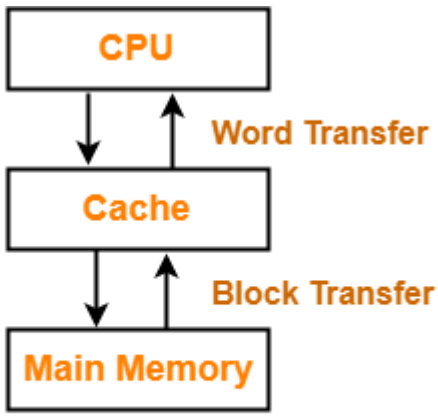
Unit-3

Cache Memory-

- Cache memory is a Random Access Memory.
- The main advantage of cache memory is its very fast speed.
- It can be accessed by the CPU at much faster speed than main memory.

Location-

- Cache memory lies on the path between the CPU and the main memory.
- It facilitates the transfer of data between the processor and the main memory at the speed which matches to the speed of the processor.



Cache and Main Memory

- Data is transferred in the form of words between the cache memory and the CPU.
- Data is transferred in the form of blocks or pages between the cache memory and the main memory.

Purpose-

- The fast speed of the cache memory makes it extremely useful.
- It is used for bridging the speed mismatch between the fastest CPU and the main memory.
- It does not let the CPU performance suffer due to the slower speed of the main memory.

Execution Of Program-

- Whenever any program has to be executed, it is first loaded in the main memory.
- The portion of the program that is mostly probably going to be executed in the near future is kept in the cache memory.
- This allows CPU to access the most probable portion at a faster speed.

Step-01:

Whenever CPU requires any word of memory, it is first searched in the CPU registers.

Now, there are two cases possible-

Case-01:

- If the required word is found in the CPU registers, it is read from there.

Case-02:

- If the required word is not found in the CPU registers, Step-02 is followed.

Step-02:

- When the required word is not found in the CPU registers, it is searched in the cache memory.
- Tag directory of the cache memory is used to search whether the required word is present in the cache memory or not.

Now, there are two cases possible-

Case-01:

- If the required word is found in the cache memory, the word is delivered to the CPU.
- This is known as **Cache hit.**

Case-02:

- If the required word is not found in the cache memory, Step-03 is followed.

- This is known as **Cache miss.**

Step-03:

- When the required word is not found in the cache memory, it is searched in the main memory.
- Page Table is used to determine whether the required page is present in the main memory or not.

Now, there are two cases possible-

Case-01:

If the page containing the required word is found in the main memory,

- The page is mapped from the main memory to the cache memory.
- This mapping is performed using cache mapping techniques.
- Then, the required word is delivered to the CPU.

Case-02:

If the page containing the required word is not found in the main memory,

- A page fault occurs.
- The page containing the required word is mapped from the secondary memory to the main memory.
- Then, the page is mapped from the main memory to the cache memory.
- Then, the required word is delivered to the CPU.

