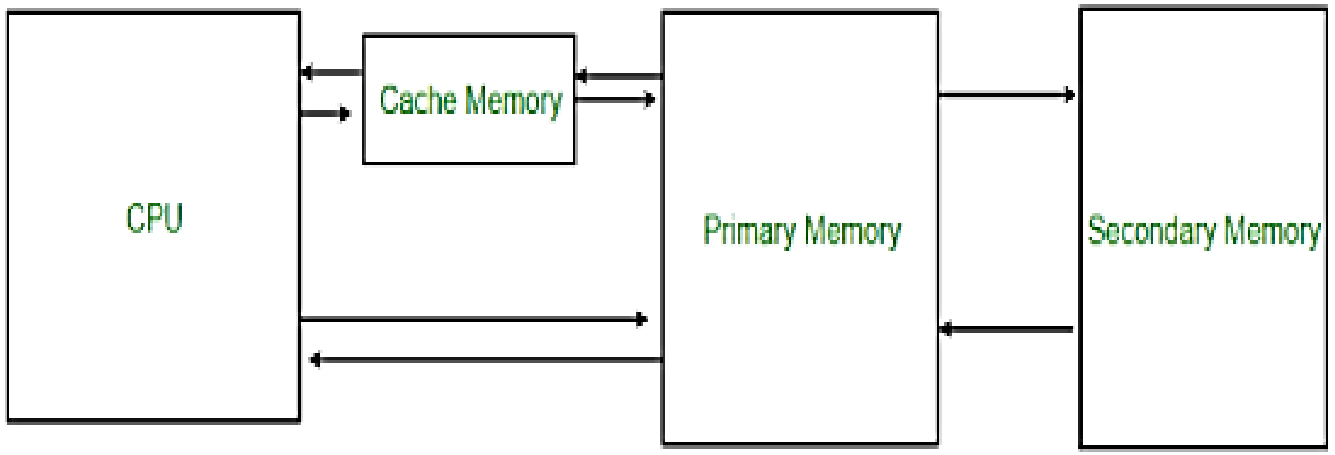
# Cache Memory

**Cache Memory** is a special very high-speed memory. It is used to speed up and synchronizing with high-speed CPU.

Cache memory is an extremely fast memory type that acts as a buffer between RAM and the CPU. It holds frequently requested data and instructions so that they are immediately available to the CPU when needed.

Cache memory is used to reduce the average time to access data from the Main memory. The cache is a smaller and faster memory which stores copies of the data from frequently used main memory locations. There are various different independent caches in a CPU, which store instructions and data.


CPU
Cache Memory
Primary Memory
Secondary Memory

# Levels of memory

**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.

**Level 2 or Cache memory –**
It is the fastest memory which has faster access time where data is temporarily stored for faster access.

**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.

**Level 4 or Secondary Memory –**
It is external memory which is not as fast as main memory but data stays permanently in this memory.

# Cache Performance:

- When the processor needs to read or write a location in main memory, it first checks for a corresponding entry in the cache.
- If the processor finds that the memory location is in the cache, a **cache hit** has occurred and data is read from cache
- 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, then the request is fulfilled from the contents of the cache.

- The performance of cache memory is frequently measured in terms of a quantity called **Hit ratio.**

