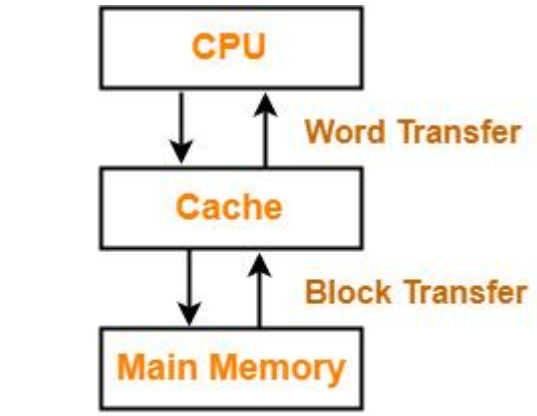
Cache Memory-

- 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)

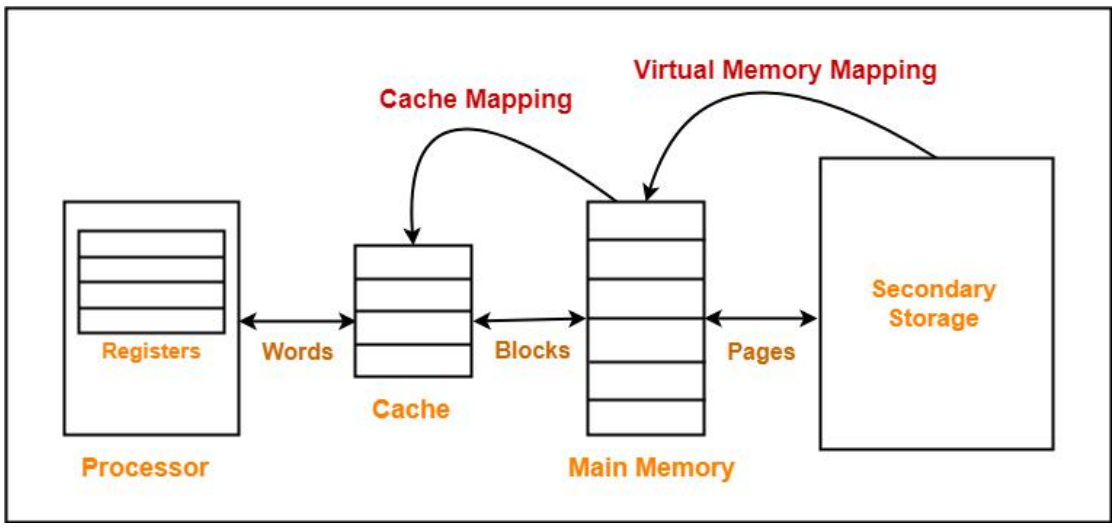
Cache Mapping-

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

OR

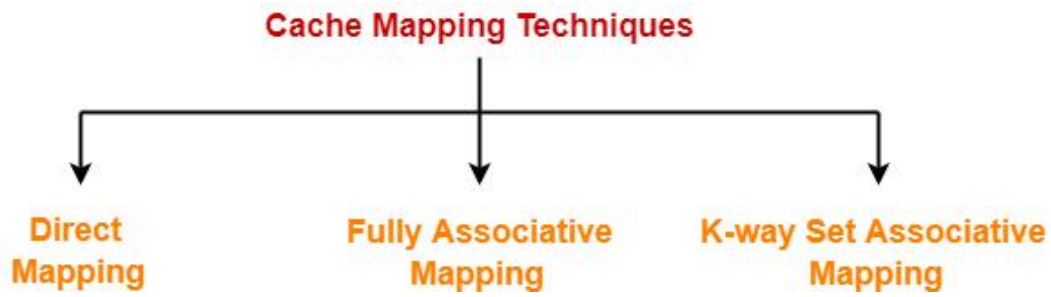
- Cache mapping is a technique by which the contents of main memory are brought into the cache memory.