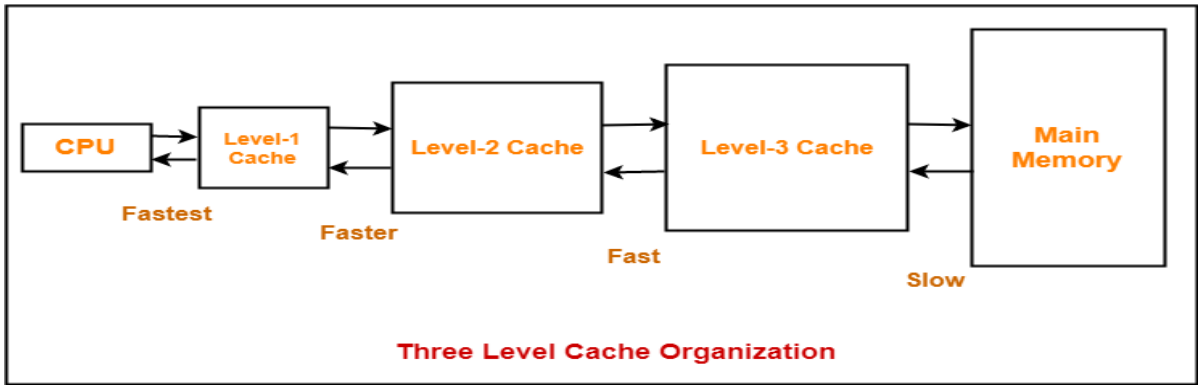
Multilevel Cache Organization-

- A multilevel cache organization is an organization where cache memories of different sizes are organized at multiple levels to increase the processing speed to a greater extent.
- The smaller the size of cache, the faster its speed.
- The smallest size cache memory is placed closest to the CPU.
- This helps to achieve better performance in terms of speed.

Example-

Three level cache organization consists of three cache memories of different size organized at three different levels as shown below-

Size (L1 Cache) < Size (L2 Cache) < Size (L3 Cache) < Size (Main Memory)



Three Level Cache Organization

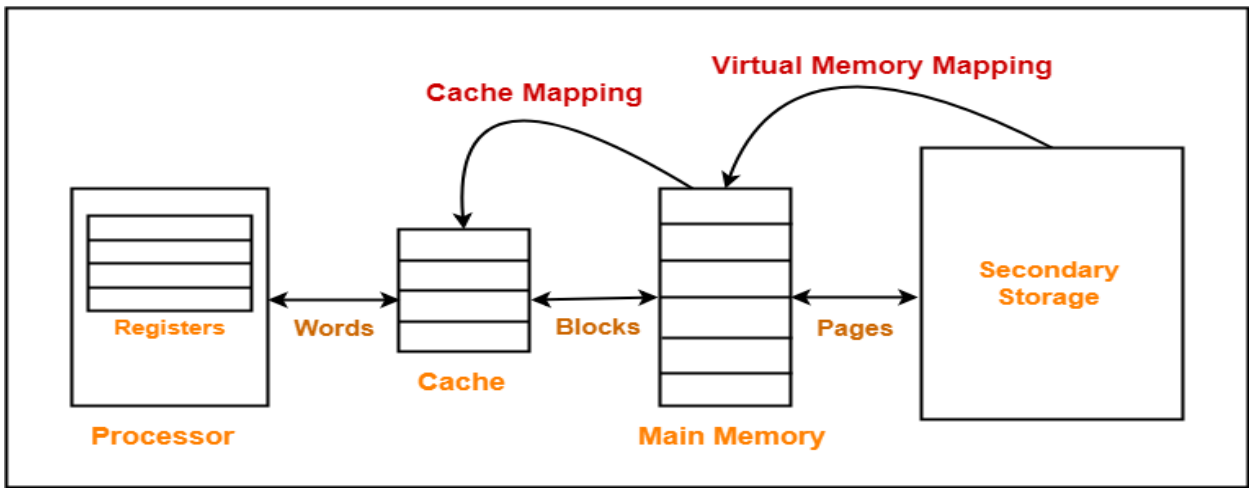
Cache Mapping-

- Cache mapping defines how a block from the main memory is mapped to the cache memory in case of a cache miss.

OR

- Cache mapping is a technique by which the contents of main memory are brought into the cache memory.

