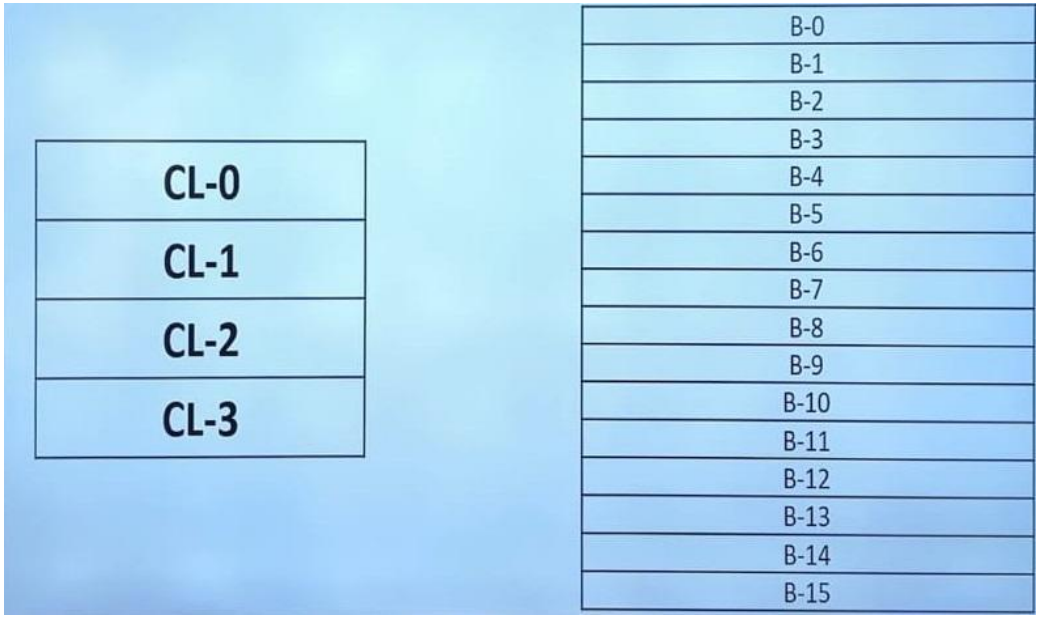
2^10	1 Kilo
2^20	1 Mega
2^30	1 Giga
2^40	1 Tera
2^50	1 Peta

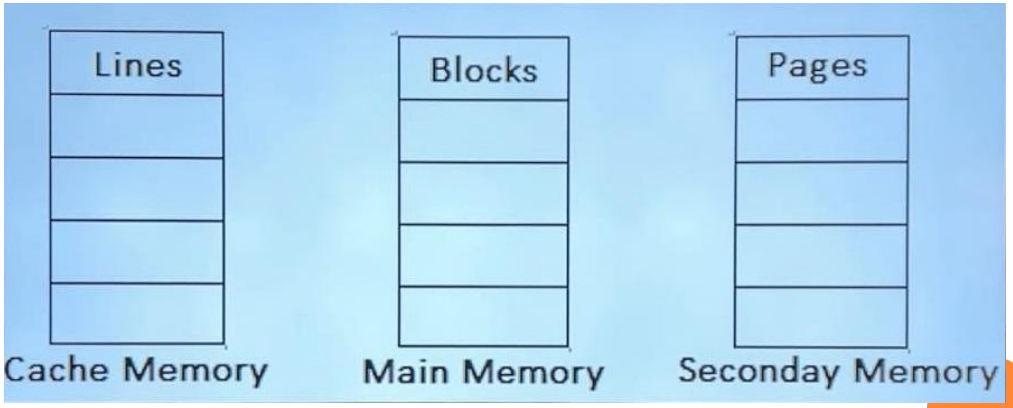
Address Length in bits: n

Number of Locations: 2^n

Locations: Upper bound($Log_2 n$)

Number of bits: n


CL-0
CL-1
CL-2
CL-3
B-0
B-1
B-2
B-3
B-4
B-5
B-6
B-7
B-8
B-9
B-10
B-11
B-12
B-13
B-14
B-15


Lines
Blocks
Pages
Cache Memory
Main Memory
Seconday Memory

Cache Mapping Algorithms

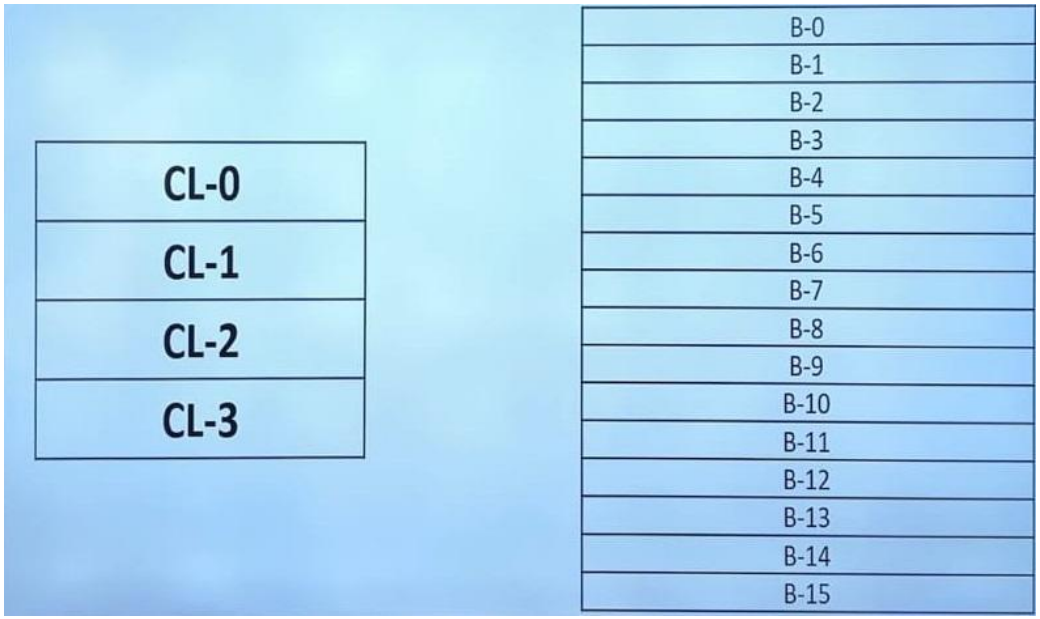
- Cache mapping refers to the process of determining where data should be stored in the cache memory.
- The cache mapping algorithms determines which cache lines are assigned to which main memory blocks