The following diagram illustrates the mapping process-Now, before proceeding further, it is important to note the following points



Cache Mapping Techniques-

Cache mapping is performed using following three different techniques-



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

Direct Mapping–

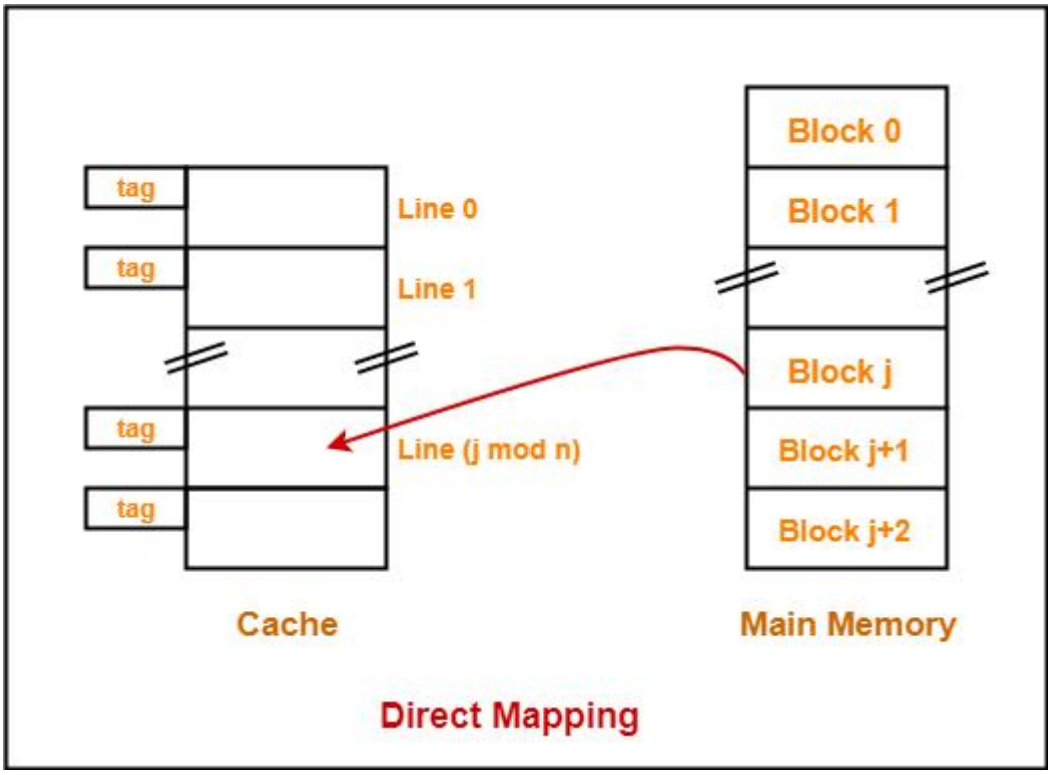
In direct mapping,

- 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.

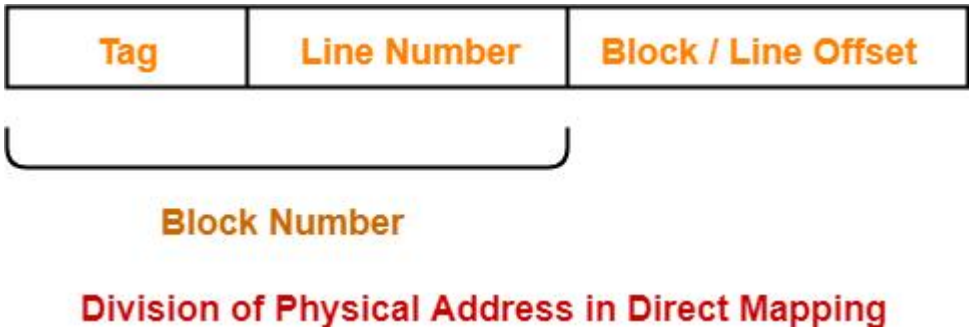


Need of Replacement Algorithm-

- 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

PRACTICE PROBLEMS BASED ON DIRECT MAPPING-

Problem-01:

Consider a direct mapped cache of size 16 KB with block size 256 bytes. The size of main memory is 128 KB. Find-

1. Number of bits in tag
2. Tag directory size

Solution-

Given-

- Cache memory size = 16 KB
- Block size = Frame size = Line size = 256 bytes
- Main memory size = 128 KB

We consider that the memory is byte addressable.