The following diagram illustrates the mapping process-



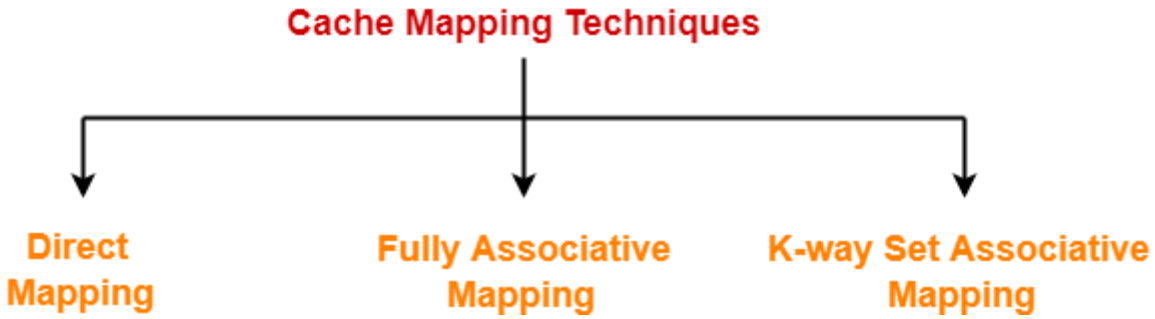
Now, before proceeding further, it is important to note the following points-

NOTES

- Main memory is divided into equal size partitions called as **blocks** or **frames**.
- Cache memory is divided into partitions having same size as that of blocks called as **lines**.
- During cache mapping, block of main memory is simply copied to the cache and the block is not actually brought from the main memory.

Cache Mapping Techniques-

Cache mapping is performed using following three different techniques-



1. Direct Mapping
2. Fully Associative Mapping
3. K-way Set Associative Mapping

1. Direct Mapping-

In direct mapping,

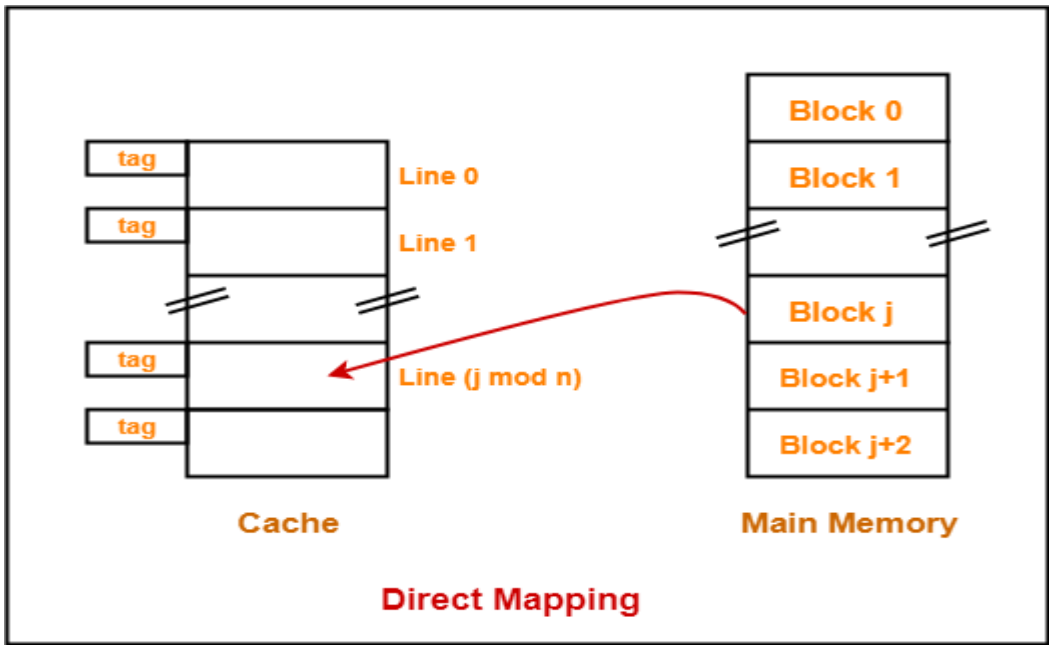
- A particular block of main memory can map only to a particular line of the cache.
- The line number of cache to which a particular block can map is given by-

Cache line number

= (Main Memory Block Address) Modulo (Number of lines in Cache)

Example-

- Consider cache memory is divided into ‘n’ number of lines.
- Then, block ‘j’ of main memory can map to line number (j mod n) only of the cache.