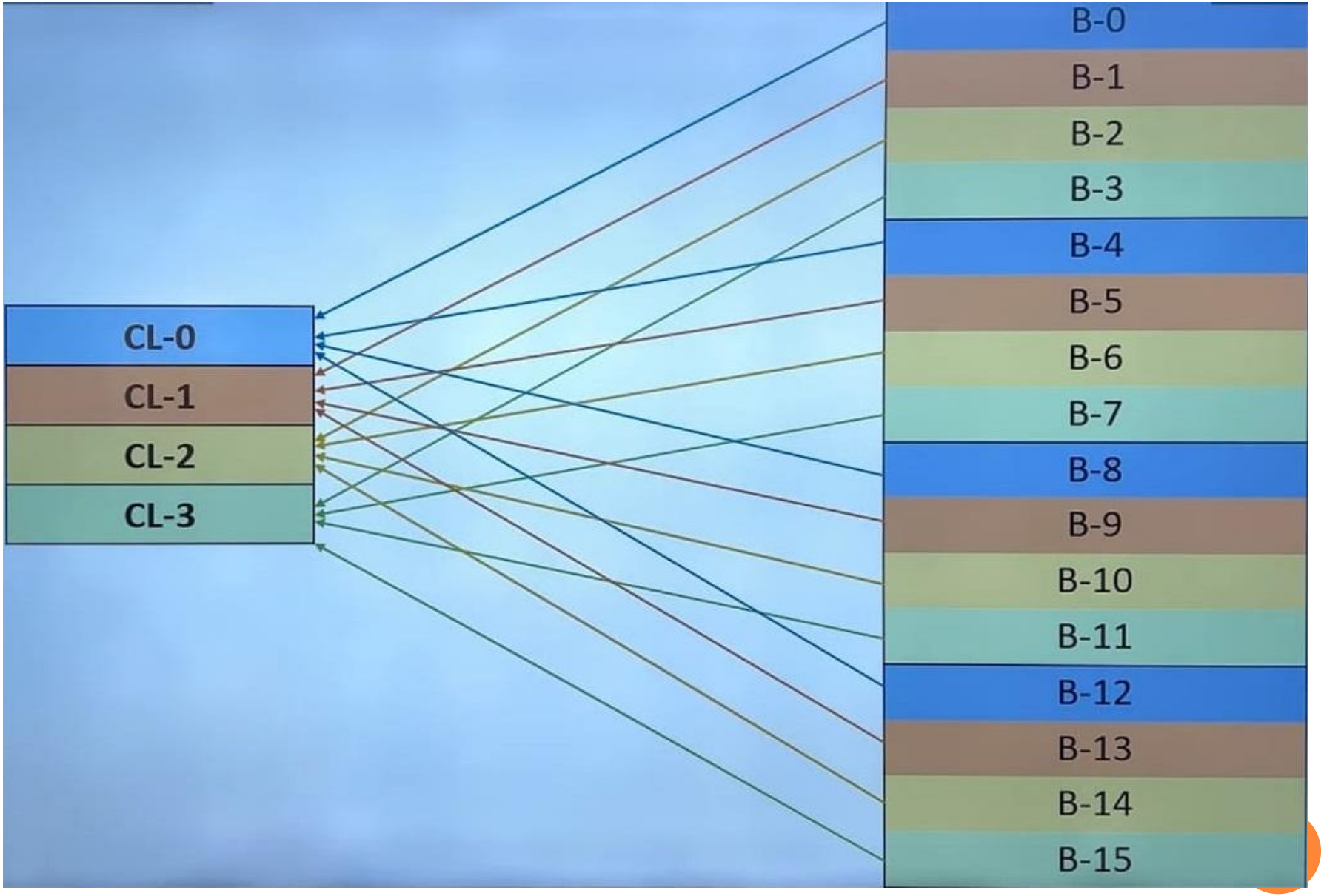
Algorithms:

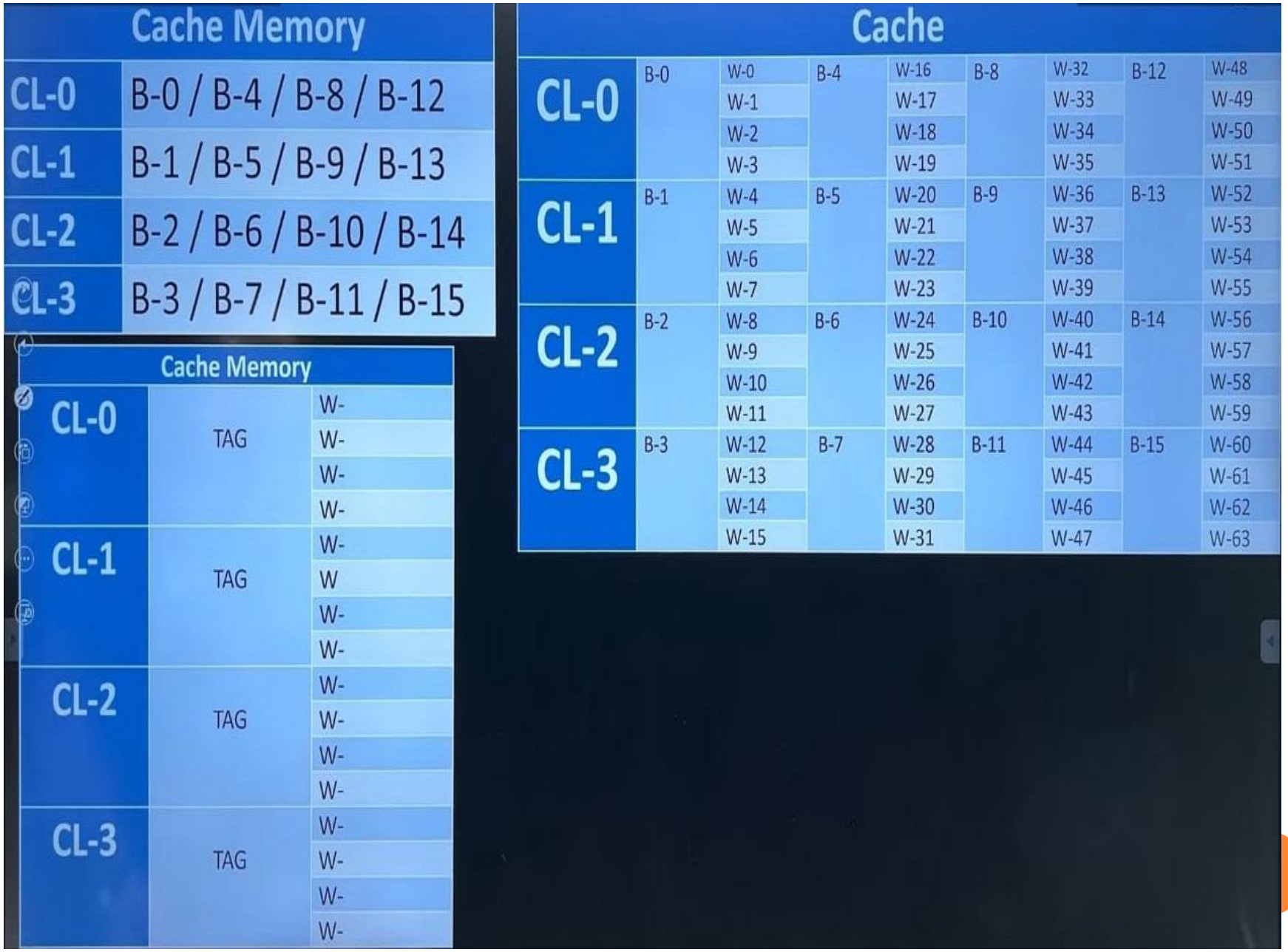
There are several ways including:

1. Direct Mapping
2. Associative mapping
3. Set-associative mapping

Direct Mapping:



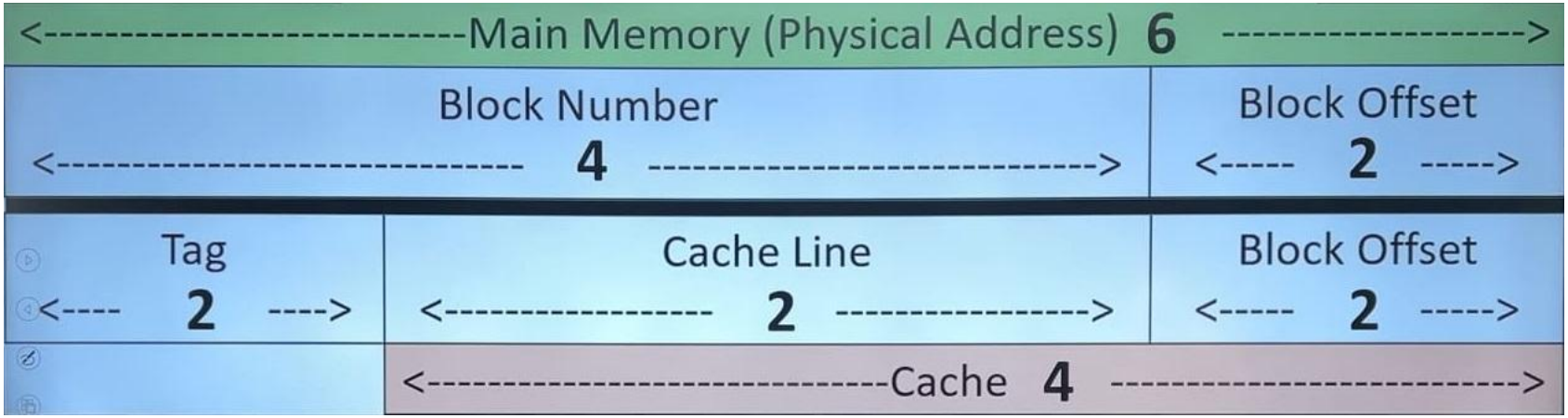

CL-0
CL-1
CL-2
CL-3
B-0
B-1
B-2
B-3
B-4
B-5
B-6
B-7
B-8
B-9
B-10
B-11
B-12
B-13
B-14
B-15



Cache Memory	
CL-0	B-0 / B-4 / B-8 / B-12
CL-1	B-1 / B-5 / B-9 / B-13
CL-2	B-2 / B-6 / B-10 / B-14
CL-3	B-3 / B-7 / B-11 / B-15

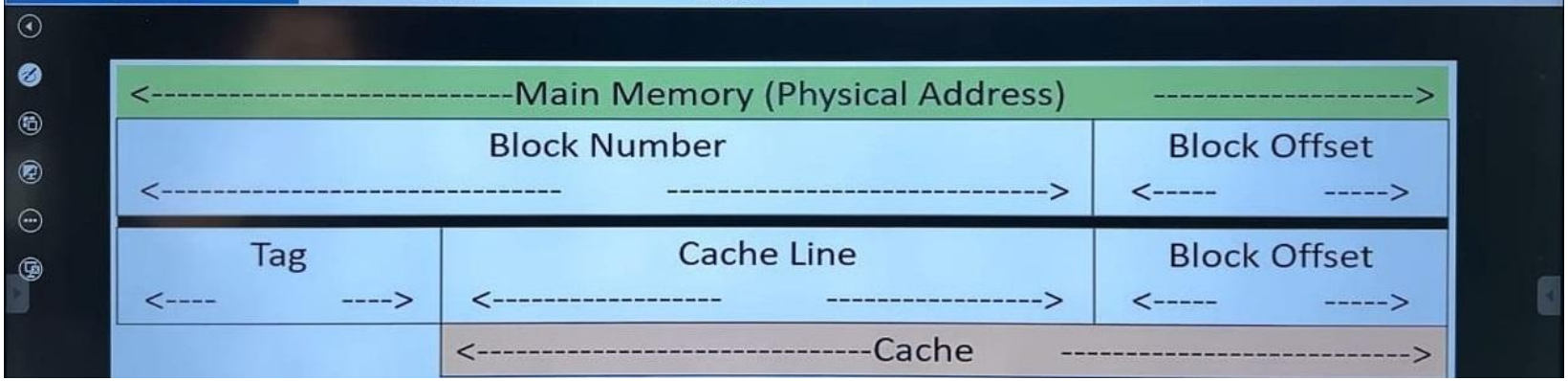
Cache Memory		
CL-0	TAG	W-
		W-
		W-
		W-
CL-1	TAG	W-
		W
		W-
		W-
CL-2	TAG	W-
		W-
		W-
		W-
CL-3	TAG	W-
		W-
		W-
		W-

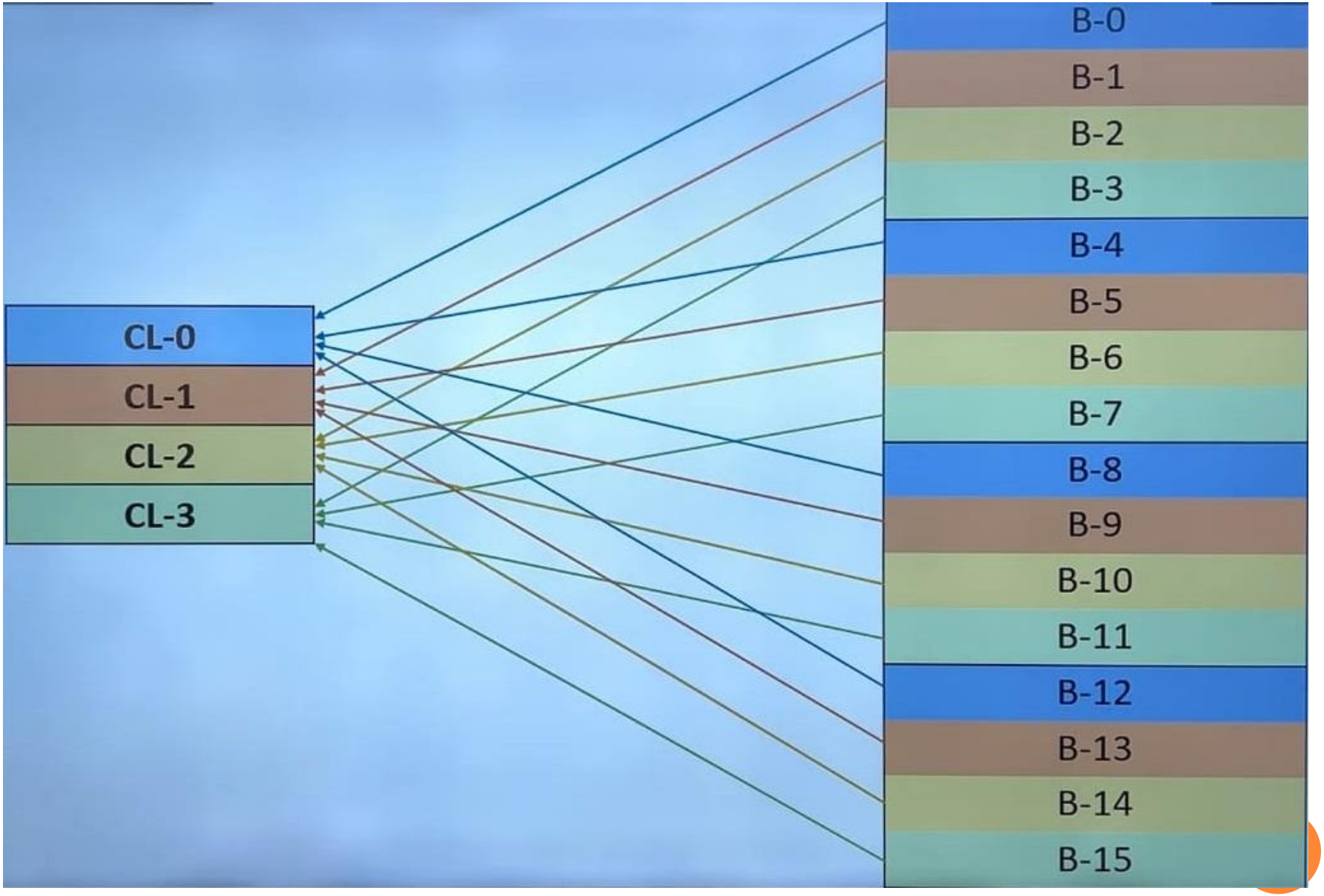
Cache								
CL-0	B-0	W-0	B-4	W-16	B-8	W-32	B-12	W-48
		W-1		W-17		W-33		W-49
		W-2		W-18		W-34		W-50
		W-3		W-19		W-35		W-51
CL-1	B-1	W-4	B-5	W-20	B-9	W-36	B-13	W-52
		W-5		W-21		W-37		W-53
		W-6		W-22		W-38		W-54
		W-7		W-23		W-39		W-55
CL-2	B-2	W-8	B-6	W-24	B-10	W-40	B-14	W-56
		W-9		W-25		W-41	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


<----------Main Memory (Physical Address) 6 ---------->
Block Number
<---------- 4 ---------->
Block Offset
<----- 2 ----->
Tag
<---- 2 ---->
Cache Line
<---------- 2 ---------->
Block Offset
<----- 2 ----->
<----------Cache 4 ---------->

- The main advantage of direct mapping is its simplicity. Since each main m/y block maps to only one specific cache line there is no need for complicated search algorithms. This makes direct mapping relatively fast and efficient.

