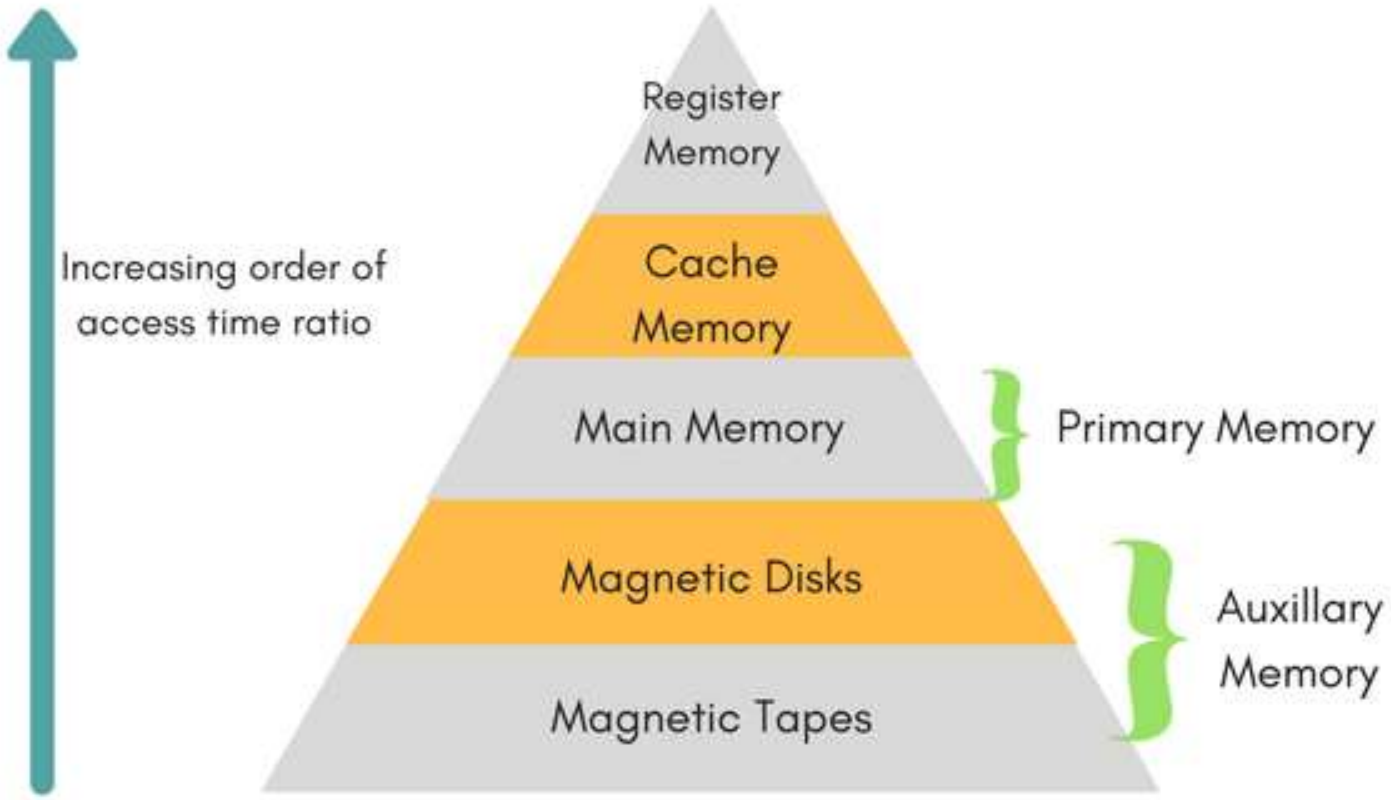
CHAPTER 2
MEMORY ORGANIZATION

Memory Organization in Computer Architecture

- A memory unit is the collection of storage units or devices together. The memory unit *stores the binary information in the form of bits*. Generally, memory/storage is classified into 2 categories:
  - **Volatile Memory**: This *loses its data*, when power is switched off.
  - **Non-Volatile Memory**: This is a *permanent storage* and does not lose any data when power is switched off.

Memory Hierarchy



The total memory capacity of a computer can be visualized by hierarchy of components. The memory hierarchy system consists of all storage devices contained in a computer system from the slow Auxiliary Memory to fast Main Memory and to smaller Cache and register memory.

- **Auxiliary memory** access time is generally **1000 times** that of the main memory, hence it is at the bottom of the hierarchy.

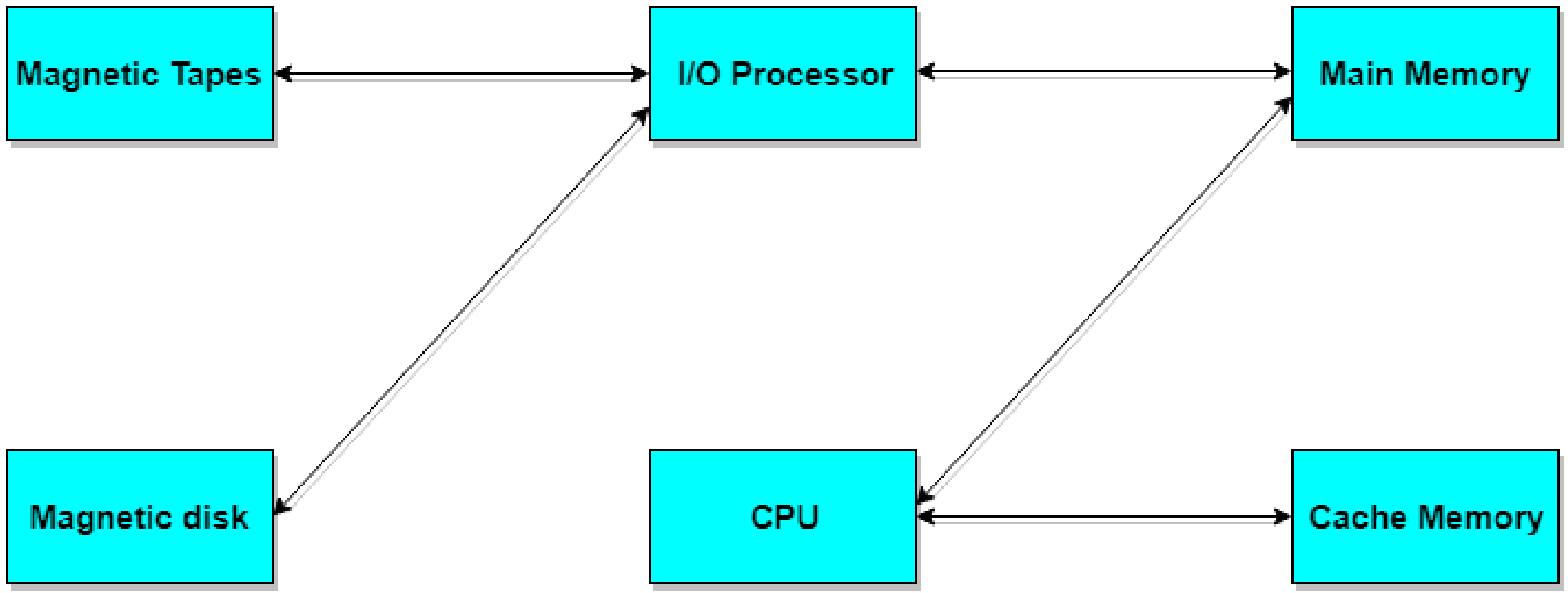
- The **main memory** occupies the central position because it is *equipped to communicate directly with the CPU and with auxiliary memory devices* through Input/output processor (I/O).

- When the program not residing in main memory is needed by the CPU, they are brought in from auxiliary memory. Programs not currently needed in main memory are transferred into auxiliary memory to provide space in main memory for other programs that are currently in use.

- The **cache memory** is used to *store program data which is currently being executed in the CPU*. Approximate access time ratio between cache memory and main memory is about **1 to 7~10**

Primary memory vs Secondary memory

Primary memory	Secondary memory
It is the main memory where the data and information are stored temporarily.	It refers to the external memory where data is stored permanently.
Data is directly accessed by the processing unit.	Data cannot be accessed directly by the processor.
It's a volatile memory meaning data cannot be retained in case of power failure.	It's a non-volatile memory so data can be retained even after power failure.
Memory is stored in semiconductor chips which are relatively expensive.	Memory is stored in external storage devices such as hard disks, flash drives, etc.
It can be categorized into cache memory and random access memory (RAM).	They are permanent storage devices such as CD, DVD, HDD, floppy disk, etc.
It's relatively faster than secondary memory because of its volatile nature.	They are usually slower than primary memory. It's like a backup memory.
It holds data or information that is currently being used by the processing unit.	It stores substantial amount of data and information, ranging from gigabytes to terabytes.


Magnetic Tapes
I/O Processor
Main Memory
Magnetic disk
CPU
Cache Memory

Characteristics of Memory

1. Location - (CPU, Internal, External)

2. Storage Capacity – (Word size i.e size of each location and No. of words i.e No. of locations)

3. Unit of Transfer – (Size of data transferred in 1 clock cycle. Depends on Bus size)

4. Access method – (Random, Sequential, Direct)

5. Performance – (Speed or Data transfer rate. It depends on Access time and Cycle time)
    1. Access time – (Time b/w providing the address and getting the valid data from the memory)
    2. Cycle time – (Time b/w 2 access i.e Time required by memory to recover before next access)

6. Physical material –(Semiconductor, Magnetic, Optical)

Memory Access Methods

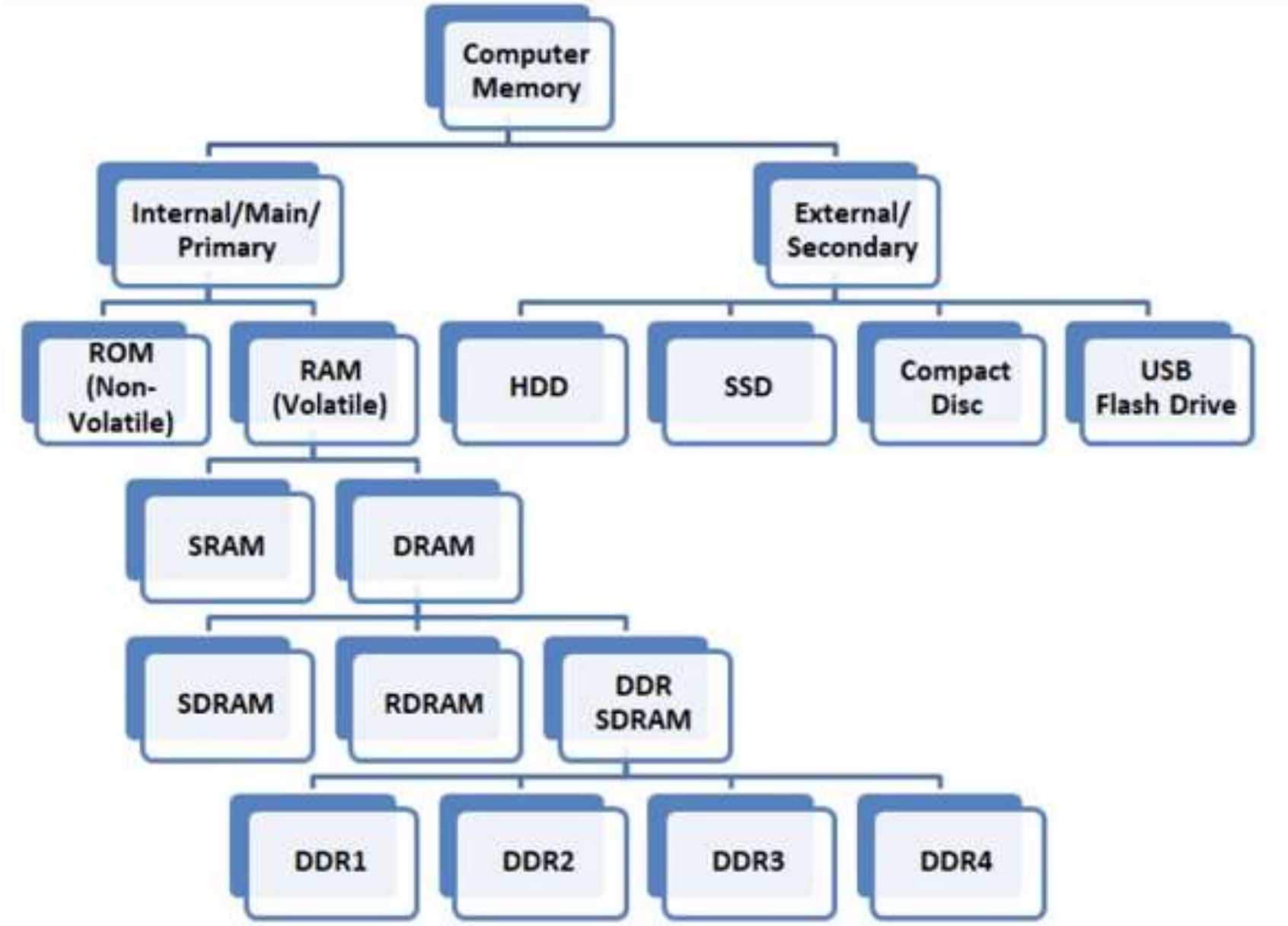
- Each memory type, is a collection of numerous memory locations. To access data from any memory, *first it must be located and then the data is read from the memory location.* Following are the methods to access information from memory locations:

  - **Random Access**: Main memories are random access memories, in which *each memory location has a unique address*. Using this unique address any memory location can be reached in the same amount of time in any order.