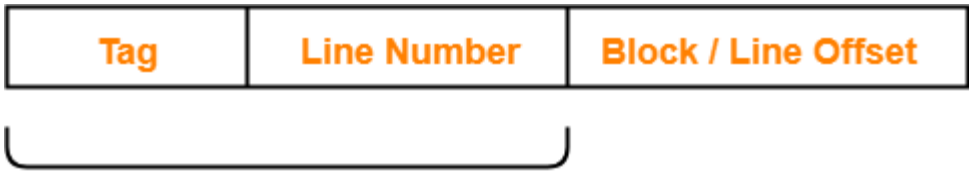
Need of Replacement Algorithm-

In direct mapping,

- There is no need of any replacement algorithm.
- This is because a main memory block can map only to a particular line of the cache.
- Thus, the new incoming block will always replace the existing block (if any) in that particular line.

Division of Physical Address-

In direct mapping, the physical address is divided as-



Division of Physical Address in Direct Mapping

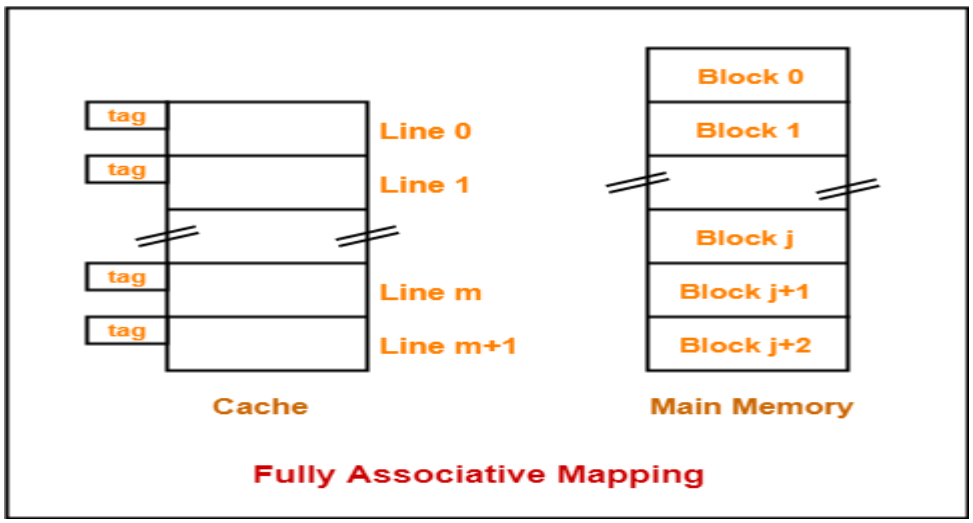
2. Fully Associative Mapping-

In fully associative mapping,

- A block of main memory can map to any line of the cache that is freely available at that moment.
- This makes fully associative mapping more flexible than direct mapping.

Example-

Consider the following scenario-



Fully Associative Mapping

Here,

- All the lines of cache are freely available.
- Thus, any block of main memory can map to any line of the cache.
- Had all the cache lines been occupied, then one of the existing blocks will have to be replaced.

Need of Replacement Algorithm-

In fully associative mapping,

- A replacement algorithm is required.
- Replacement algorithm suggests the block to be replaced if all the cache lines are occupied.
- Thus, replacement algorithm like FCFS Algorithm, LRU Algorithm etc is employed.