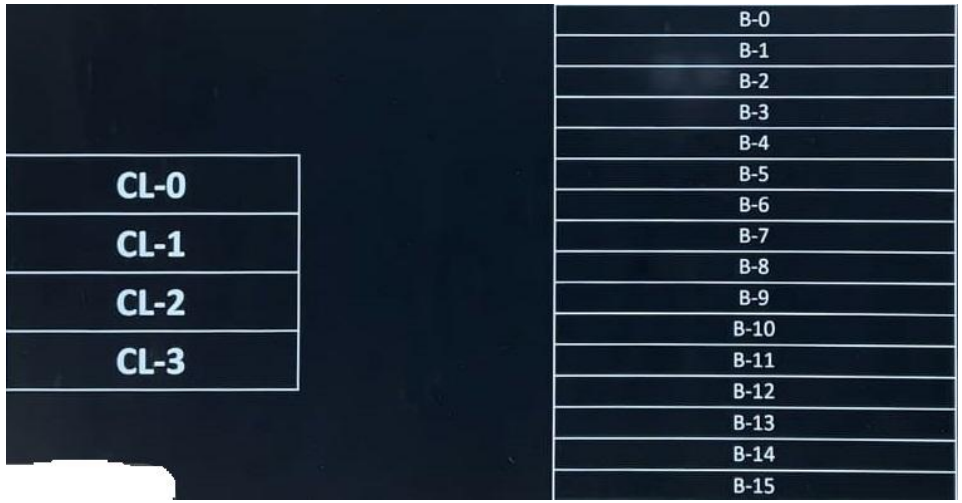
MM Size	Cache Size	Block Size	No of bits in Tag	Tag Directory Size
~~16 GB~~	32 MB	4 KB		
128 MB	256 KB	512 B		
32 GB	128 MB	1 KB		
256 MB	16 KB	1 KB		
4 GB	8 MB	2 KB		
512 KB	2 KB	128 B		




CL-0
CL-1
CL-2
CL-3
B-0
B-1
B-2
B-3
B-4
B-5
B-6
B-7
B-8
B-9
B-10
B-11
B-12
B-13
B-14
B-15

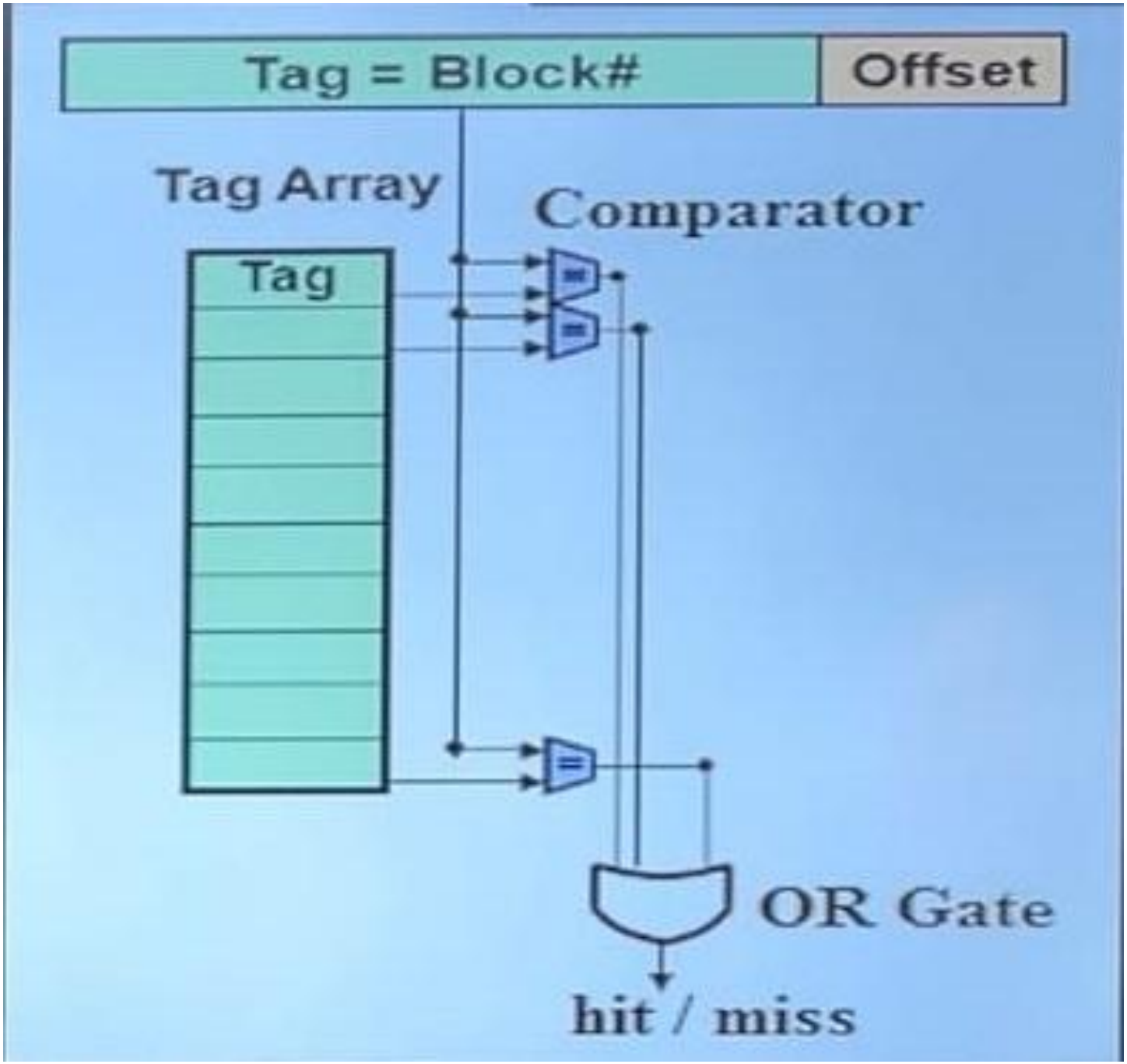
Associative Mapping

- Resolution of the problem of conflict miss.
- Block memory can be mapped to any freely available cache line.
- Fully Associative is more flexible than direct mapping
- Also known as many to many mapping.
- Conflict miss doesn't occur while capacity miss may occur.