  - **Sequential Access**: This methods allows *memory access in a sequence or in order*.

  - **Direct Access**: In this mode, *information is stored in tracks, with each track having a separate read/write head*.

Types of Memory



- There are basically two kinds of internal memory: ROM and RAM.

- **ROM** stands for read-only memory. It is non-volatile, which means it can retain data even without power. It is used mainly to start or boot up a computer.

- Once the operating system is loaded, the computer uses **RAM**, which stands for random-access memory, which temporarily stores data while the central processing unit (CPU) is executing other tasks.
- With more RAM on the computer, the less the CPU has to read data from the external or secondary memory (storage device), allowing the computer to run faster.
- RAM is fast but it is volatile, which means it will not retain data if there is no power. It is therefore important to save data to the storage device before the system is turned off.

Types of RAM

Integrated RAM chips are available in two form:

1. SRAM(Static RAM) is made up of Flip-Flops. It keeps data in the memory as long as power is supplied to the system unlike DRAM, which has to be refreshed periodically. As such, SRAM is faster but also more expensive, making DRAM the more prevalent memory in computer systems.

2. DRAM(Dynamic RAM) is widely used as a computer's main memory. It is a type of RAM which allows you to store each bit of data in a separate capacitor within a specific integrated circuit. This type of RAM is a volatile memory that needs to be refreshed with voltage regularly else it loses the information stored on it.

SRAM vs DRAM

SRAM	DRAM
SRAM has **lower access time**, so it is faster compared to DRAM.	DRAM has **higher access time**, so it is slower than SRAM.
SRAM is **costlier** than DRAM.	DRAM **costs less** compared to SRAM.
SRAM **requires a constant power supply**, which means this type of memory consumes more power.	DRAM offers **reduced power consumption** because the information is stored in the capacitor.
It is a complex internal circuitry, and it **offers less storage capacity** compared to the same physical size of a DRAM memory chip.	It is a small internal circuitry in the one-bit memory cell of DRAM. Thus **large storage capacity is available.**
No refreshing is required	**Refreshing** is required
Mainly used for **Cache memory**	Mainly used for **Main memory**

Types of DRAM

1. **Synchronous DRAM (SDRAM):** "synchronizes" the memory speed with CPU clock speed so that the memory controller knows the exact clock cycle when the requested data will be ready. SDRAM performs its operations on the rising edge of the clock signal

These memories operate at the CPU-memory bus without imposing wait states which allows the CPU to perform more instructions at a given time. Typical SDRAM transfers data at speeds up to 133 MHz.

2. **Double-Data-Rate SDRAM (DDR SDRAM):** This faster version of SDRAM performs its operations on both edges(rising & falling) of the clock signal;

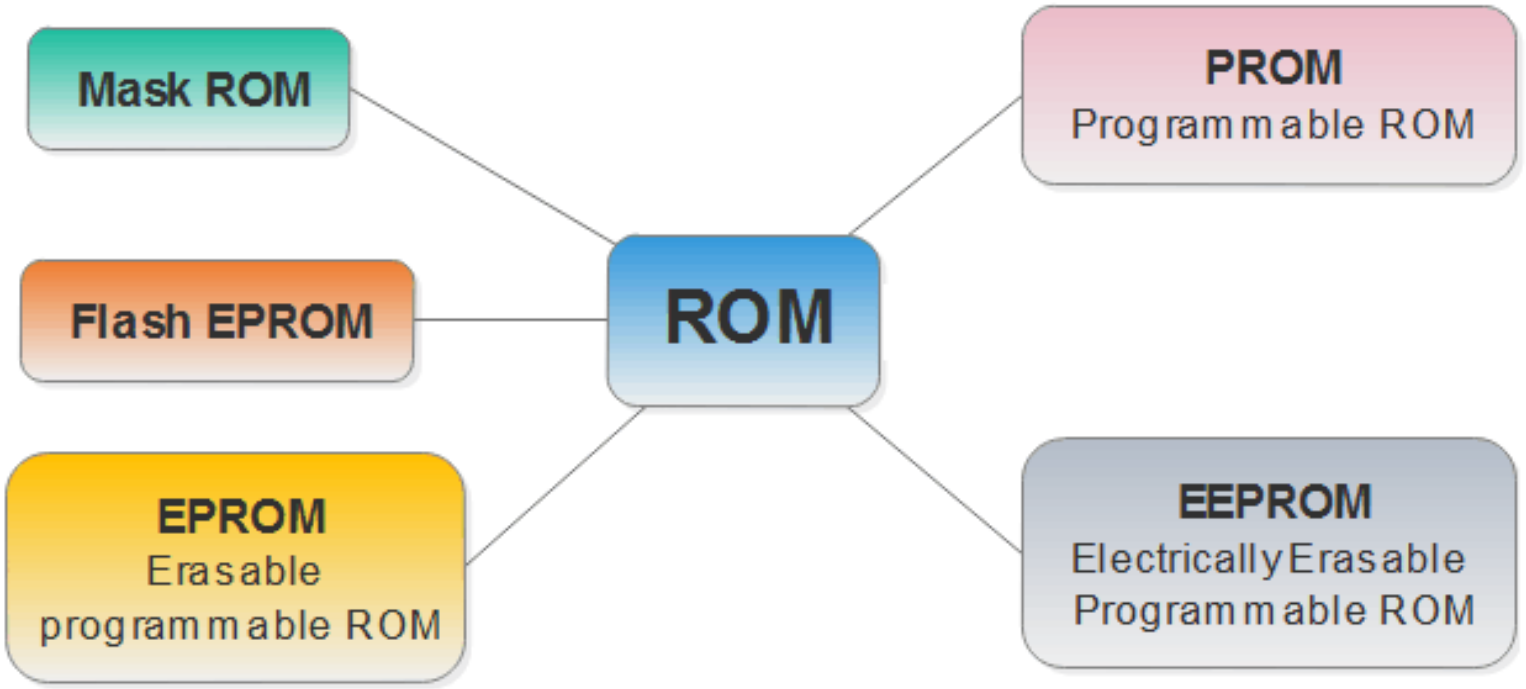
Since they transfer data on both edges of the clock, the data transfer rate is doubled. To access the data at high rate, the memory cells are organized into two groups & each group is accessed separately.

3. **RAM-**based **SOLID STATE DRIVE** (SSD-RAM or RD-RAM):

It uses DRAM or SRAM chips, both of which are volatile. This means that a RAM-based drive will lose its contents when the power is turned off.

To preserve the contents of a RAM-based solid state drive, data is copied from volatile memory to non-volatile memory upon instruction or when the drive is powered down. Generally, a RAM-based SSD have batteries that keep the data alive long enough for it to be copied to non-volatile memory in the event that power accidently gets shut off.

Types of ROM


Mask ROM
Flash EPROM
EPROM
Erasable
programmable ROM
ROM
PROM
Programmable ROM
EEPROM
Electrically Erasable
Programmable ROM

1. **MROM (Masked ROM):** The very first ROMs were hard-wired devices that contained a pre-programmed set of data or instructions. These kind of ROMs are known as masked ROMs, which are inexpensive.

2. **PROM (Programmable Read Only Memory):** PROM is read-only memory that can be modified only once by a user. The user buys a blank PROM and enters the desired contents using a PROM program. Inside the PROM chip, there are small fuses which are burnt open during programming. It can be programmed only once and is not erasable.

3. **EPROM (Erasable Programmable ROM):** It allow you to write and rewrite them many times. These chips feature a quartz window through which a specialized EPROM programmer emits a specific frequency of ultraviolet light.

   This light burns out all the tiny charges in the EPROM to reopen its circuits. This exposure effectively renders the chip blank again, after which you can reprogram it according to the same process as a PROM

4. **EEPROM (Electrically Erasable and Programmable Read Only Memory)** is a special type of PROM that can be erased by exposing it to an electrical charge. Like other types of PROM, EEPROM retains its contents even when the power is turned off. However, EEPROM is not as fast as RAM.

5. **Flash ROM:** Flash memory is a modern type of EEPROM. Flash memory can be erased and rewritten faster than ordinary EEPROM. Flash can keep its data intact with no power at all.

Flash memory is one kind of Non-volatile random-access memory. It is slower than RAM but faster than hard drives. It is mostly used in small electronics because it is small and has no moving parts.

Memory Allocation Techniques:

1. **First-Fit:** First sufficient block is allocated from the top of Main Memory.

2. **Best-Fit:** Allocate the process to the partition which is the first smallest sufficient partition among the free available partition.

3. **Worst-Fit:** Allocate the process to the partition which is the largest sufficient among the freely available partitions available in the main memory.

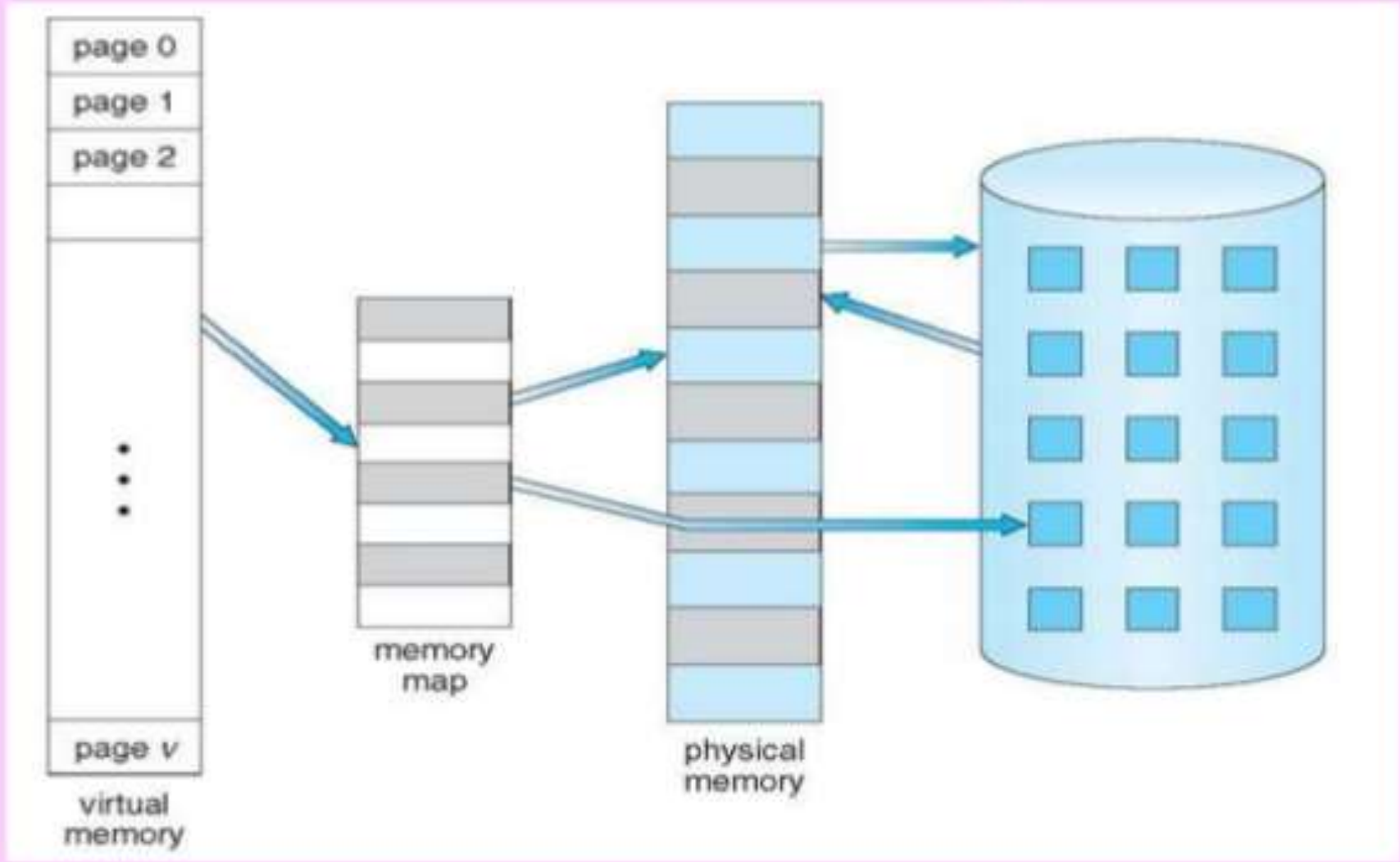
VIRTUAL MEMORY

- Virtual memory is a valuable concept in computer architecture that allows you to run large, sophisticated programs on a computer even if it has a relatively small amount of RAM.