Division of Physical Address-

In fully associative mapping, the physical address is divided as-

Division of Physical Address in Fully Associative Mapping

3. K-way Set Associative Mapping-

In k-way set associative mapping,

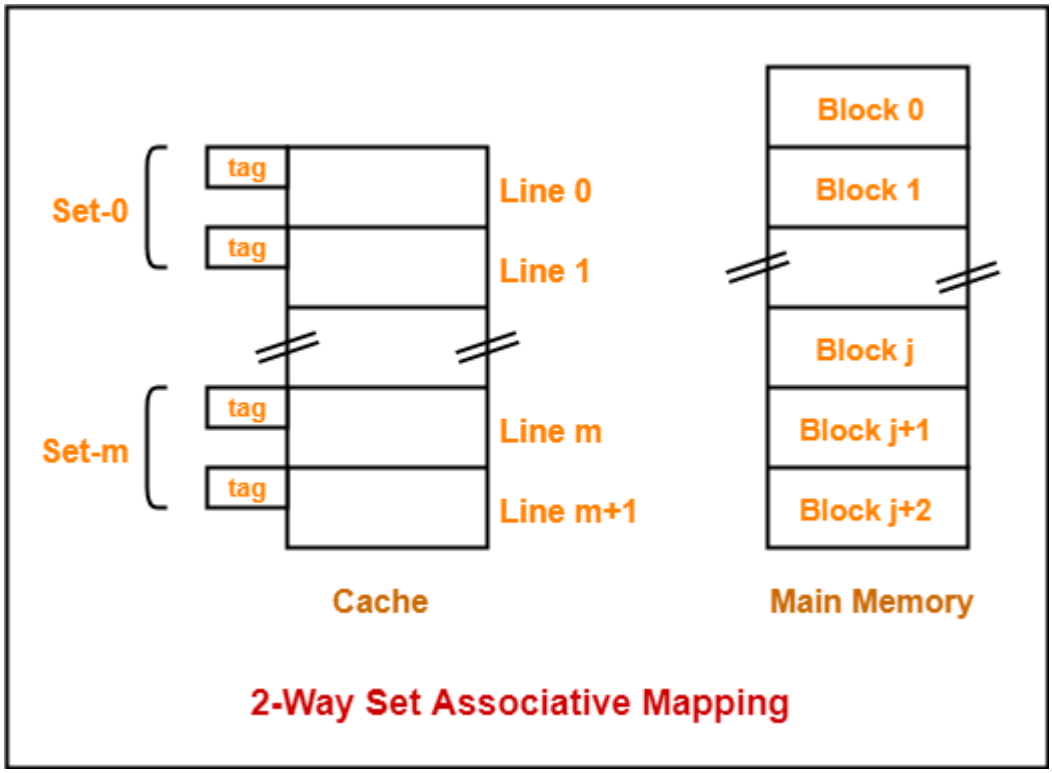
- Cache lines are grouped into sets where each set contains k number of lines.
- A particular block of main memory can map to only one particular set of the cache.
- However, within that set, the memory block can map any cache line that is freely available.
- The set of the cache to which a particular block of the main memory can map is given by-

Cache set number

= (Main Memory Block Address) Modulo (Number of sets in Cache)

Example-

Consider the following example of 2-way set associative mapping-



Here,

- k = 2 suggests that each set contains two cache lines.
- Since cache contains 6 lines, so number of sets in the cache = 6 / 2 = 3 sets.
- Block ‘j’ of main memory can map to set number (j mod 3) only of the cache.
- Within that set, block ‘j’ can map to any cache line that is freely available at that moment.
- If all the cache lines are occupied, then one of the existing blocks will have to be replaced.

Need of Replacement Algorithm-

- Set associative mapping is a combination of direct mapping and fully associative mapping.
- It uses fully associative mapping within each set.
- Thus, set associative mapping requires a replacement algorithm.

Division of Physical Address-

In set associative mapping, the physical address is divided as-

Division of Physical Address in K-way Set Associative Mapping

Special Cases-

- If k = 1, then k-way set associative mapping becomes direct mapping i.e.

1-way Set Associative Mapping ≡ Direct Mapping

- If k = Total number of lines in the cache, then k-way set associative mapping becomes fully associative mapping.