Number of Bits in Physical Address-

We have,

Size of main memory

= 128 KB

= 2^{17} bytes

Thus, Number of bits in physical address = 17 bits

Number of Bits in Block Offset-

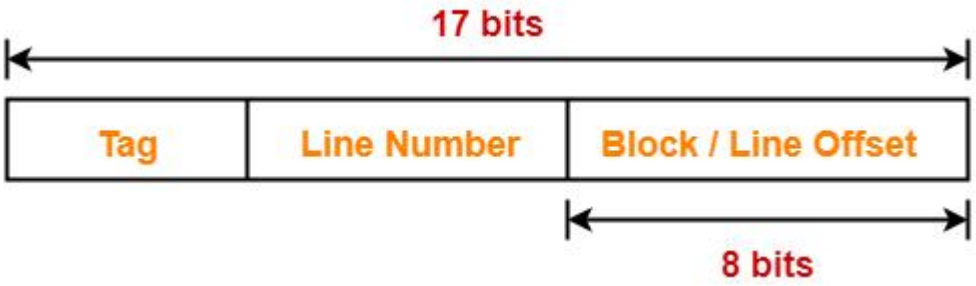
We have,

Block size

= 256 bytes

= 2^{8} bytes

Thus, Number of bits in block offset = 8 bits



Number of Bits in Line Number-

Total number of lines in cache

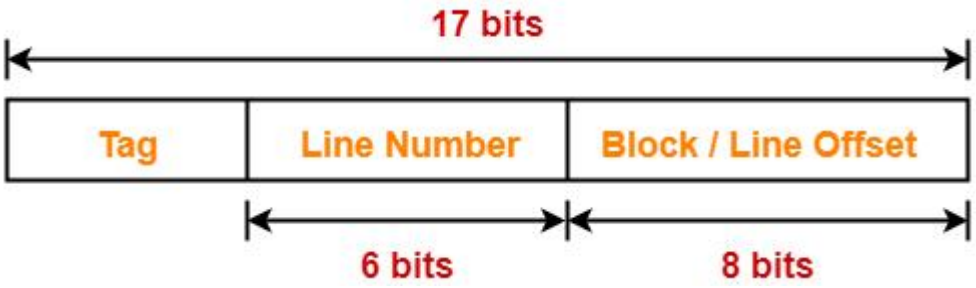
= Cache size / Line size

= 16 KB / 256 bytes

= 2^{14} bytes / 2^{8} bytes

= 2^{6} lines

Thus, Number of bits in line number = 6 bits



Number of Bits in Tag-

Number of bits in tag

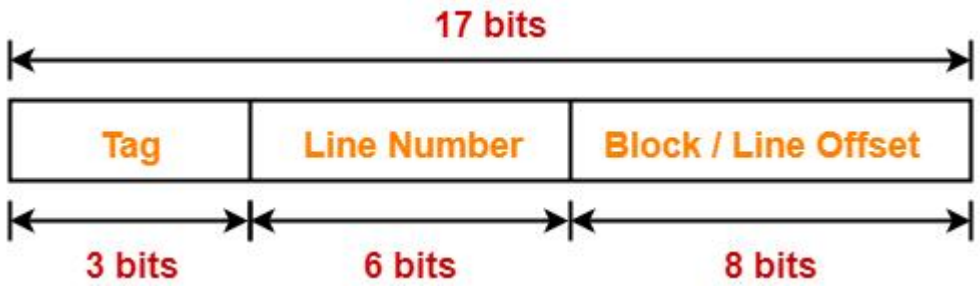
= Number of bits in physical address – (Number of bits in line number + Number of bits in block offset)