- A PC that's low on memory can run the same programs as one with abundant RAM, although more slowly.

- Virtual memory is used to give programmers the illusion that they have a very large memory even though the computer has a small main memory. It makes the task of programming easier because the programmer no longer needs to worry about the amount of physical memory available.

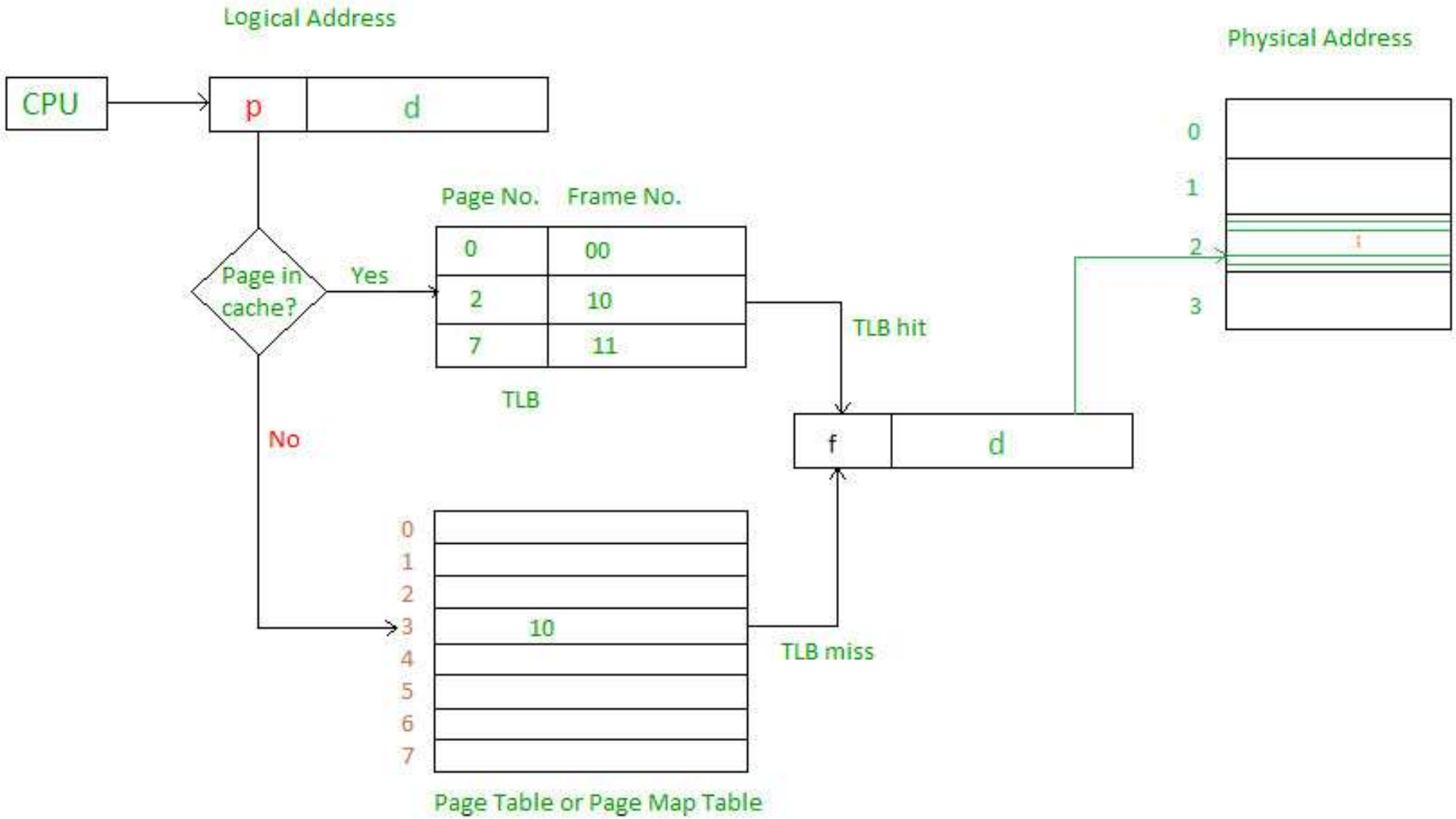
Virtual Memory



PAGING

- Paging is a memory management scheme that eliminates the need for contiguous allocation of physical memory. This scheme permits the physical address space of a process to be non – contiguous.
  - Logical Address or Virtual Address: An address generated by the CPU
  - Physical Address: An address actually available on memory unit

- The mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device and this mapping is known as paging technique.
  - The Physical Address Space is conceptually divided into a number of fixed-size blocks, called **frames**.
  - The Logical address Space is also split into fixed-size blocks, called **pages**.
  - Page Size = Frame Size


Logical Address
Physical Address
CPU
p
d
Page No.
Frame No.
Page in cache?
Yes
No
0
2
7
00
10
11
TLB
TLB hit
f
d
0
1
2
3
10
4
5
6
7
TLB miss
Page Table or Page Map Table
0
1
2
3

Page Replacement Policies

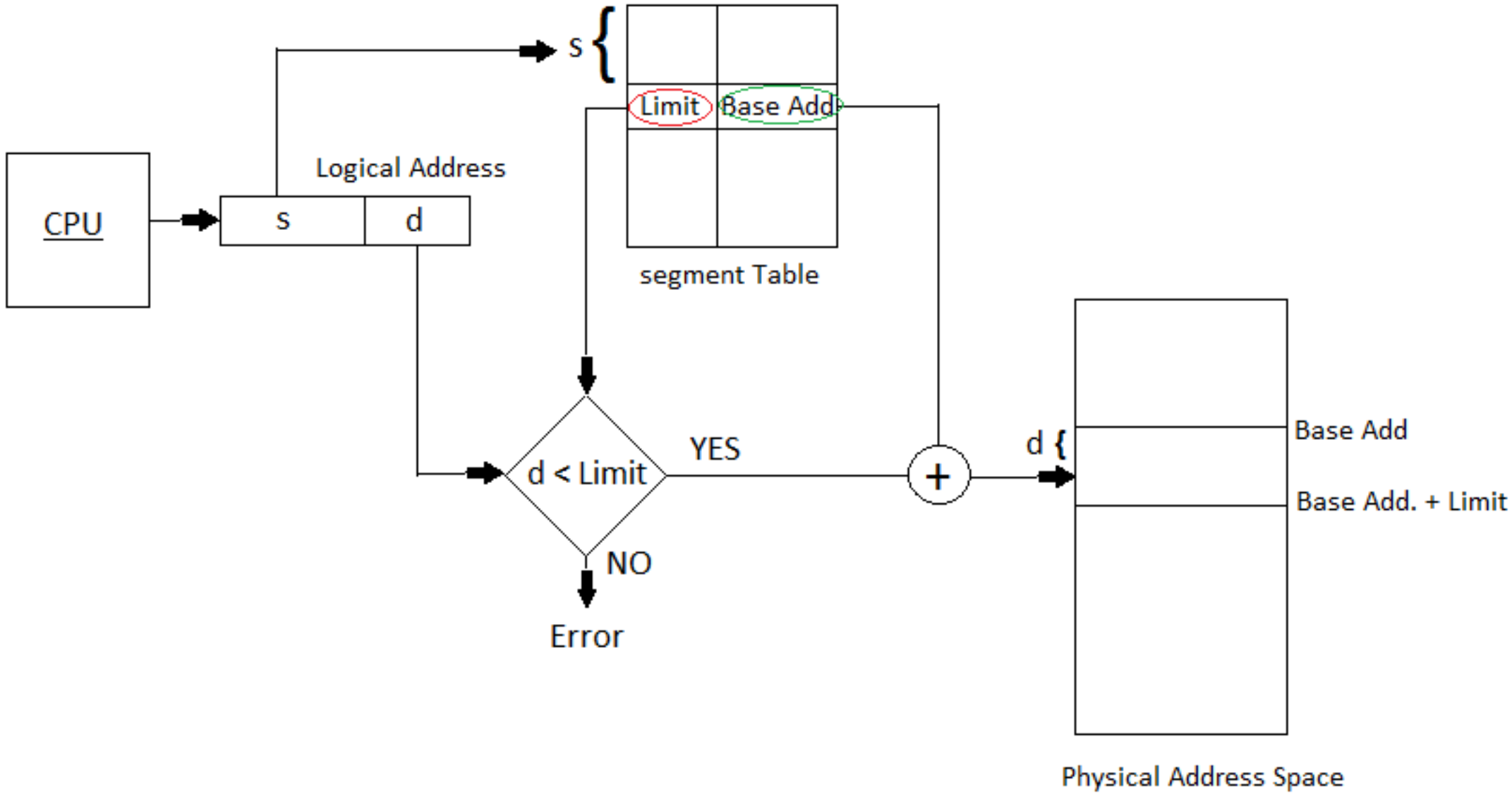
- FIFO
- LRU
- LFU
- OPTIMAL

- If we find the required page in the Main Memory while C.P.U wants to access the page then it is a Page Hit.
- If we do not find the required pages then they are called as Page Faults. At Page Faults, we have to look into the Secondary Memory and fetch the required pages and load into the Main Memory.

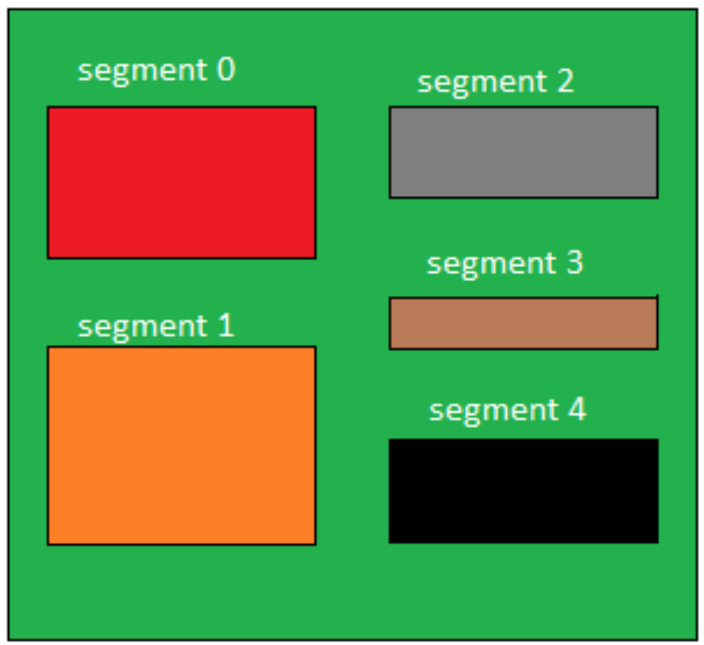
SEGMENTATION

- A process is divided into Segments. The chunks that a program is divided into which are not necessarily all of the same sizes are called segments.
- A table stores the information about all such segments and is called Segment Table.

- **Segment Table –** It maps two-dimensional Logical address into one-dimensional Physical address. It's each table entry has:
  - **Base Address:** It contains the starting physical address where the segments reside in memory.
  - **Limit:** It specifies the length of the segment.


CPU
Logical Address
s
d
s
Limit
Base Add
segment Table
d < Limit
YES
NO
Error
+
d
Base Add
Base Add. + Limit
Physical Address Space

Logical View of Segmentation

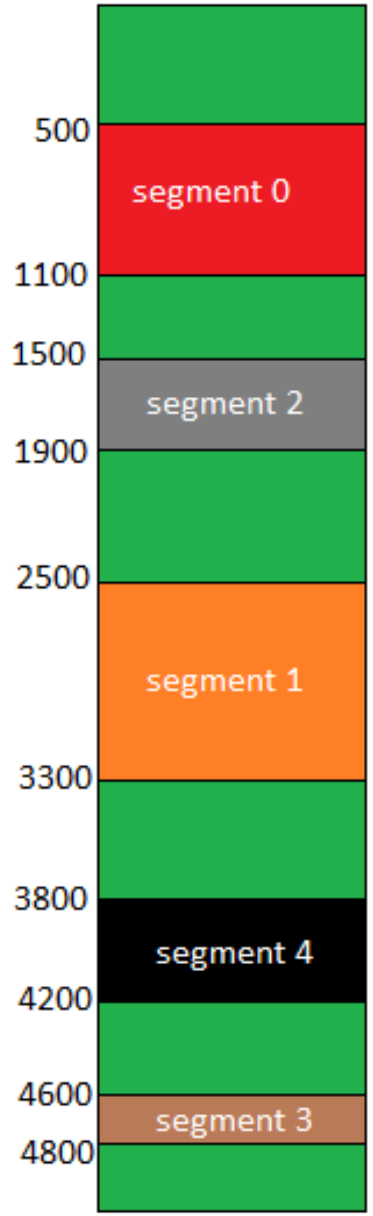


Logical Address Space

Segment Number

	base address	Limit
0	500	600
1	2500	800
2	1500	400
3	4600	200
4	3800	400

Segment Table



Physical Address Space

Interleaved memory.