Direct Mapping-

In direct mapping,

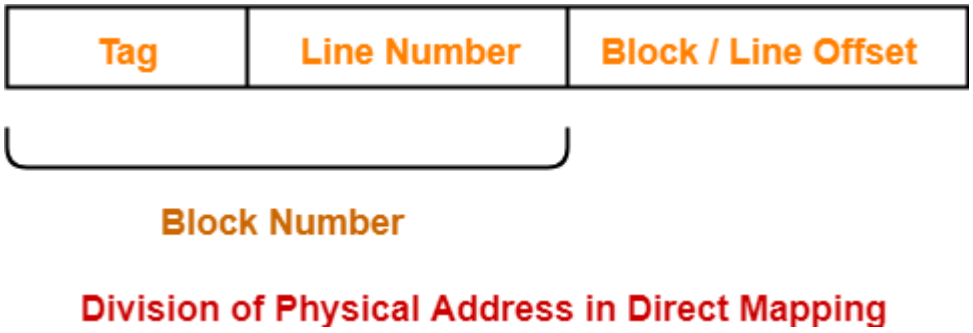
- A particular block of main memory can map to only one particular line of the cache.
- The line number of cache to which a particular block can map is given by-

Cache line number

= (Main Memory Block Address) Modulo (Number of lines in Cache)

Division of Physical Address-

In direct mapping, the physical address is divided as-



Division of Physical Address in Direct Mapping

Direct Mapped Cache-

Direct mapped cache employs direct cache mapping technique.