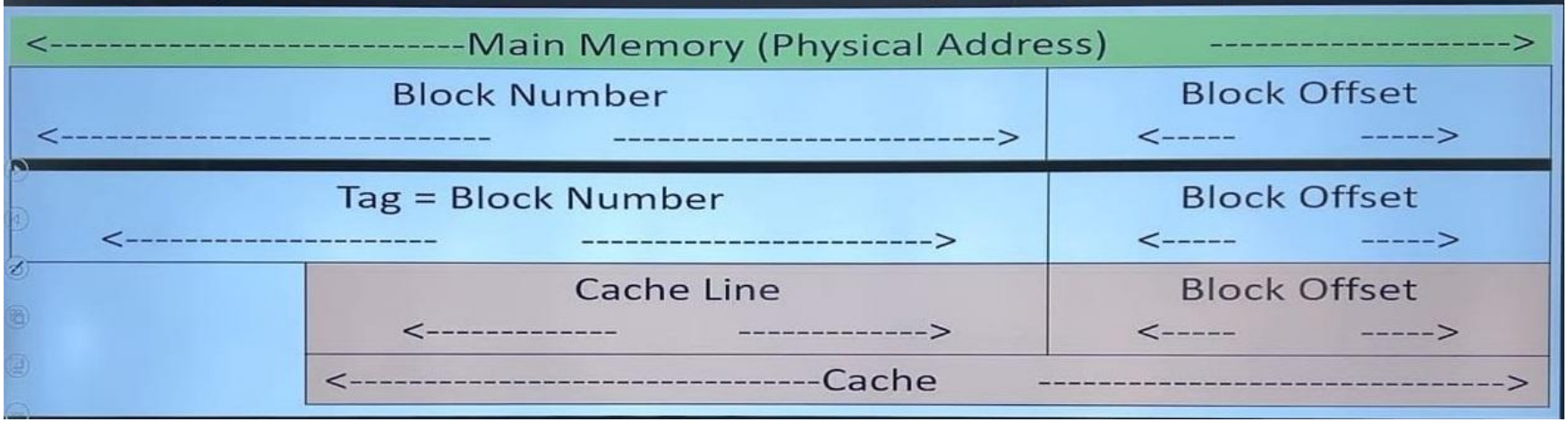


Cache Memory		
CL-0	TAG = Block no	W-
		W-
		W-
		W-
CL-1	TAG = Block no	W-
		W
		W-
		W-
CL-2	TAG = Block no	W-
		W-
		W-
		W-
CL-3	TAG = Block no	W-
		W-
		W-
		W-

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


Tag = Block#
Offset
Tag Array
Comparator
Tag
OR Gate
hit / miss

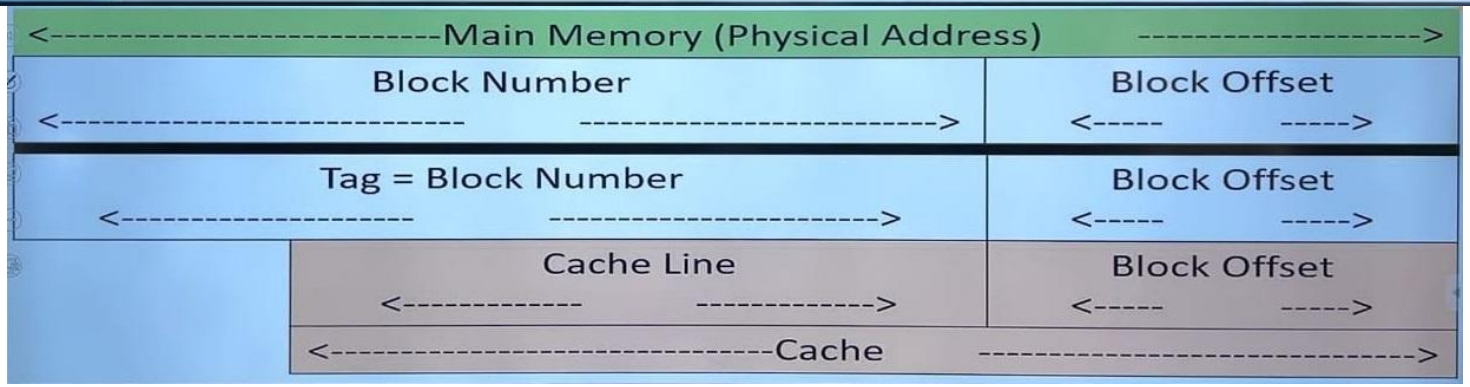
Q Consider a fully associative mapped cache of size 16 KB with block size 256 bytes. The size of main memory is 128 KB. Find out the: Number of bits in tag and Tag directory size?



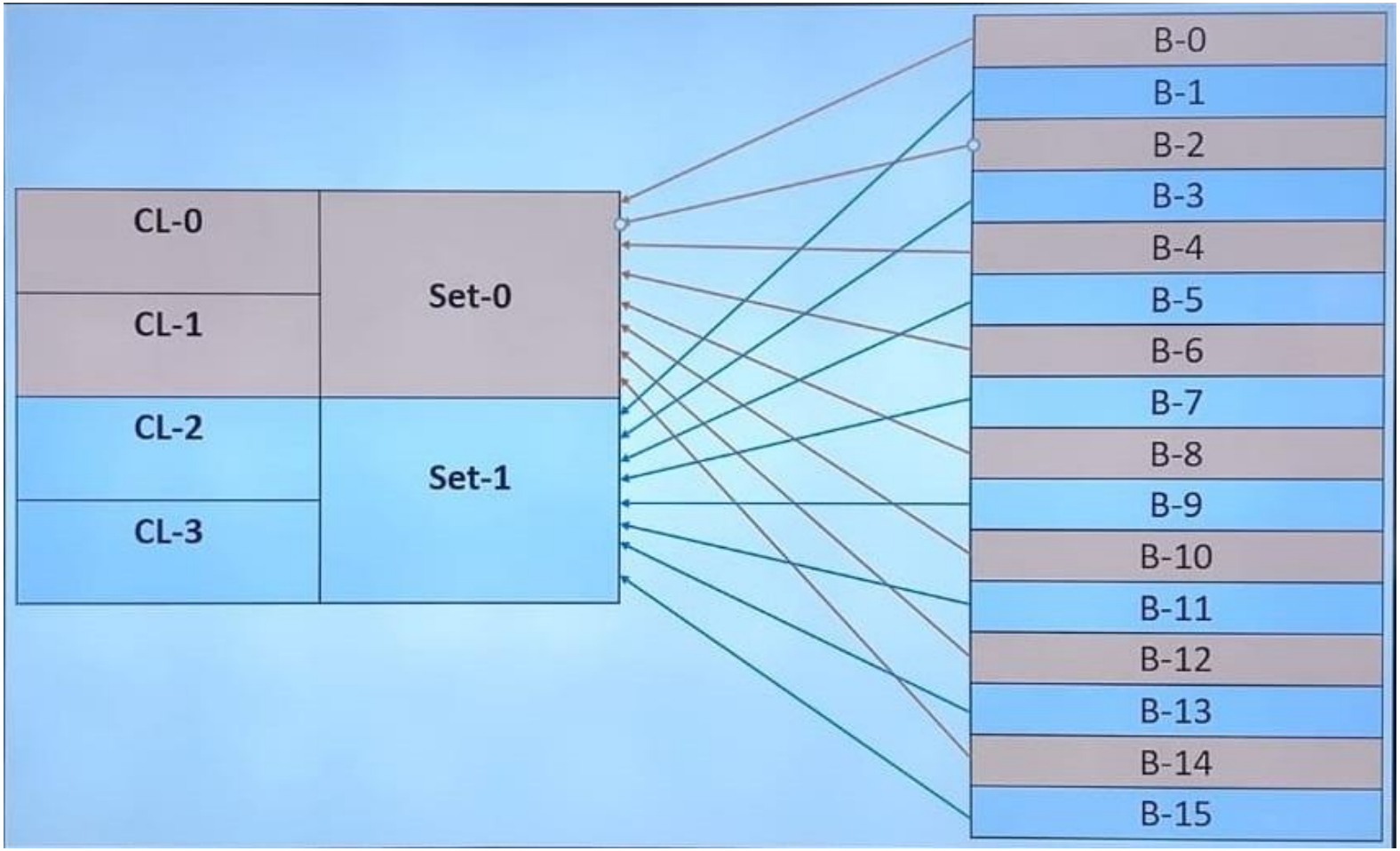
Disadvantage

- Hardware cost is high
- Tag directory size is more as compared to direct mapping.