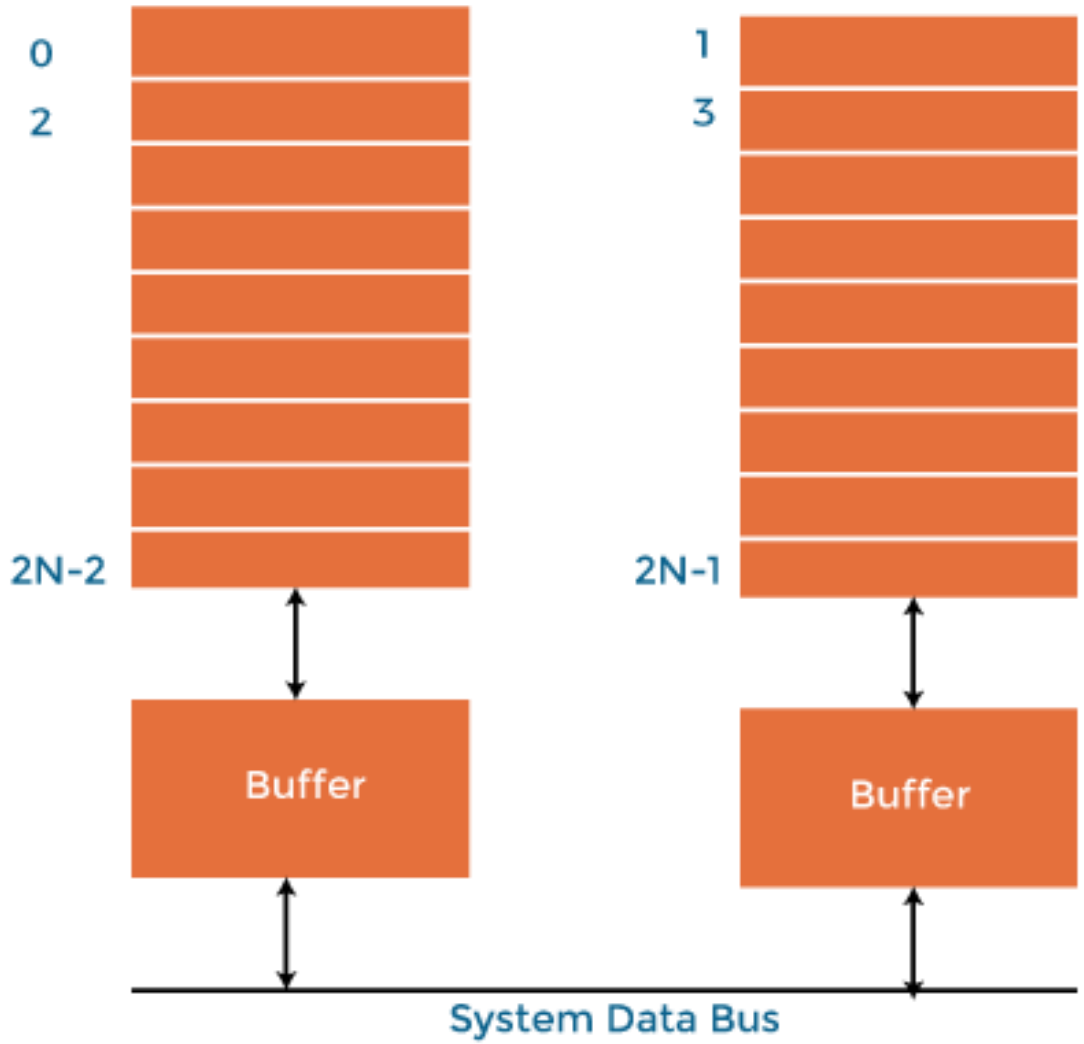
- Interleaved memory is *designed to compensate for the relatively slow speed of dynamic random-access memory (D-RAM)* or core memory by *spreading memory addresses evenly across memory banks*.

- In this way, *contiguous memory reads and writes use each memory bank, resulting in higher memory throughput* due to reduced waiting for memory banks to become ready for the operations.

- It is an abstraction technique that *divides memory into many modules* such that *successive words in the address space are placed in different modules*.

- Example:
  - Suppose we have 4 memory banks, each containing 256 bytes, and then the Block Oriented scheme (no interleaving) will assign virtual addresses 0 to 255 to the first bank and 256 to 511 to the second bank.
  - But in Interleaved memory, virtual address 0 will be with the first bank, 1 with the second memory bank, 2 with the third bank and 3 with the fourth, and then 4 with the first memory bank again.

- Hence, the *CPU can access alternate sections immediately without waiting* for memory to be cached. There are multiple memory banks that take turns for the supply of data.
- In the above example of 4 memory banks, *data with virtual addresses 0, 1, 2 and 3 can be accessed simultaneously as they reside in separate memory banks*. Hence *we do not have to wait to complete a data fetch to begin the next operation*.



Interleaved Memory with 2 memory banks

- When the processor requests data from the main memory, a block (chunk) of data is transferred to the cache and then to processor. So *whenever a cache miss occurs, the data is to be fetched from the main memory*. But main memory is relatively slower than the cache. So to improve the access time of the main memory, interleaving is used.
- For example, we can access all four modules at the same time, thus achieving parallelism. This method uses memory effectively.

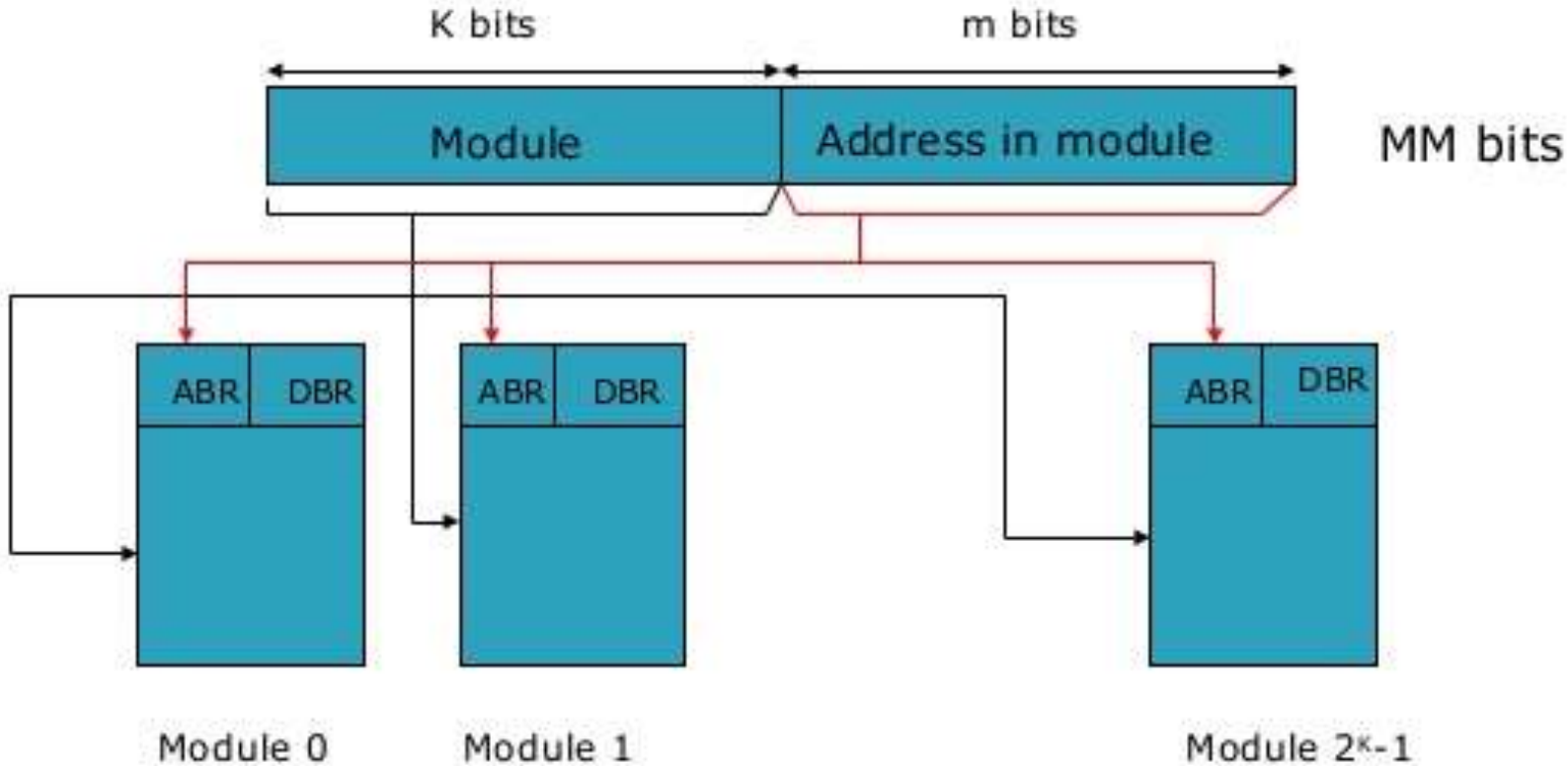
Types of Interleaved Memory

- In an operating system, interleaved memory could be of the following 2 types:
  1. *High order interleaving*
  2. *Low order interleaving*

1. High order interleaving

- In high order memory interleaving, the *most significant bits (MSB) of the memory address decides memory banks* where a particular location resides.
- The *least significant bits are sent as addresses to each chip*.
- One problem is that *consecutive addresses tend to be in the same chip*.
- The *maximum rate of data transfer is limited by the memory cycle time.*
- It is also known as ***Memory Banking***.

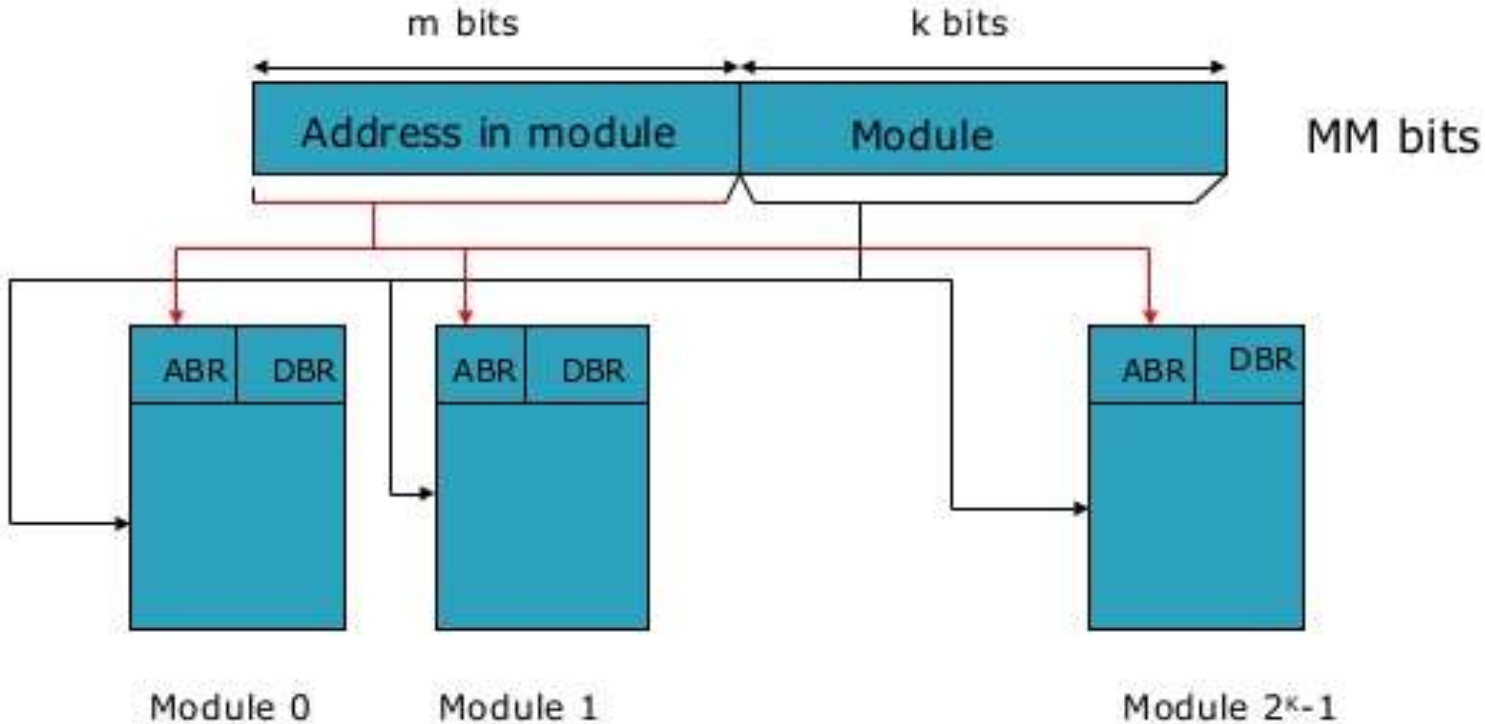
- The higher order k bits of an n-bit MM address select the module and (n-k) bits i.e m-bits select address in module



2. *Low order interleaving*

- The *least significant bits (LSB) select the memory bank* (module) in low-order interleaving.
- In this case, *consecutive memory addresses are in different memory modules*.
- Allows memory access faster than the cycle time.