= 17 bits – (6 bits + 8 bits)

= 17 bits – 14 bits

= 3 bits

Thus, Number of bits in tag = 3 bits



Tag Directory Size-

Tag directory size

= Number of tags x Tag size

= Number of lines in cache x Number of bits in tag

= 2^6 x 3 bits

= 192 bits

= 24 bytes

Thus, size of tag directory = 24 bytes

Problem-02:

Consider a direct mapped cache of size 512 KB with block size 1 KB. There are 7 bits in the tag. Find-

1. Size of main memory
2. Tag directory size

Solution-

Given-

- Cache memory size = 512 KB
- Block size = Frame size = Line size = 1 KB
- Number of bits in tag = 7 bits

We consider that the memory is byte addressable.

Number of Bits in Block Offset-

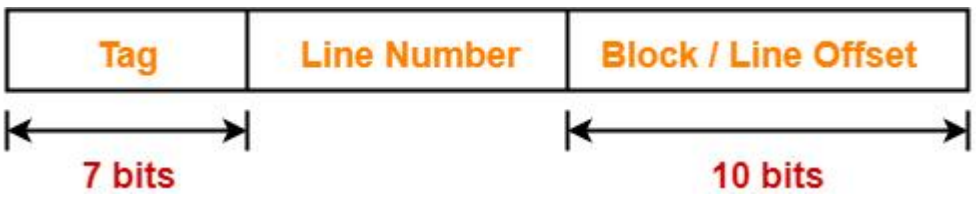
We have,

Block size

= 1 KB

= 2^{10} bytes

Thus, Number of bits in block offset = 10 bits



Number of Bits in Line Number-

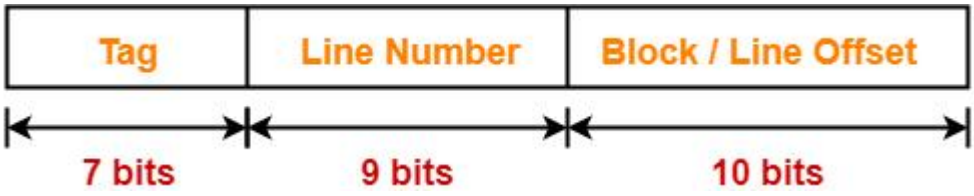
Total number of lines in cache

= Cache size / Line size

= 512 KB / 1 KB

= 2^{9} lines

Thus, Number of bits in line number = 9 bits



Number of Bits in Physical Address-

Number of bits in physical address

= Number of bits in tag + Number of bits in line number + Number of bits in block offset

= 7 bits + 9 bits + 10 bits

= 26 bits

Thus, Number of bits in physical address = 26 bits

Size of Main Memory-

We have,

Number of bits in physical address = 26 bits

Thus, Size of main memory

= 2^{26} bytes

= 64 MB

Tag Directory Size-

Tag directory size

= Number of tags x Tag size

= Number of lines in cache x Number of bits in tag

= 2^9 x 7 bits

= 3584 bits

= 448 bytes

Thus, size of tag directory = 448 bytes

Problem-03:

Consider a direct mapped cache with block size 4 KB. The size of main memory is 16 GB and there are 10 bits in the tag. Find-

1. Size of cache memory
2. Tag directory size

Solution-

Given-

- Block size = Frame size = Line size = 4 KB
- Size of main memory = 16 GB
- Number of bits in tag = 10 bits

We consider that the memory is byte addressable.