MM Size	Cache Size	Block Size	No of bits in Tag	Tag Directory Size	Comp
128 KB	16 KB	256 B	9	9 * 64	64
32 GB	32 KB	1 KB	25	25 * 32	32
128 MB	512 KB	1 KB	17	512 *17	512
16 GB	?	4 KB	22	?	?
64MB	?	64 KB	10	?	?
?	512 KB		7	?	?



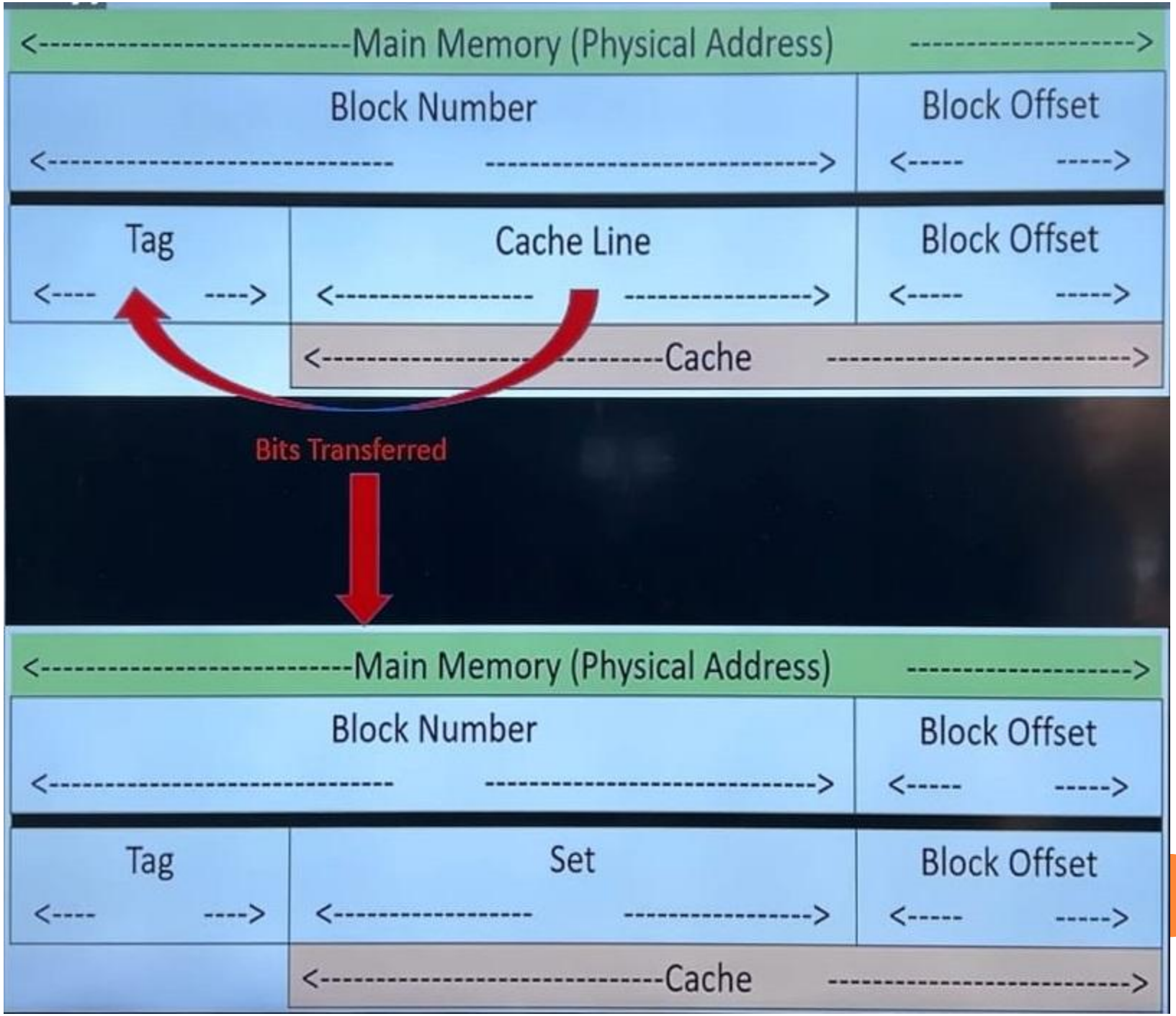
SET-ASSOCIATIVE MAPPING



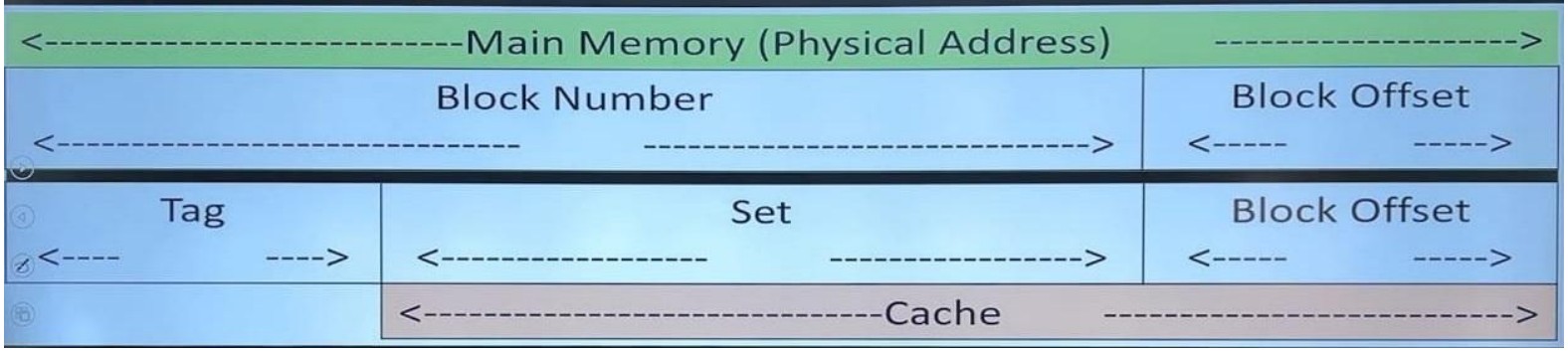
Stage 1: Fixed for sets (Direct Mapping)
Stage 2: flexible (Associative Mapping)

Set-associative mapping

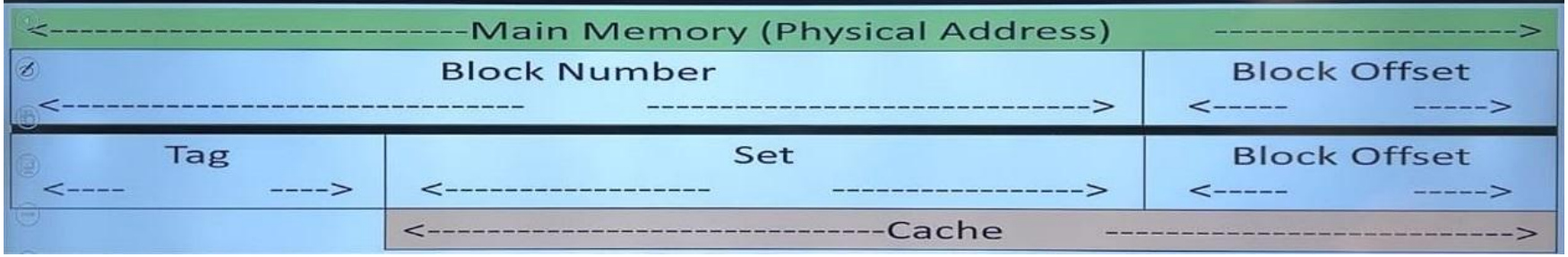
- In K-way set associative mapping, cache lines are grouped into sets where each set contains K number of lines
- A particular block of main m/y can map to only one particular set of cache
- However, within a set, the m/y block can map to any freely available cache line.