- The lower order k bits of an n-bit MM address select the module and higher order m-bits select address in module



Consecutive words in a module

Advantages of Memory Interleaving

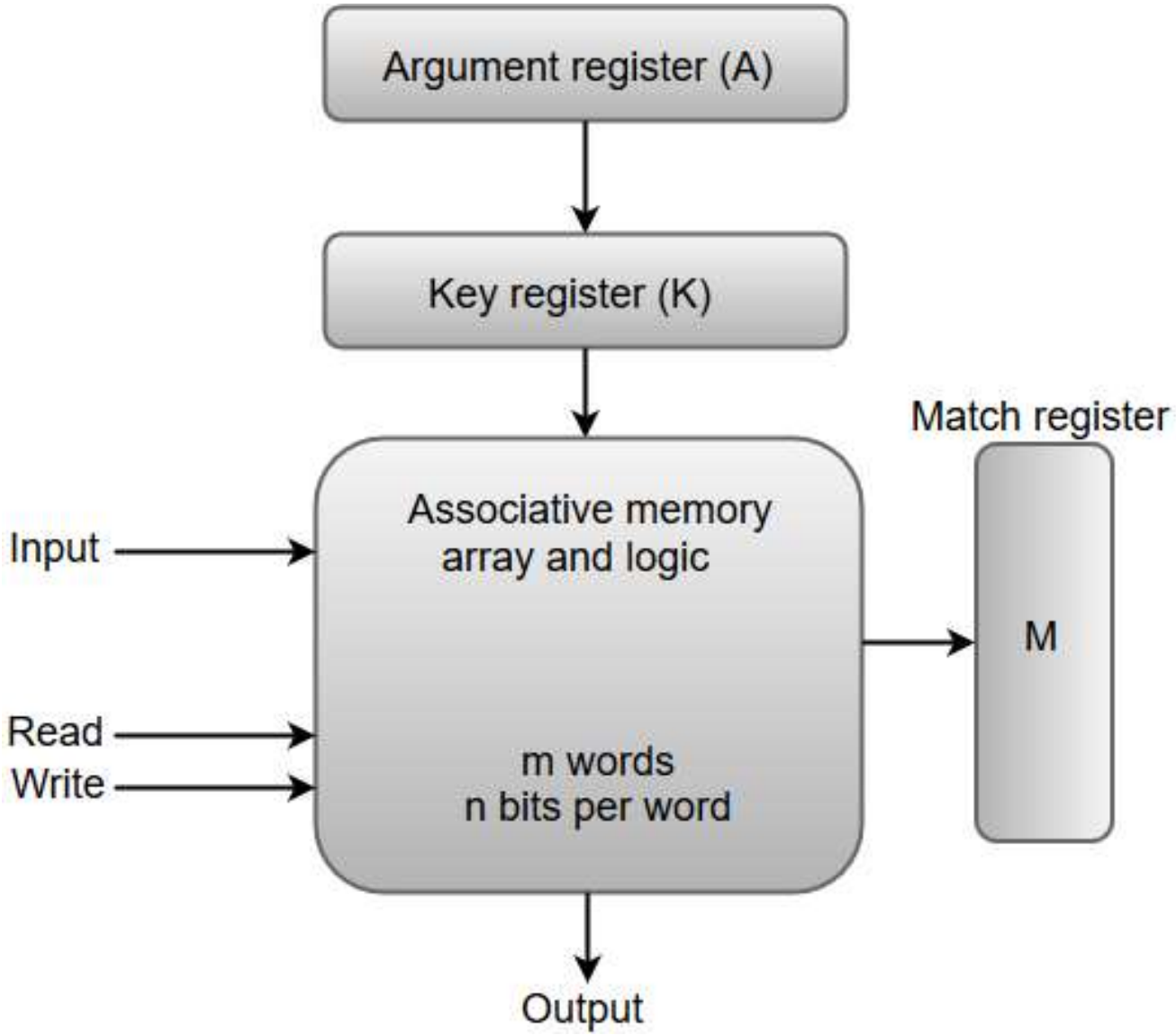
- An *instruction pipeline may require instruction and operands both at the same time* from main memory, which is not possible in the traditional method of memory access. Similarly, an *arithmetic pipeline requires two operands to be fetched simultaneously from the main memory*. So, to overcome this problem, memory interleaving techniques are used as they provide the following advantages.

1. It *allows simultaneous access to different modules of memory*. The modular memory technique *allows the CPU to initiate memory access with one module while others are busy with the CPU in reading or write operations*.
2. Interleaved memory *makes a system more responsive and faster compared to non-interleaving*.
3. Due to simultaneous memory access, the *CPU processing time also decreases and increasing throughput*.

Associative Memory

- An associative memory can be considered as a *memory unit whose stored data can be identified for access by the content of the data itself rather than by an address or memory location.*
- Associative memory is often referred to as ***Content Addressable Memory (CAM)***.
- When a write operation is performed on associative memory, *no address or memory location is given* to the word. The *memory itself is capable of finding an empty unused location* to store the word.
- On the other hand, when the word is to be read from an associative memory, the *content of the word, or part of the word, is specified*. The *words which match the specified content are located by the memory and are marked for reading*.

Block Representation Of An Associative Memory.

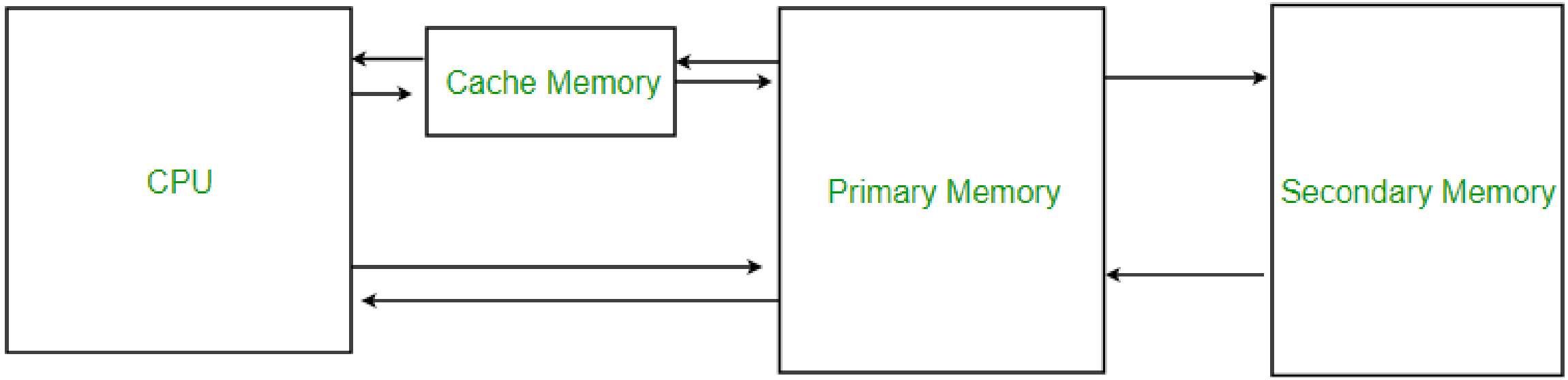


- Fig shows the block diagram of associative memory of m locations of n bits each.
- To perform a search operation, the argument is given to the argument register .
- Two types of searches are possible
  - Searching on the entire argument
  - Searching on a part within the argument
- The key register is supplied a mask pattern that indicates the bits of argument to be included for search pattern.
- If any bit is 0 in the mask pattern, then the corresponding bits of the argument register should not be included in the search.

- For an entire argument search, the key register is supplied all 1's
- While searching, the associative memory locates all the words which match the given argument and marks them by setting the corresponding bits in the match register
- Example
- A = 10111100
- K = 11100000
- Word1 = 10011100 no match
- Word 2 = 101100001 match

Cache Memory

Cache memory is an *extremely fast memory* type that *acts as a buffer between RAM and the CPU.*



- **Cache Memory** is a special *very high-speed memory*.
- It is *used to speed up and synchronize with high-speed CPU*.
- Cache memory is *costlier than main memory or disk memory but economical than CPU registers.*
- It *holds frequently requested data and instructions* so that they are immediately available to the CPU when needed.
- It *stores copies of the data from frequently used main memory locations.*
- Cache memory is used to *reduce the average time to access data* from the Main memory.
- There are *various different independent caches in a CPU*, which store instructions and data.

- Levels of memory:

1. ***Level 1 or Register*** –
   - It is a type of memory in which data is stored and accepted that are *immediately stored in CPU*. Most commonly used register is *accumulator, Program counter, address register etc*.
2. ***Level 2 or Cache memory*** –
   - It is the fastest memory which has faster access time where *data is temporarily stored for faster access*.
3. ***Level 3 or Main Memory*** –
   - It is *memory on which computer works currently*. It is small in size and once power is off data no longer stays in this memory.
4. ***Level 4 or Secondary Memory*** –
   - It is *external memory* which is not as fast as main memory but *data stays permanently in this memory.*

The basic operation of a cache memory is as follows:

1. When the processor needs to read or write a location in main memory, it *first checks for a corresponding entry in the cache*.
2. *If the processor finds that the memory location is in the cache*, a ***cache hit*** has occurred and *data is read from cache*
3. If the *processor does not find the memory location in the cache, a* ***cache miss*** *has occurred*. For a cache miss, the *cache allocates a new entry and copies in data from main memory,* and then the request is fulfilled from the contents of the cache.
4. The *performance of cache memory is frequently measured in terms of a quantity called* ***Hit ratio***.

Hit ratio = no. of hits / total memory accesses made

i.e Hit ratio = No. of hits / (hit + miss)

Types of Cache :

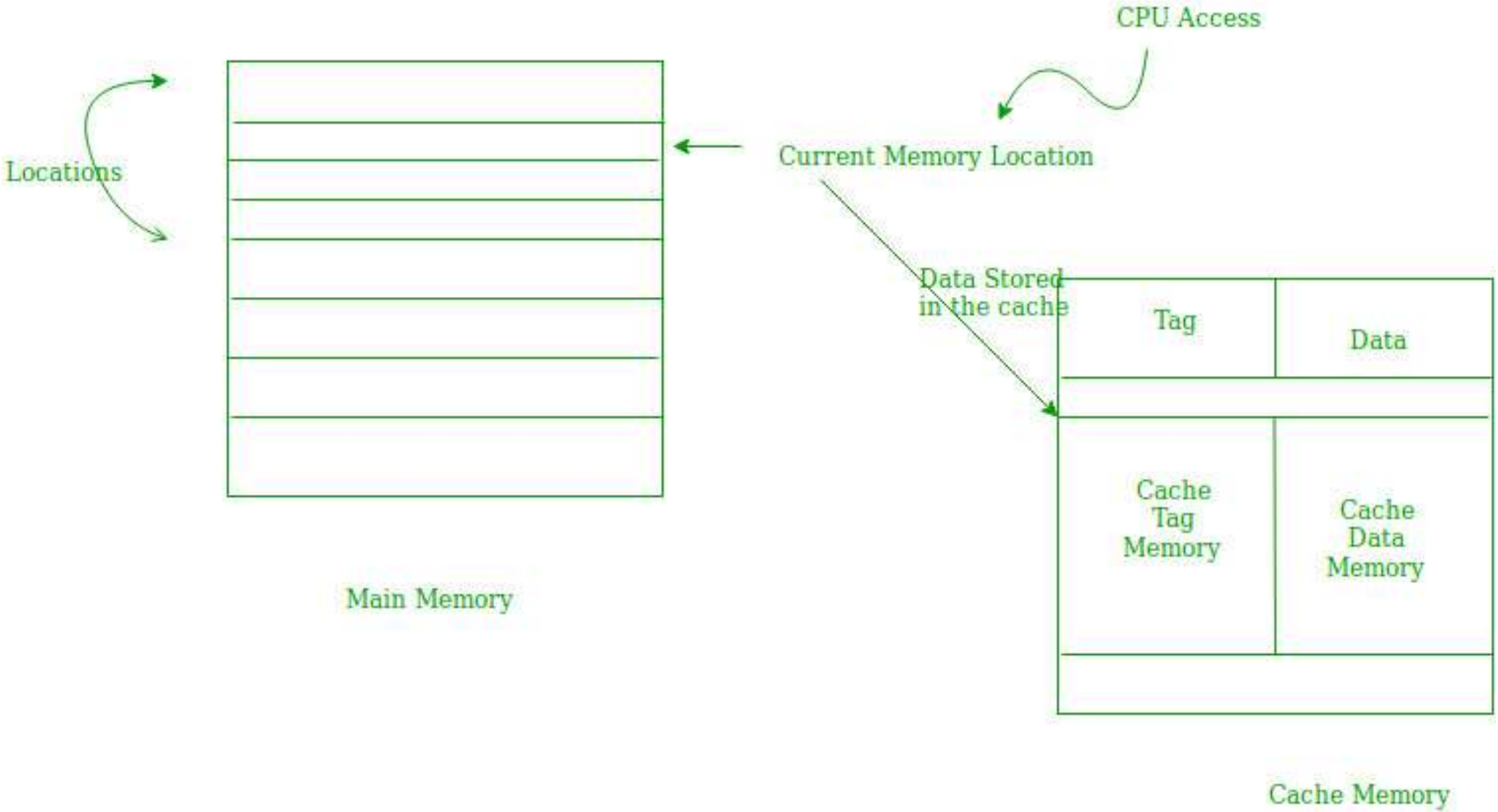
1. **Primary Cache –**
   - A primary cache is always located on the processor chip.
   - This cache is small and its access time is comparable to that of processor registers.