- Hit ratio = hit / (hit + miss) = no. of hits/total accesses

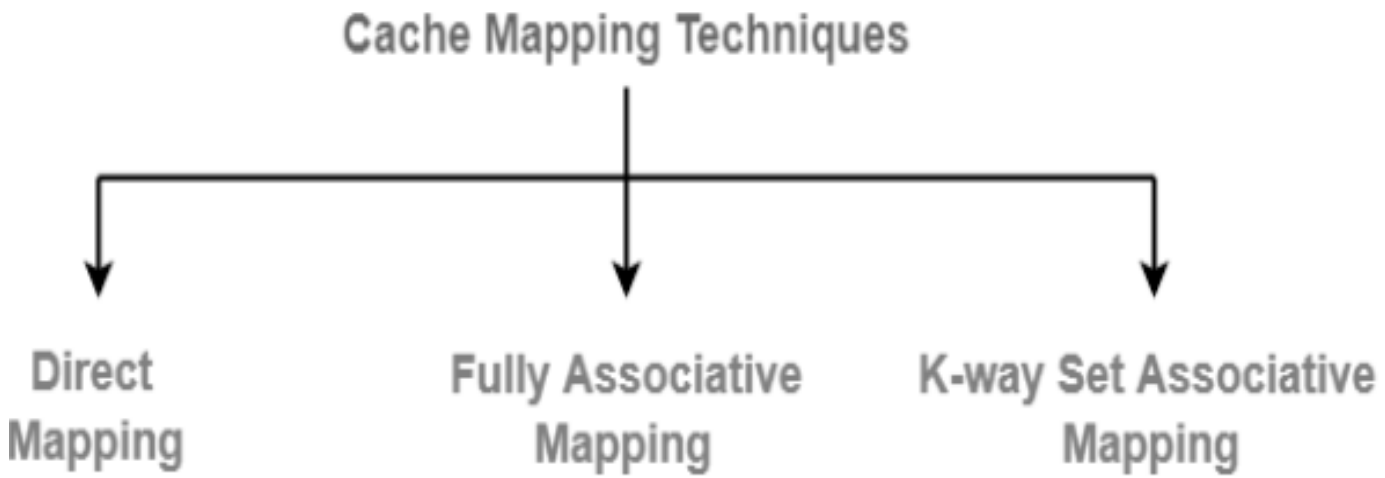
# Cache Mapping:

There are three different types of mapping used for the purpose of cache memory which are as follows:

Direct mapping

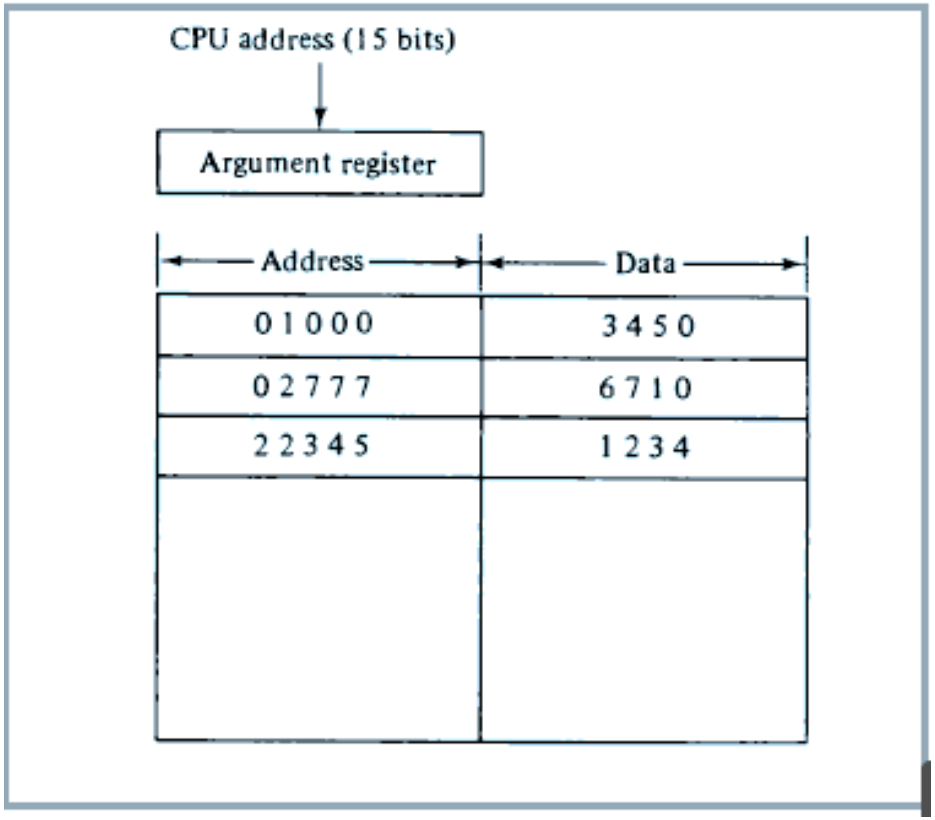
Associative mapping

Set-Associative mapping


Cache Mapping Techniques
Direct Mapping
Fully Associative Mapping
K-way Set Associative Mapping