Main Memory (Physical Address)
Block Number
Block Offset
Tag
Cache Line
Block Offset
Cache
Bits Transferred
Main Memory (Physical Address)
Block Number
Block Offset
Tag
Set
Block Offset
Cache

Q Consider the main memory size is of 128 KB, the cache size is of 16 KB, the block size is of 256 B, the set size is 2. Find Tag.



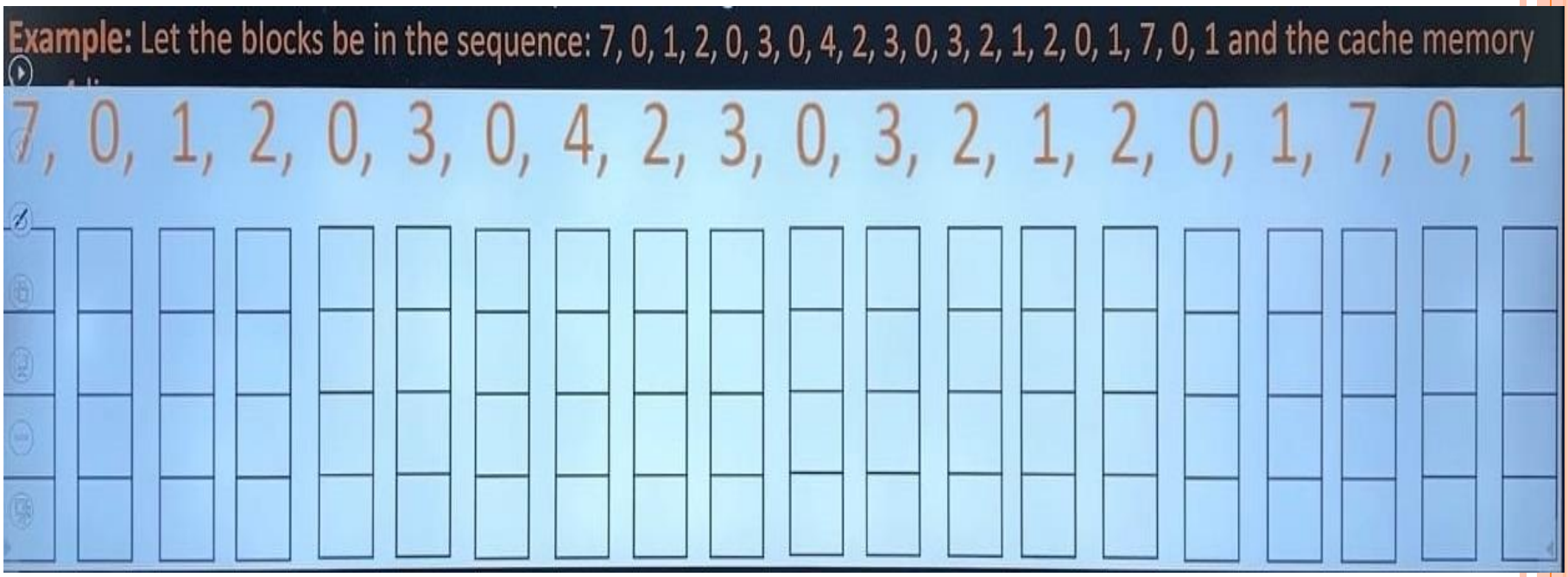
MM Size	Cache Size	Block Size	No of bits in Tag	Tag Directory Size	Set Associative
128 KB	16 KB	256 B			2-way
32 GB	32 KB	1 KB			4-way
	512 KB	1 KB	7		8-way
16 GB		4 KB	10		4-way
64MB			10		4-way
	512 KB		7		8-way



Cache Replacement Policies

- FIFO
- Optimal
- LRU
- MRU

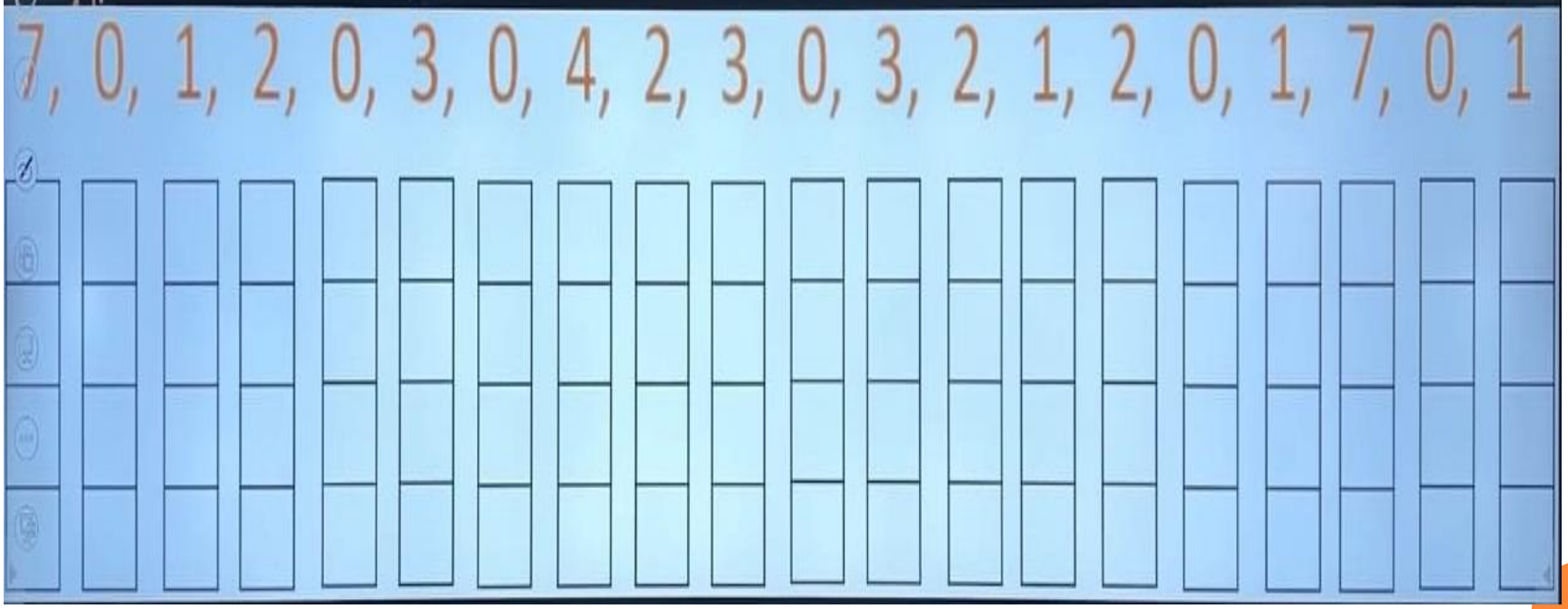
- Note: In direct mapping no replacement policy exists.


Example: Let the blocks be in the sequence: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1 and the cache memory
7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

Optimal Algorithm

- The page which will not be used for the longest period of time in future references will be replaced first.
- The optimal algorithm will provide the best performance but it is difficult to implement as it requires the future knowledge of pages which is not possible.
- It is used as a benchmark for cache replacement algorithms.