2. **Secondary Cache –**
   - Secondary cache is placed between the primary cache and the rest of the memory.
   - It is referred to as the level 2 (L2) cache.
   - Often, the Level 2 cache is also housed on the processor chip.

Locality of Reference

- Since size of cache memory is less as compared to main memory, so checking which part of main memory should be given priority and loaded in cache is decided based on *LOCALITY OF REFERENCE*.

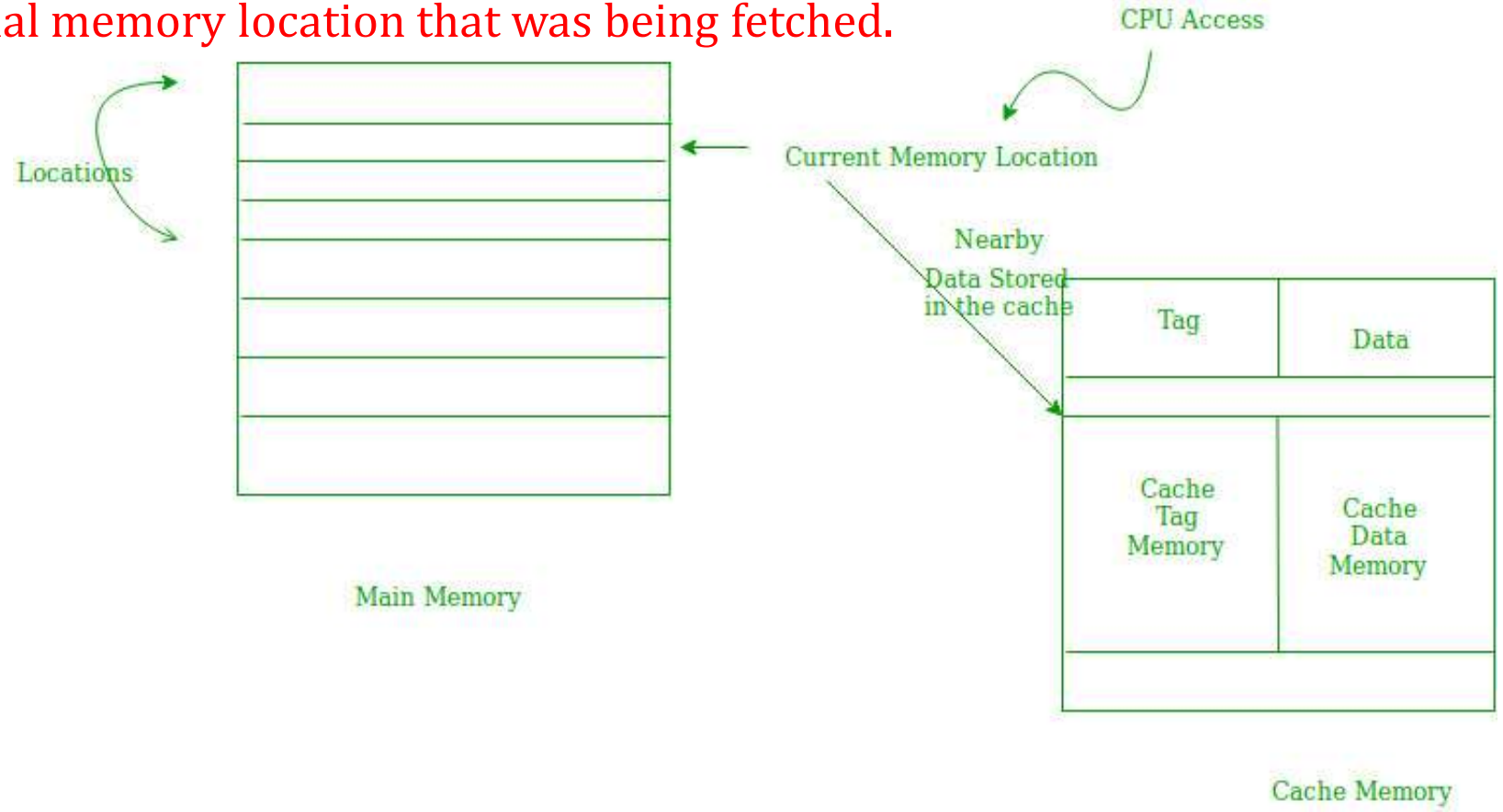
- **Types of Locality of reference**
  1. **Temporal Locality of reference**
     - This concept says that since programs use loops and recursions, they may tend to use the most recently used data/instructions again and again.
     - Temporal locality means current data or instruction that is being fetched may be needed soon. So we should store that data or instruction in the cache memory so that we can avoid again searching in main memory for the same data.
     - When CPU accesses the current main memory location for reading the required data or instruction, it also gets stored in the cache memory which is based on the fact that same data or instruction may be needed in near future. i.e. If some data is referenced, then there is a high probability that it will be referenced again in the near future.



Temporal Locality of reference

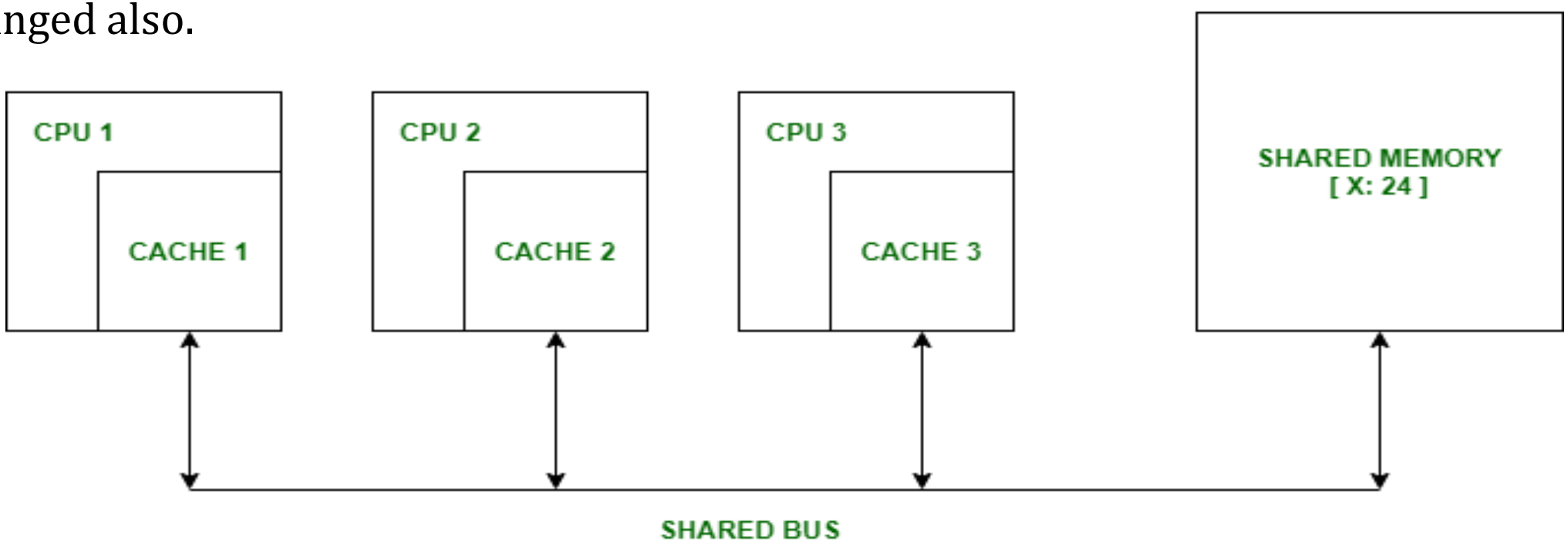
2. Spatial Locality of reference:

- It works on the concept that programs and data currently being accessed by the processor mostly reside in consecutive memory locations.
- Thus, Spatial locality means instruction or data near to the current memory location that is being fetched, may be needed soon in the near future.
- This is slightly different from the temporal locality as in Spatial we are talking about nearly located memory locations while in temporal locality we were talking about the actual memory location that was being fetched.



Cache Coherence

- In a multiprocessor system, *data inconsistency may occur* among adjacent levels or within the same level of the memory hierarchy.
- In a shared memory multiprocessor with a separate cache memory for each processor, *it is possible to have many copies of any one instruction operand* i.e. *one copy in the main memory and one in each cache memory.*
- When one copy of an operand is changed, the other copies of the operand must be changed also.



- Suppose there are three processors, each having cache. Consider the following scenario:-

1. Processor 1 *read* X : *obtains 24* from the memory and caches it.
2. Processor 2 *read* X : *obtains 24* from memory and caches it.
3. Again, *processor 1 writes as X* : 64, Its locally cached copy is updated.
4. Now, processor 3 *reads* X, what value should it get?
5. Memory and processor 2 thinks it is 24 and processor 1 thinks it is 64.

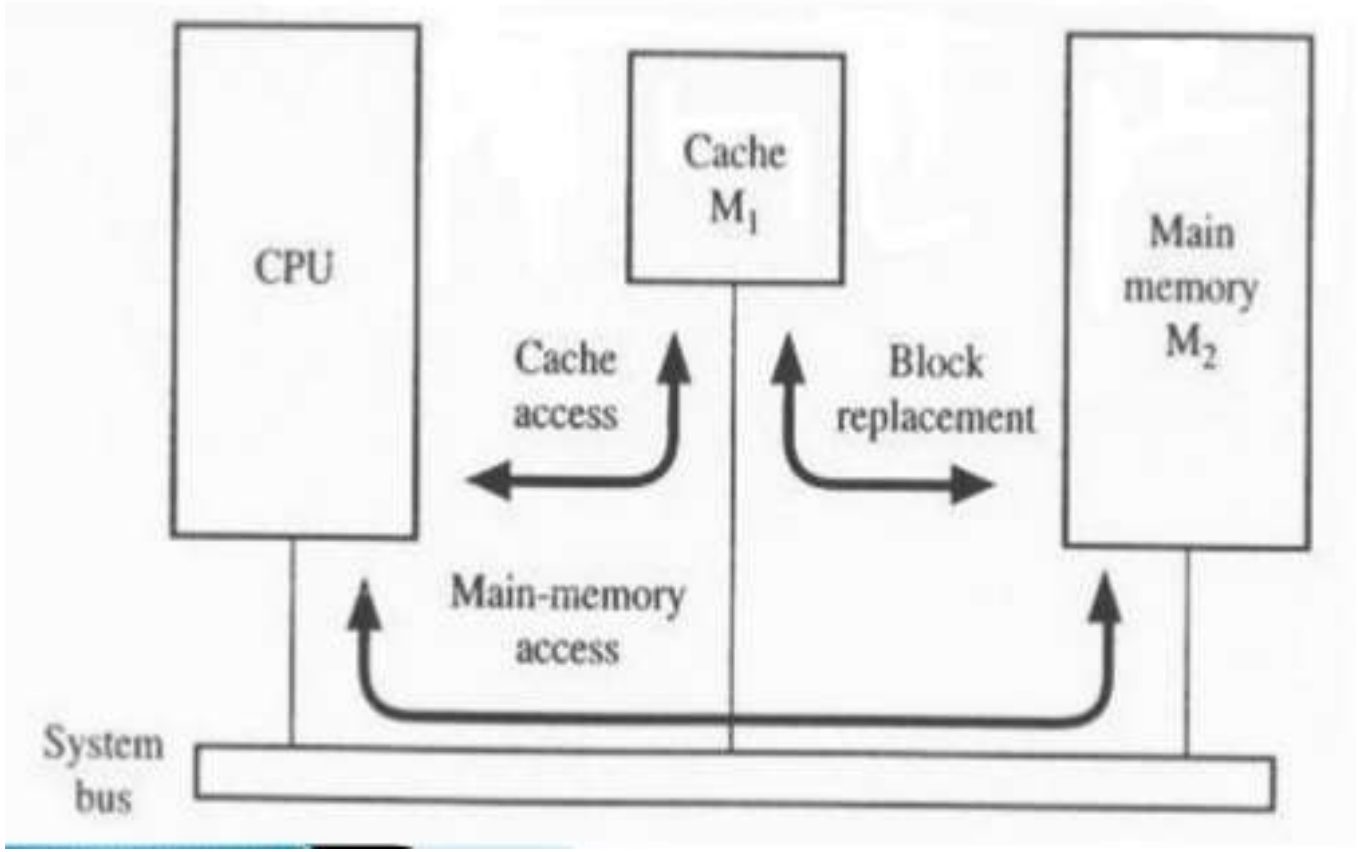
- As multiple processors operate in parallel, and *independently multiple caches may possess different copies of the same memory block*, this creates a cache coherence problem.
- ***Cache coherence is the discipline that ensures that changes in the values of shared operands are propagated throughout the system in a timely fashion.***

Cache Organization

2 ways of introducing cache into a computer :

1. Look aside cache design
2. Look through cache design

1. Look aside Cache design



- CPU initiates a read operation by placing the address on the address bus.
- Cache immediately checks to see if the data is a cache hit.