# Associative Mapping

- The fastest and most flexible cache organization uses an associative memory.
- The associative memory stores both the address and the content of memory word.
- This permits any location in cache to store nay word in the cache to store any word from main memory.




- The address value of 15 bits is shown as a five digit octal number and 12 bit word is representing in four digit octal number.

- A CPU address is placed in argument register and associative memory searched for matching address.

- If the address is found then corresponding data is read.

## 2. Direct Mapping

Associative mapping are expensive compared to random access memories because of the added logic associated with each cell.

The simplest technique, known as direct mapping, maps each block of main memory into only one possible cache line.

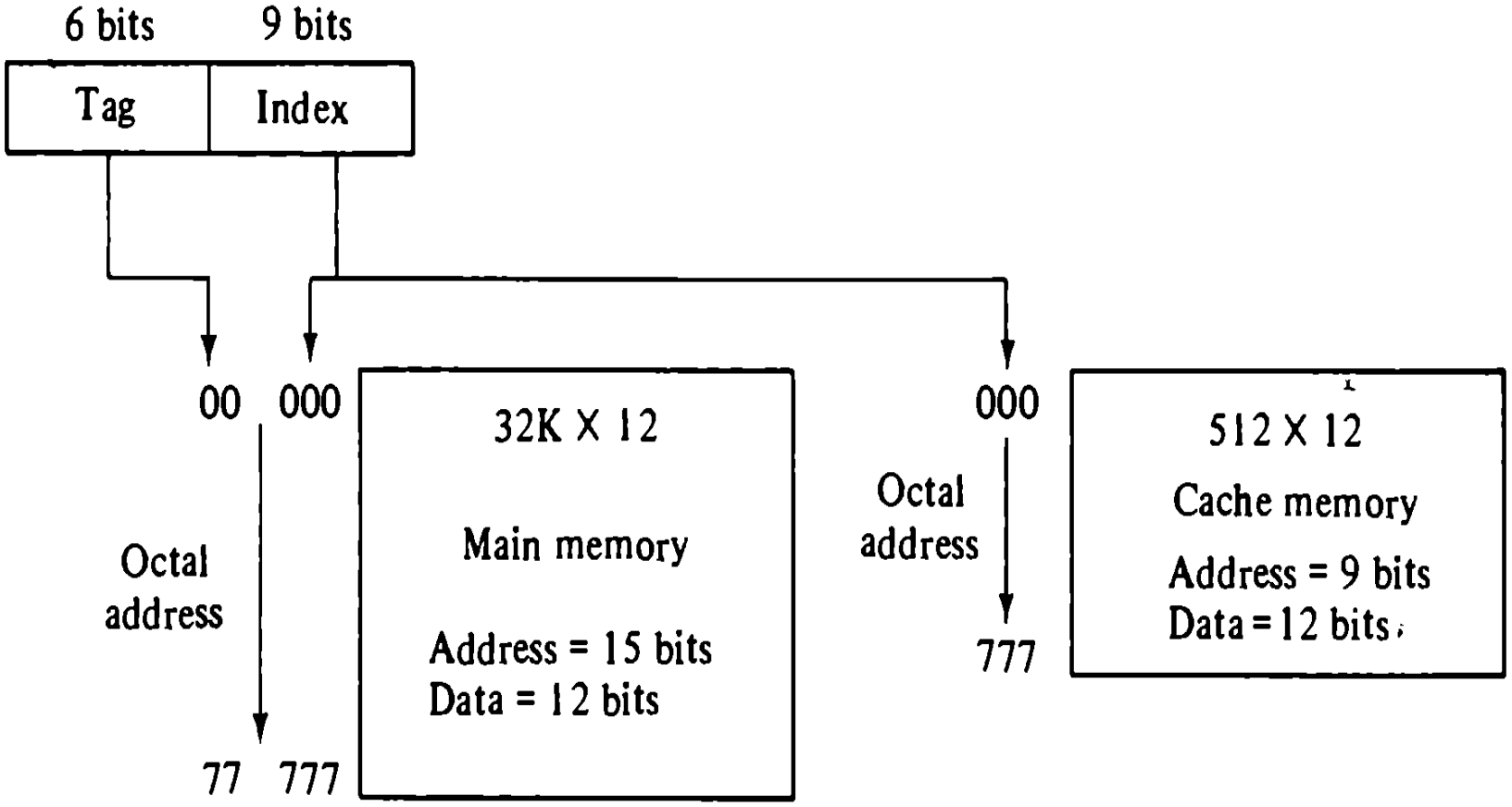
In Direct mapping, assign each memory block to a specific line in the cache. If a line is previously taken up by a memory block when a new block needs to be loaded, the old block is trashed.