The following steps explain the working of direct mapped cache-

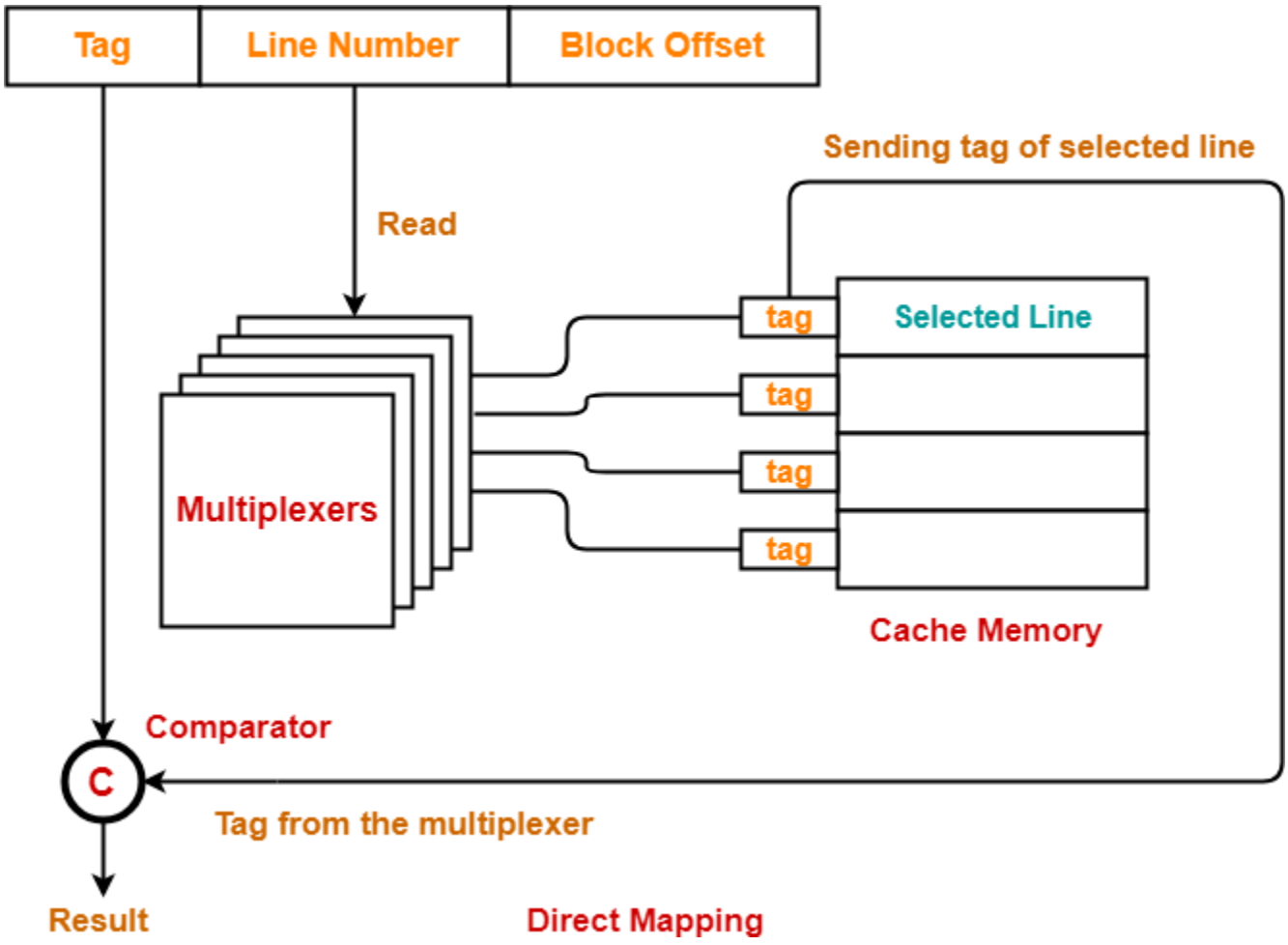
After CPU generates a memory request,

- The line number field of the address is used to access the particular line of the cache.
- The tag field of the CPU address is then compared with the tag of the line.
- If the two tags match, a cache hit occurs and the desired word is found in the cache.
- If the two tags do not match, a cache miss occurs.

- In case of a cache miss, the required word has to be brought from the main memory.
- It is then stored in the cache together with the new tag replacing the previous one.

Implementation-

The following diagram shows the implementation of direct mapped cache-



The steps involved are as follows-

Step-01:

- Each multiplexer reads the line number from the generated physical address using its select lines in parallel.

- To read the line number of L bits, number of select lines each multiplexer must have = L.

Step-02:

- After reading the line number, each multiplexer goes to the corresponding line in the cache memory using its input lines in parallel.
- Number of input lines each multiplexer must have = Number of lines in the cache memory

Step-03:

- Each multiplexer outputs the tag bit it has selected from that line to the comparator using its output line.
- Number of output line in each multiplexer = 1.

UNDERSTAND

It is important to understand-

- A multiplexer can output only a single bit on output line.
- So, to output the complete tag to the comparator,

  Number of multiplexers required = Number of bits in the tag
- Each multiplexer is configured to read the tag bit at specific location.

Example-

- 1st multiplexer is configured to output the first bit of the tag.
- 2nd multiplexer is configured to output the second bit of the tag.
- 3rd multiplexer is configured to output the third bit of the tag and so on.

So,

- Each multiplexer selects the tag bit of the selected line for which it has been configured and outputs on the output line.
- The complete tag as a whole is sent to the comparator for comparison in parallel.

Step-04:

- Comparator compares the tag coming from the multiplexers with the tag of the generated address.
- Only one comparator is required for the comparison where-

  Size of comparator = Number of bits in the tag

- If the two tags match, a cache hit occurs otherwise a cache miss occurs.

Hit latency-

The time taken to find out whether the required word is present in the **Cache Memory** or not is called as **hit latency**.

For direct mapped cache,

> Hit latency = Multiplexer latency + Comparator latency

Note:

Following are the few important results for direct mapped cache-

- Block j of main memory can map to line number (j mod number of lines in cache) only of the cache.
- Number of multiplexers required = Number of bits in the tag
- Size of each multiplexer = Number of lines in cache x 1
- Number of comparators required = 1
- Size of comparator = Number of bits in the tag
- Hit latency = Multiplexer latency + Comparator latency