If HIT

Cache responds to the read operation.

MM is not involved.

If MISS

1. MM responds to the processor and the required word is transferred to the CPU.
2. Block is transferred to the cache.

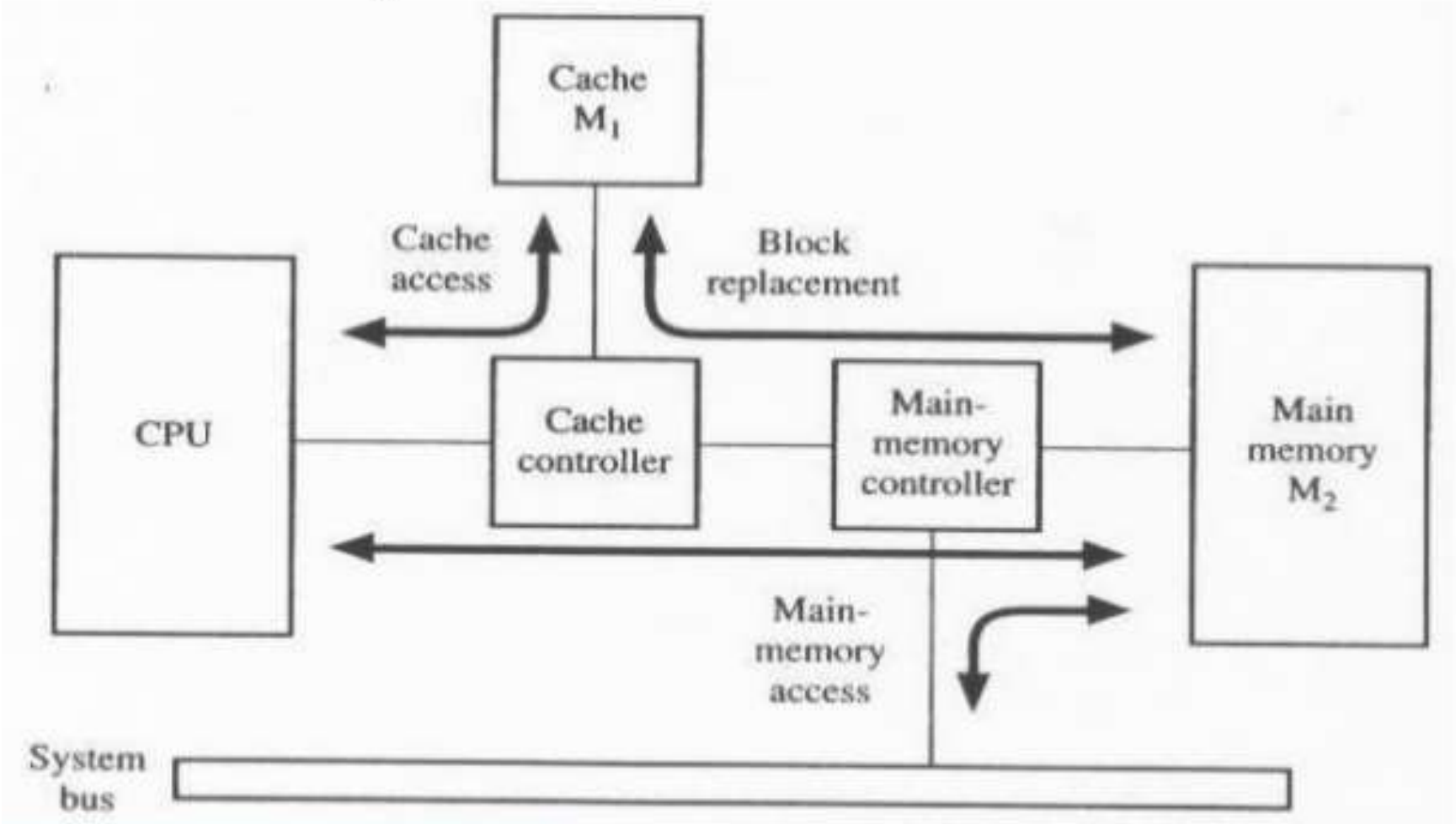
Adv:

Provides better response since data is directly transferred from MM to the CPU

Disadv:

Processor cannot access the cache while another device is accessing the MM (since there is a common bus that connects CPU, MM and other devices in the system)

2. Look Through Cache design



- CPU communicates with the cache via a local bus isolated from the system bus.
- The CPU initiates a read operation by checking if data is in cache.

If HIT

Cache responds to the read operation.

MM is not involved.

If MISS

1. Block is first read into cache.
2. Word is transferred from cache to the processor.

Cache Mapping

- Mapping functions and replacement algo together decides where a line from the main memory can reside in the cache.

- The different mapping techniques used for the same are as follows:
  1. ***Direct mapping.***
  2. ***Fully Associative mapping***
  3. ***Set Associative mapping***

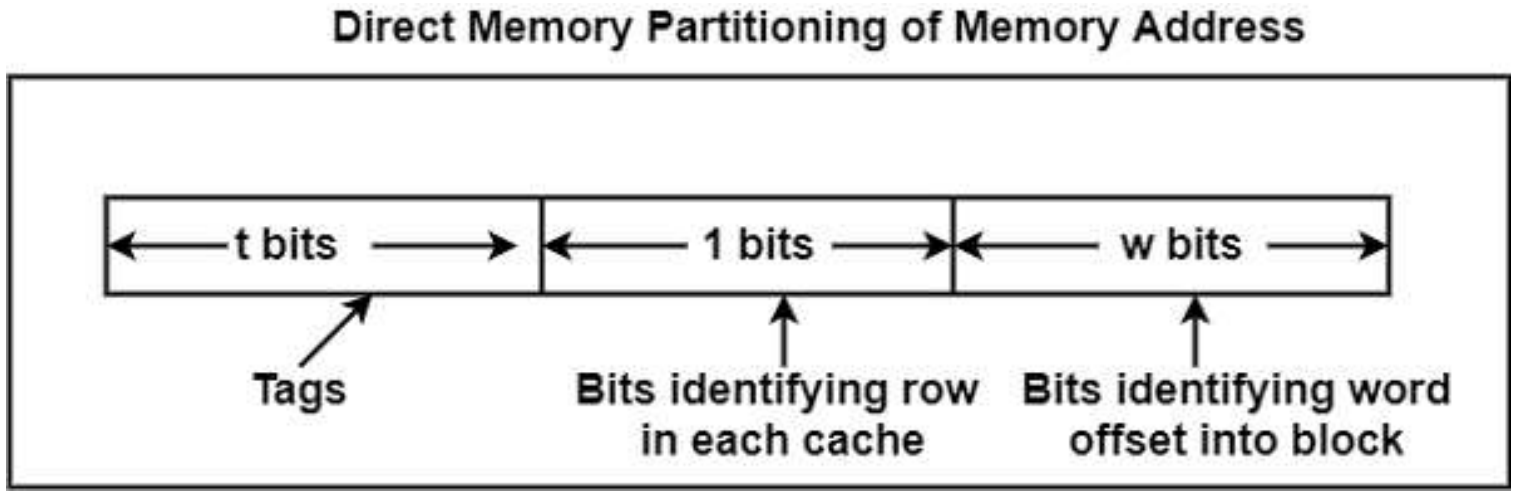
- Direct mapping *maps each block of main memory into only one possible cache line*, i.e. In Direct mapping, *assign each memory block to a specific line in the cache*.
- If a line is previously taken up by a memory block when a new block needs to be loaded, *the old block is trashed*.
- An *address space is split into two parts **index field** and a **tag field***.
- The *cache is used to store the tag field* whereas the *rest is stored in the main memory.*
- Direct mapping`s *performance is directly proportional to the Hit ratio*.

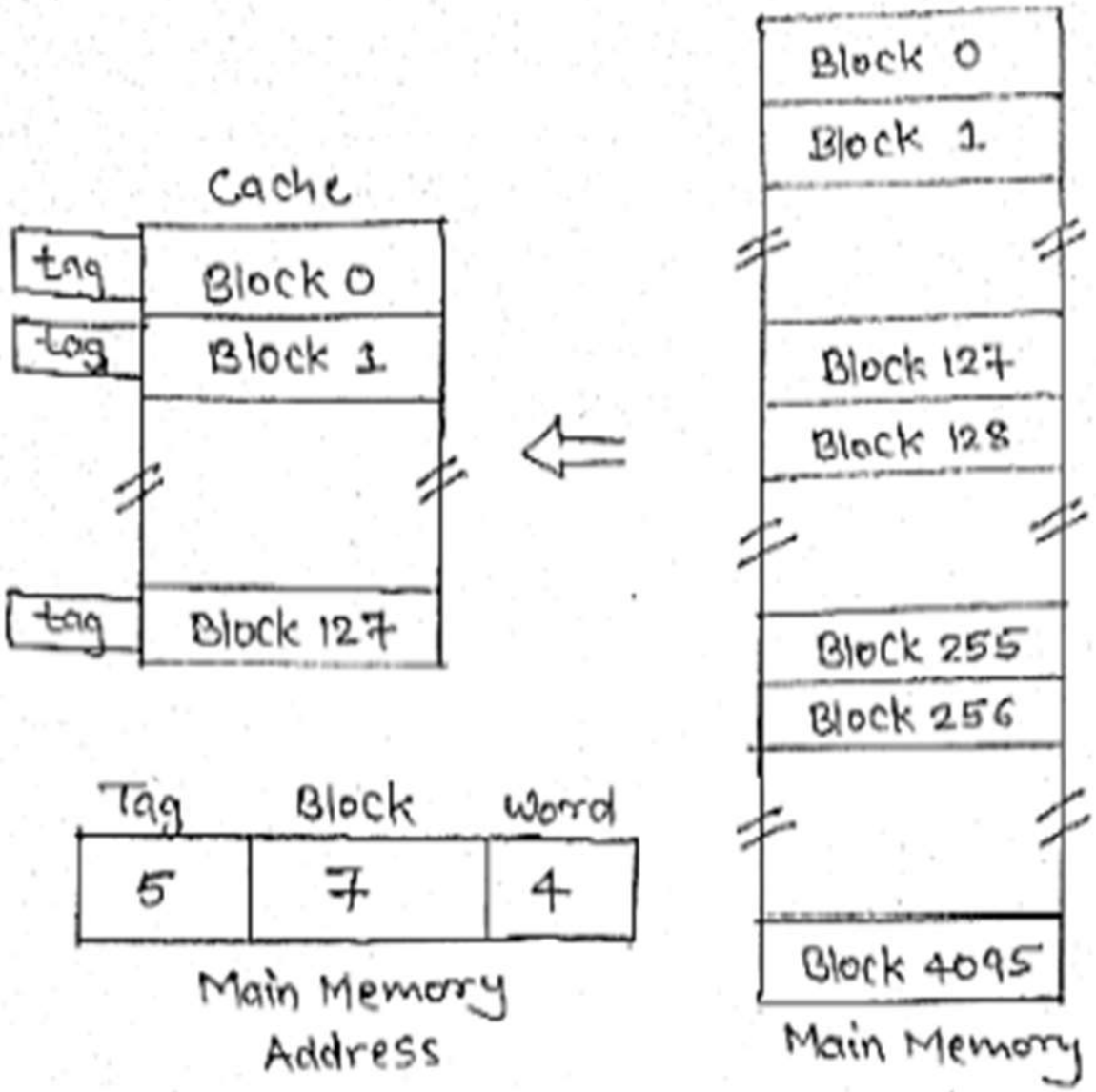
$$i = j \text{ modulo } m$$

where
i=cache line number
j= main memory block number
m=number of lines in the cache

- In this *block J of the main memory maps on to block J modulo 128 of the cache*. Thus main memory blocks 0,128,256,….is loaded into cache is stored at block 0. Block 1,129,257,….are stored at block 1 and so on.
- *Placement of a block in the cache is determined from memory address*. Memory address is divided into 3 fields,

1. The *word bits are the least significant bits* that *identify the specific word within a block of memory*.
2. The *line bits* are the next least significant bits that *identify the line of the cache in which the block is stored*.
3. The remaining bits are stored along with the block as the *tag* which *locates the block's position in the main memory*.
4. It is easy to implement, but not Flexible





Main Memory

Cache Memory

Example of Direct mapping

- Computer has a 32 bit MM add. Space

Cache size= 8KB

Block size =16 bytes

Since each block is of 16 bytes hence 4 bits in the WORD field.