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.

**Figure 12-12** Addressing relationships between main and cache memories.



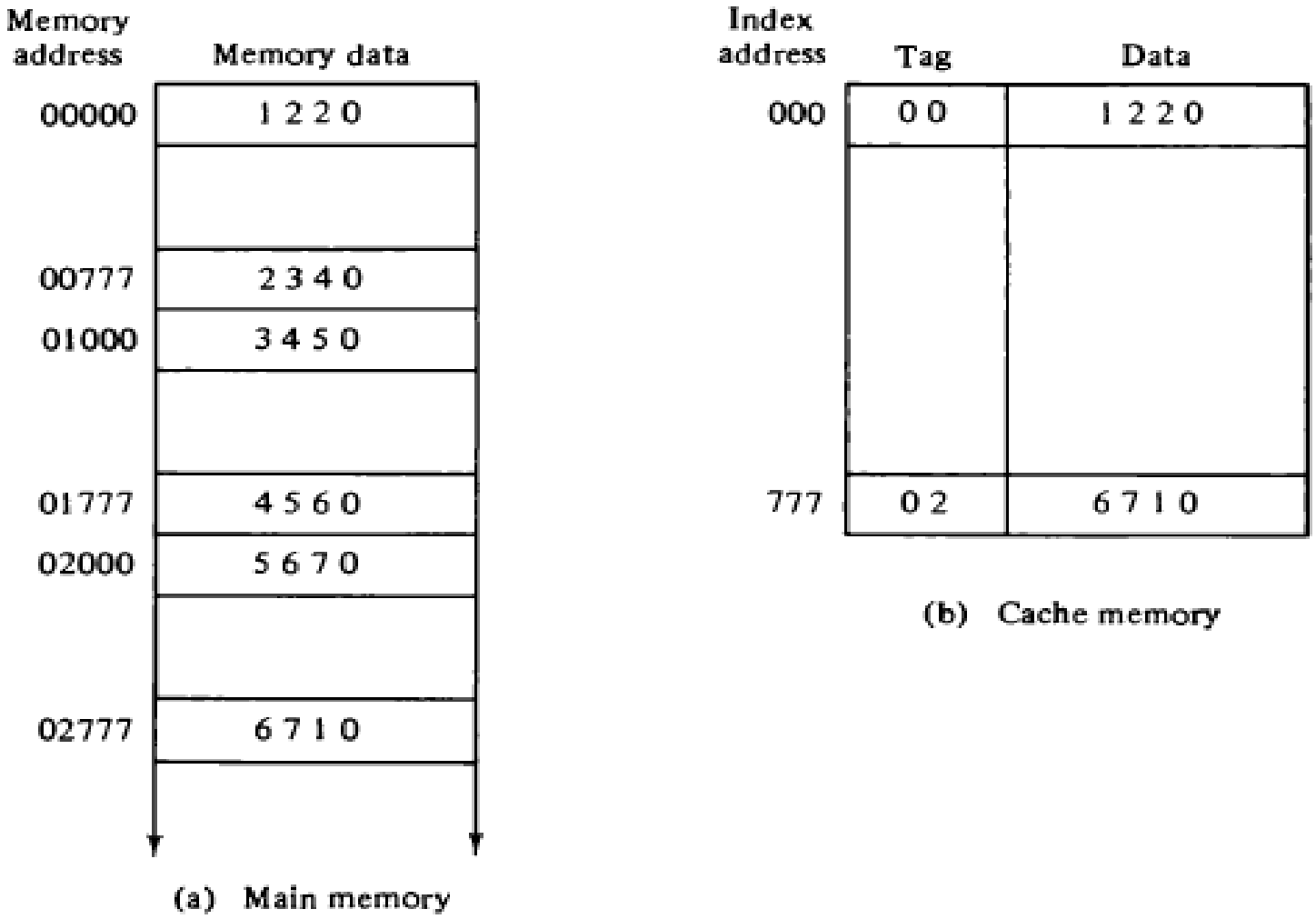



**Figure 12-13** Direct mapping cache organization.

- Hit ration can drop if two or more word whose address have the same index but different tag are accessed repeatedly.