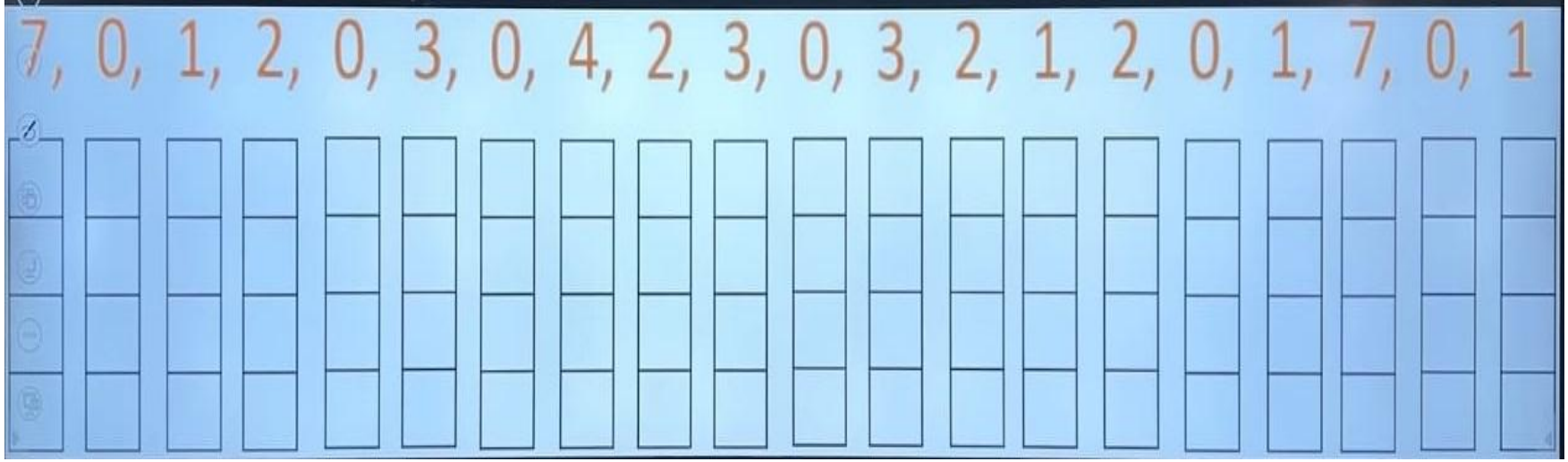
Example: Let the blocks be in the sequence: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1 and the cache memory



LRU (Least Recently Used)

- The page which was not used for the longest period of time in the past will get replaced first.

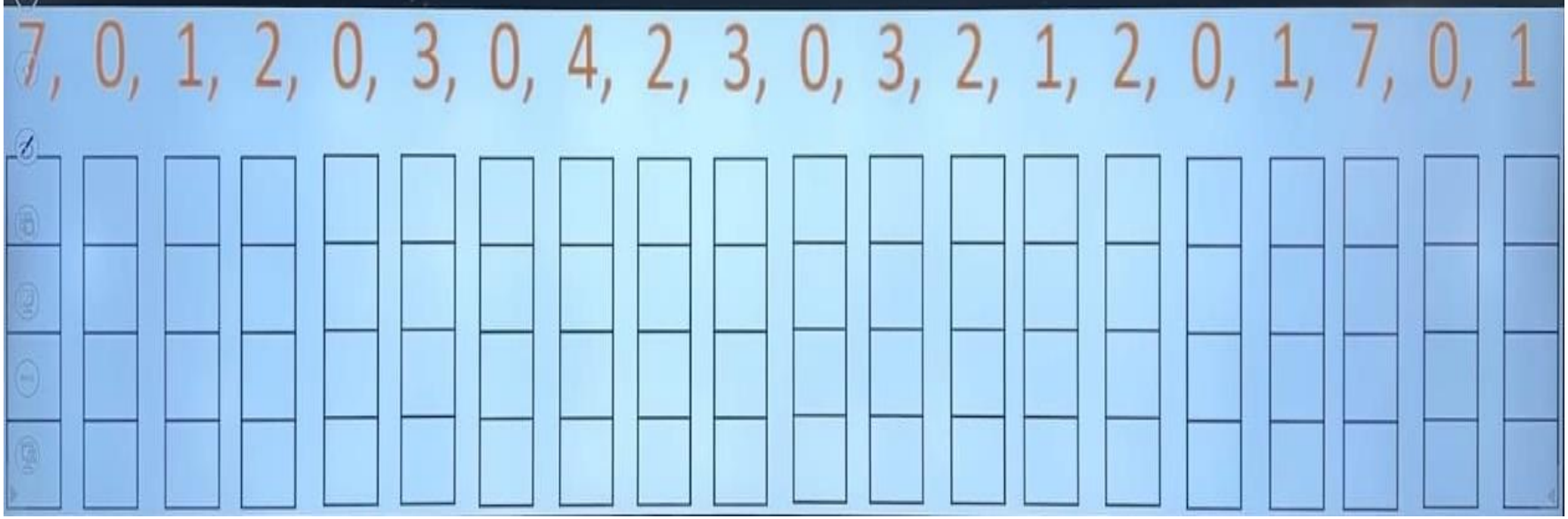
Example: Let the blocks be in the sequence: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1 and the cache memory has 4 lines.



Most Recently Used (MRU)

- The page which was used recently will be replaced first.

Example: Let the blocks be in the sequence: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1 and the cache memory has 4 lines.

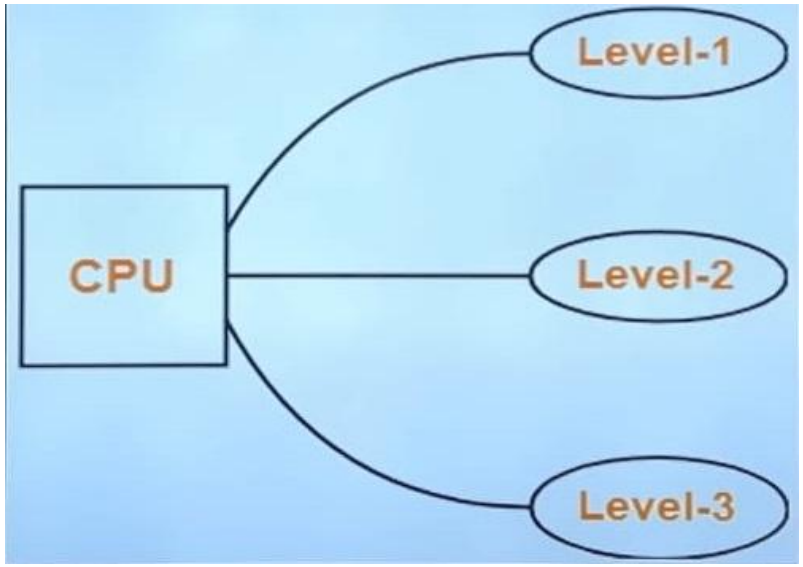


Types of Miss:

- Compulsory Miss: CPU demands a block for the first time.
- Conflict Miss: When several blocks are mapped to same set.
- Capacity Miss: No available space is in cache.

Memory Organization

- Simultaneous Access
- Hierarchical Access



Example: Calculate the EMAT for a machine with a cache hit rate of 80% where cache access time is 5ns and main memory access time is 100ns, both for simultaneous and hierarchical access.

Cache Coherence Problem

- If multiple copy of same data is maintained at different level of memories, then inconsistency may occur, this problem is known as cache coherence problem.
- Solution: Write through and Write back