28 bits for TAG and LINE.

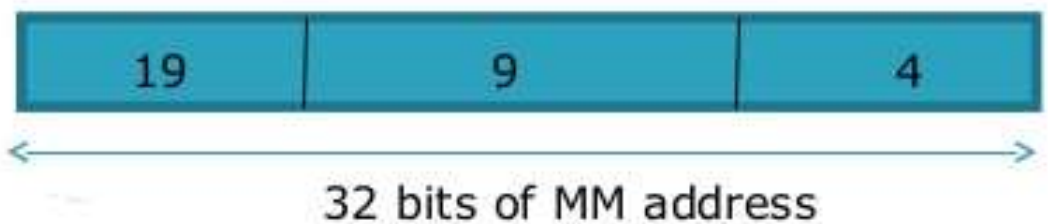
Cache size is 8KB

No. of cache lines = 8KB/16= 512 lines

$512 = 2^9$

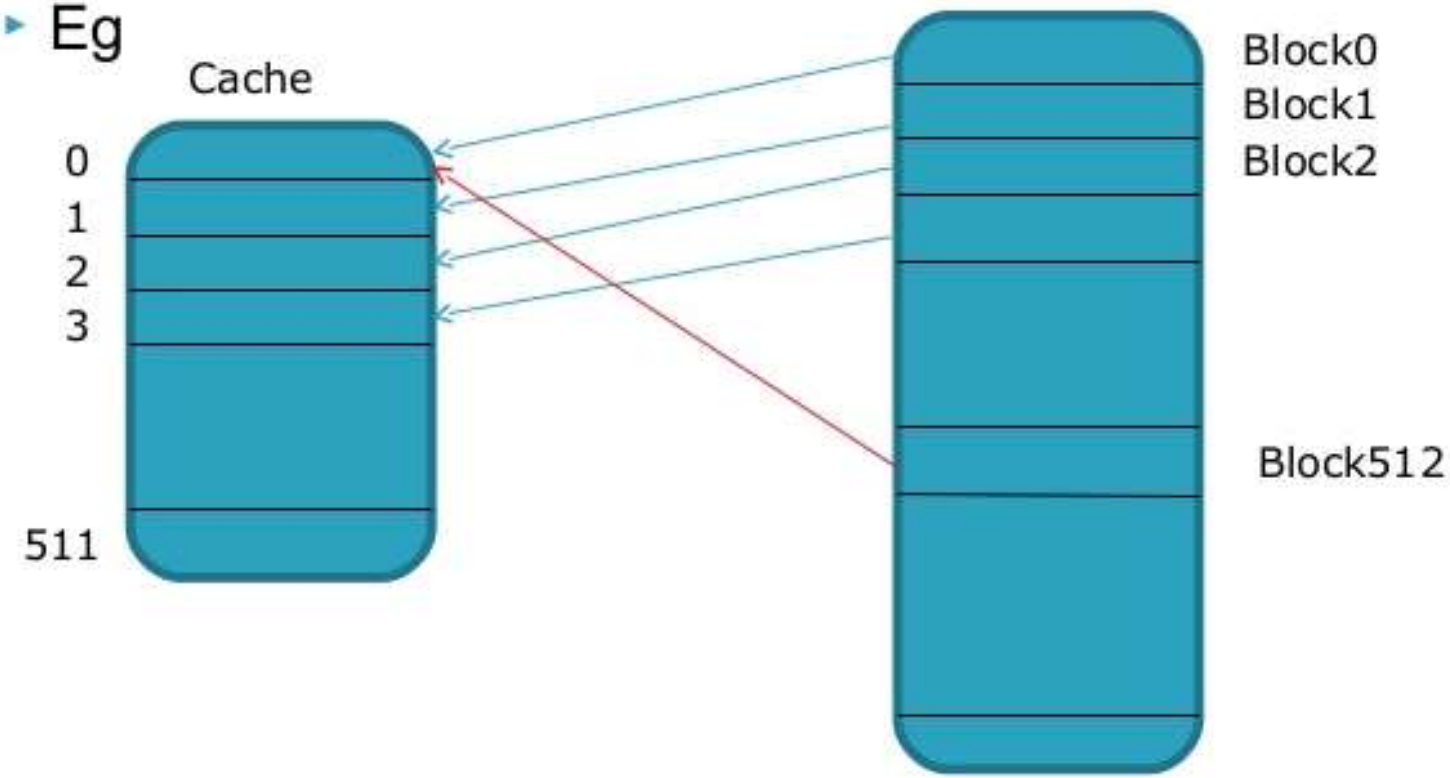
Hence 9 bits for the LINE field.

Hence TAG = 19 bits



Example of Direct mapping

- Eg



2. Associative mapping

- In this type of mapping, the *associative memory is used to store content and addresses of the memory word*.
- *Any block can go into any line of the cache*.
- This means that the *word id bits are used to identify which word in the block is needed*, but the *tag becomes all of the remaining bits*.
- This enables the placement of any word at any place in the cache memory.
- It is considered to be the *fastest and the most flexible mapping form*.

- Best and most expensive.
- A block of MM can be mapped to any cache line.
- MM address consists of 2 fields

- When the CPU needs a word, the cache controller has to match the TAG field in the address with the TAG contents of all the lines in cache.

Adv:

Offers flexibility since a block of MM can be moved to any cache line. Hence replacement is needed only if cache is totally full.

- No. of bits in the WORD field :

No. of bits in Word field depends on the block size.

- No. of bits in the TAG field :

Total no. of bits in MM address – No. of bits in WORD field

- Computer has a 32 bit MM add. Space

Cache size= 8KB

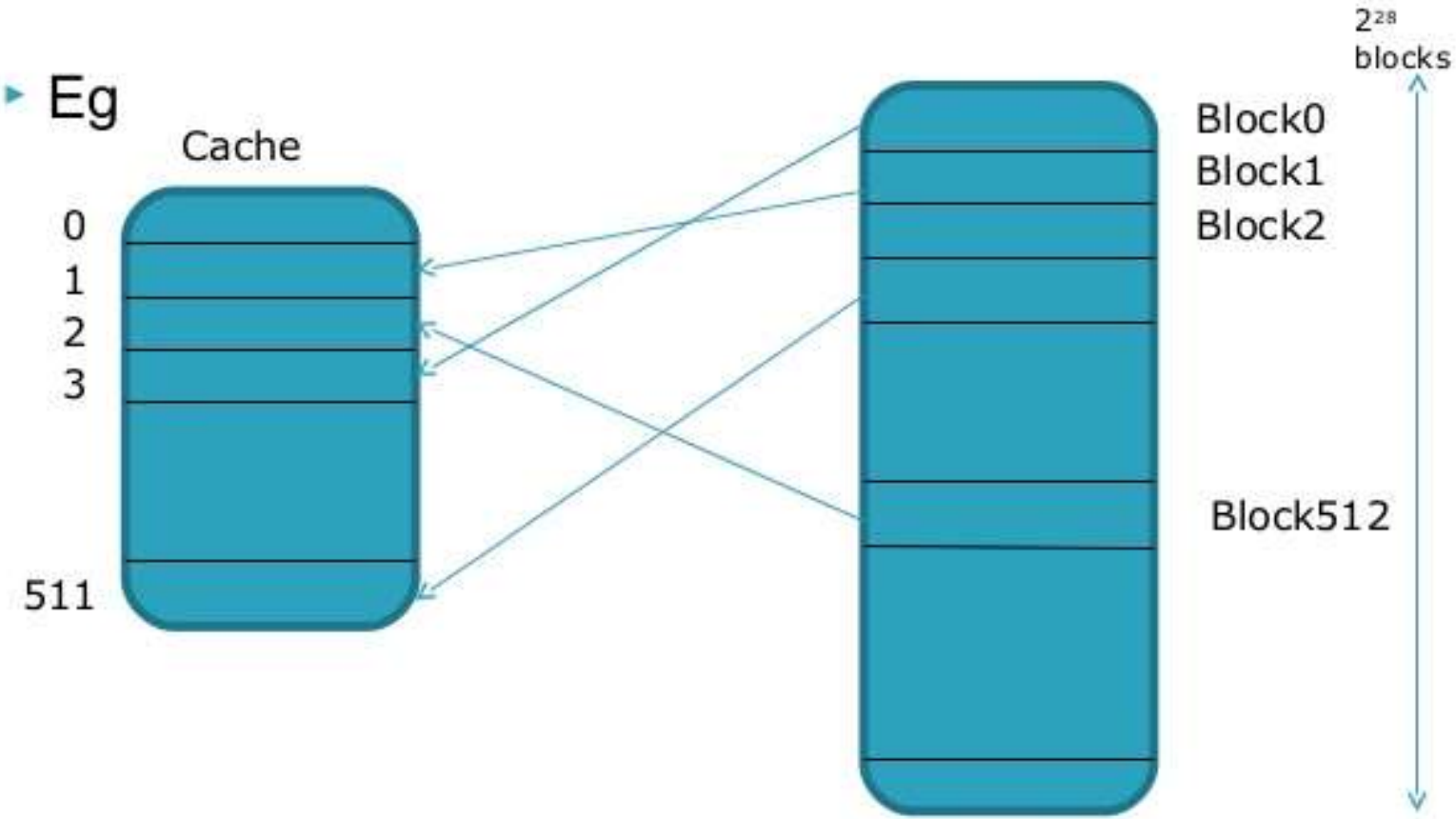
Block size =16 bytes

Since each block is of 16 bytes hence 4 bits in the WORD field.

Remaining 28 bits for the TAG

Example of Associatve mapping

- Eg



3. Set- Associative mapping

- This form of mapping is an *enhanced form of direct mapping* where the drawbacks of direct mapping are removed.
- Set associative *addresses the problem of possible thrashing* in the direct mapping method.
- It does this by saying that instead of having exactly one line that a block can map to in the cache, *we will group a few lines together creating a* ***set****.*
- Then ***a block in memory can map to any one of the lines of a specific set.***

- Set-associative mapping allows that *each word that is present in the cache can have two or more words in the main memory for the same index address*.
- Set associative cache mapping *combines the best of direct and associative cache mapping techniques*.
- In this case, the *cache consists of a number of sets, each of which consists of a number of lines*. The relationships are

$$m = v * k$$
$$i = j \bmod v$$

where
i=cache set number
j=main memory block number
v=number of sets
m=number of lines in the cache number of sets
k=number of lines in each set

- Combination of direct mapping and associative mapping.
- Cache is divided into v sets consisting of k lines each.
- Hence total lines in cache = v x k

Total no. sets — No.of lines in each set

- K lines in each set ---- Called k-way set associative
- MM block can be mapped into a specific set only.
- Which set?

  Block i of MM gets mapped into set --→ i mod v

- Within a set a block can be placed in any line.
- MM address has 3 fields

- No. of bits in the WORD field :

No. of bits in Word field depends on the block size.

- No. of bits in the SET field :

No. of bits in Set field depends on the number of sets.

- No. of bits in the TAG field :

Total no. of bits in MM address – (No. of bits in WORD field +
No. of bits in Set field)

- When the CPU needs to check the cache for accessing a word, the cache controller uses the SET field to access the cache.
- Within the set there are k lines, hence the TAG field in address is matched with TAG contents of all k lines.

Computer has a 32 bit MM add. space

Cache size= 8KB

Block size =16 bytes

4 way set associative.

- No. of cache lines = 8KB/16= 512 lines
- Total no. of sets =512/4 =128 = 2^7

Hence no. of bits in SET = 7

No. of bits in WORD = 4

No. of bits in TAG = 21

Example of Set associative mapping

- Eg