# 3. Set Associative Mapping

Each word of cache can store two or more memory words of memory under the same index address.

| Index | Tag | Data | Tag | Data |
|---|---|---|---|---|
| 000 | 01 | 3450 | 02 | 5670 |
| | | | | |
| 777 | 02 | 6710 | 00 | 2340 |

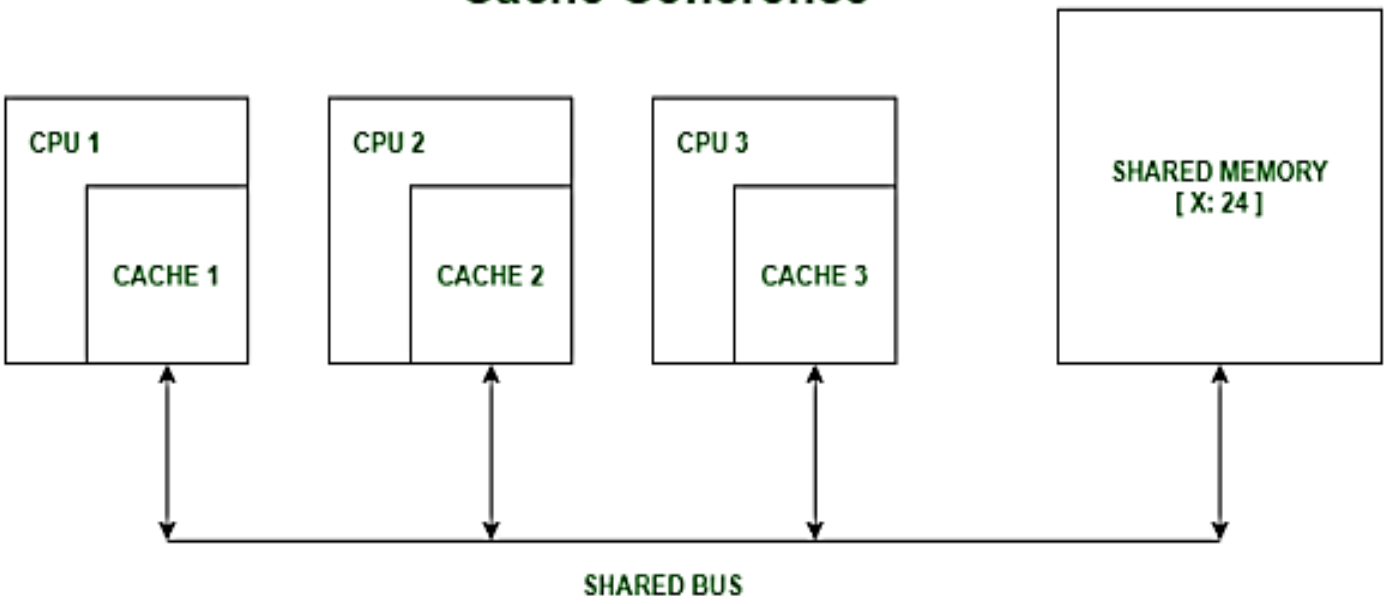
**Figure 12-15** Two-way set-associative mapping cache.

# 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: 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.

## Cache Coherence




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

**Processor 1 read X :** obtains 24 from the memory and caches it.

**Processor 2 read X :** obtains 24 from memory and caches it.

**Again, processor 1 writes as X :** 64, Its locally cached copy is updated. Now, processor 3 reads X, what value should it get?

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.

There are three distinct level of cache coherence :-

- Every write operation appears to occur instantaneously.
- All processors see exactly the same sequence of changes of values for each separate operand.
- Different processors may see an operation and assume different sequences of values; this is known as non-coherent behavior.

# Auxiliary Memory

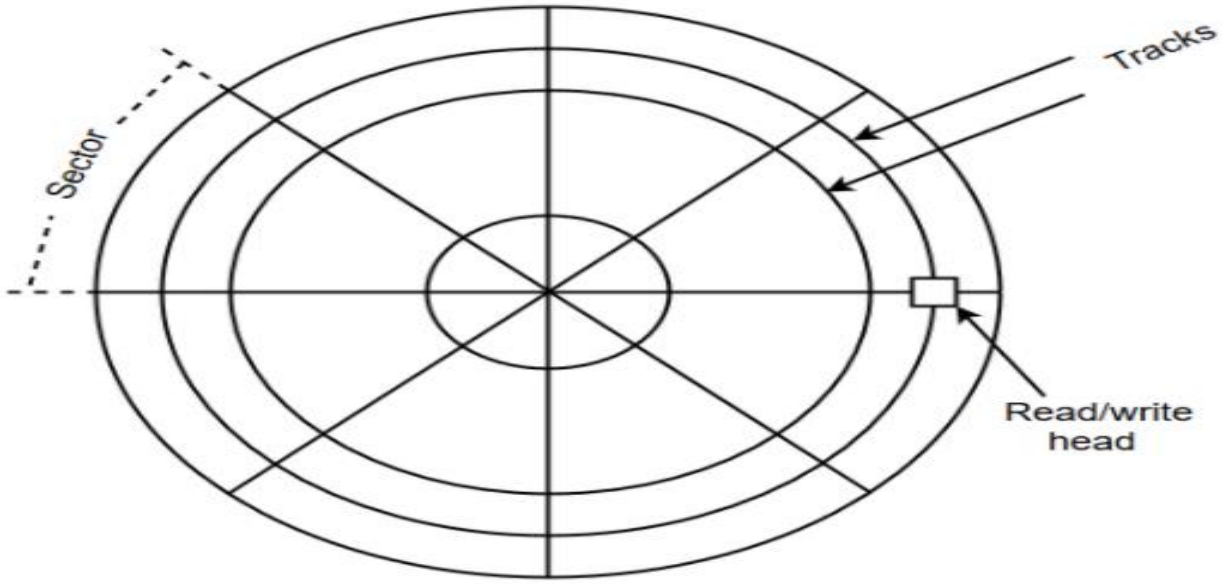
An Auxiliary memory is known as the lowest-cost, highest-capacity and slowest-access storage in a computer system. It is where programs and data are kept for long-term storage or when not in immediate use. The most common examples of auxiliary memories are magnetic tapes and magnetic disks.

# Magnetic Disks

A magnetic disk is a type of memory constructed using a circular plate of metal or plastic coated with magnetized materials.

Usually, both sides of the disks are used to carry out read/write operations.

However, several disks may be stacked on one spindle with read/write head available on each surface




The memory bits are stored in the magnetized surface in spots along the concentric circles called tracks.

The concentric circles (tracks) are commonly divided into sections called sectors.

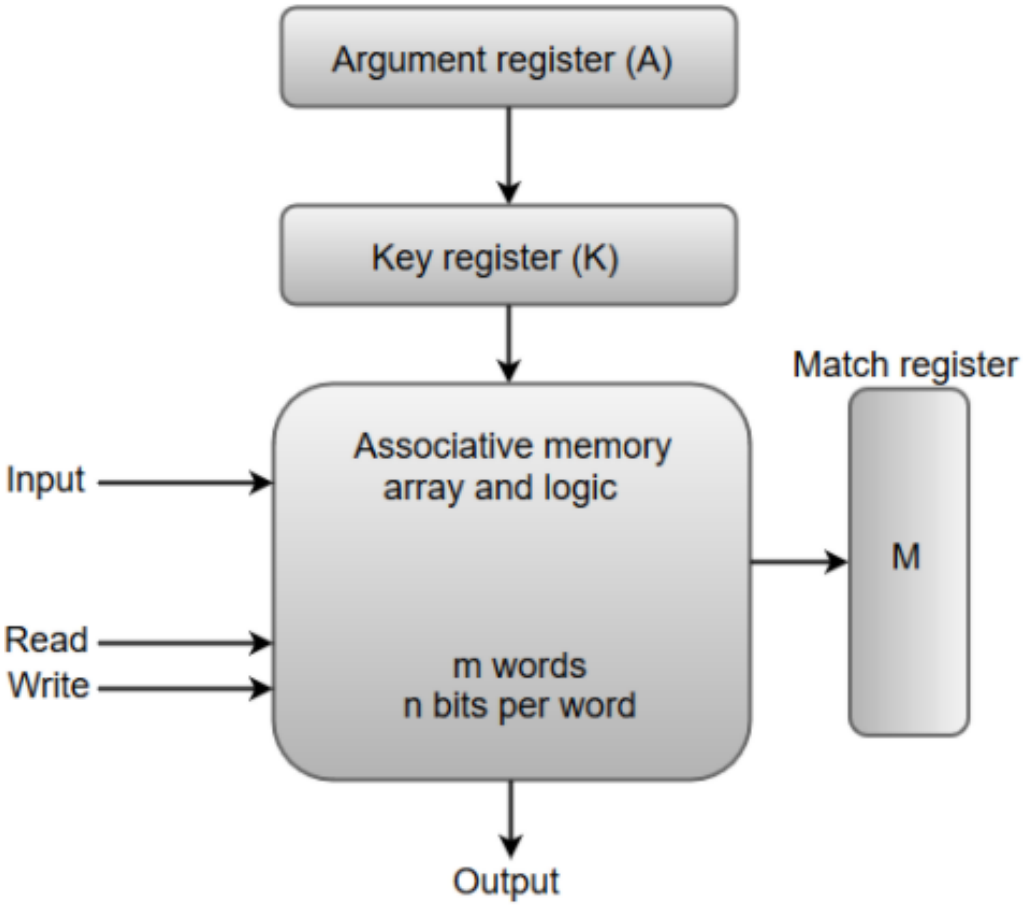
# Associative Memory

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.




From the block diagram, we can say that an associative memory consists of a memory array and logic for 'm' words with 'n' bits per word.

The functional registers like the argument register **A** and key register **K** each have **n** bits, one for each bit of a word. The match register **M** consists of **m** bits, one for each memory word.

The key register (K) provides a mask for choosing a particular field or key in the argument word.

If the key register contains a binary value of all 1's, then the entire argument is compared with each memory word. Otherwise, only those bits in the argument that have 1's in their corresponding position of the key register are compared.

Thus, the key provides a mask for identifying a piece of information which specifies how the reference to memory is made.

| | | |
|---|---|---|
| *A* | 101 111100 | |
| *K* | 111 000000 | |
| Word 1 | 100 111100 | no match |
| Word 2 | 101 000001 | match |