Number of Bits in Physical Address-

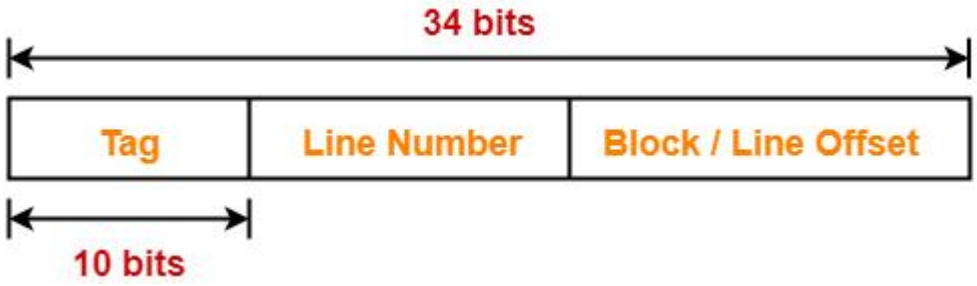
We have,

Size of main memory

= 16 GB

$= 2^{34}$ bytes

Thus, Number of bits in physical address = 34 bits



Number of Bits in Block Offset-

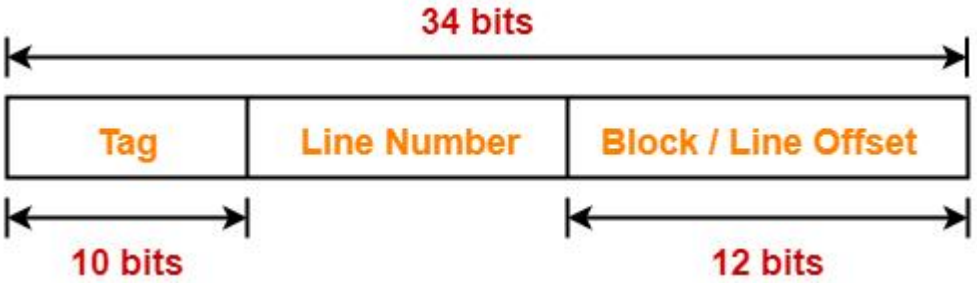
We have,

Block size

= 4 KB

$= 2^{12}$ bytes

Thus, Number of bits in block offset = 12 bits



Number of Bits in Line Number-

Number of bits in line number

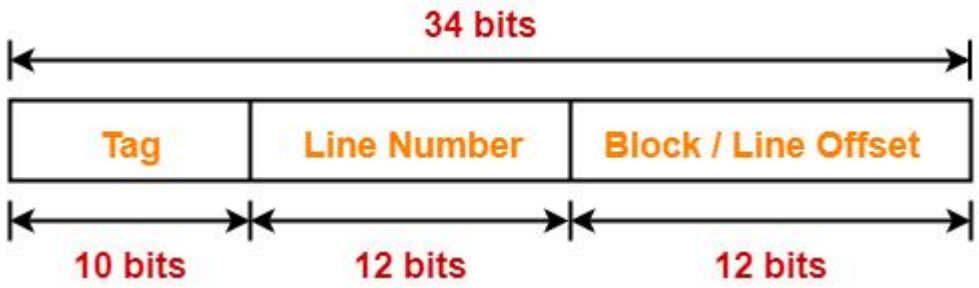
= Number of bits in physical address – (Number of bits in tag + Number of bits in block offset)

= 34 bits – (10 bits + 12 bits)

= 34 bits – 22 bits

= 12 bits

Thus, Number of bits in line number = 12 bits



Number of Lines in Cache-

We have-

Number of bits in line number = 12 bits

Thus, Total number of lines in cache = 2^{12} lines

Size of Cache Memory-

Size of cache memory

= Total number of lines in cache x Line size

= 2^{12} x 4 KB

= 2^{14} KB

= 16 MB

Thus, Size of cache memory = 16 MB

Tag Directory Size-

Tag directory size

= Number of tags x Tag size

= Number of lines in cache x Number of bits in tag

= 2^{12} x 10 bits

= 40960 bits

= 5120 bytes

Thus, size of tag directory = 5120 bytes

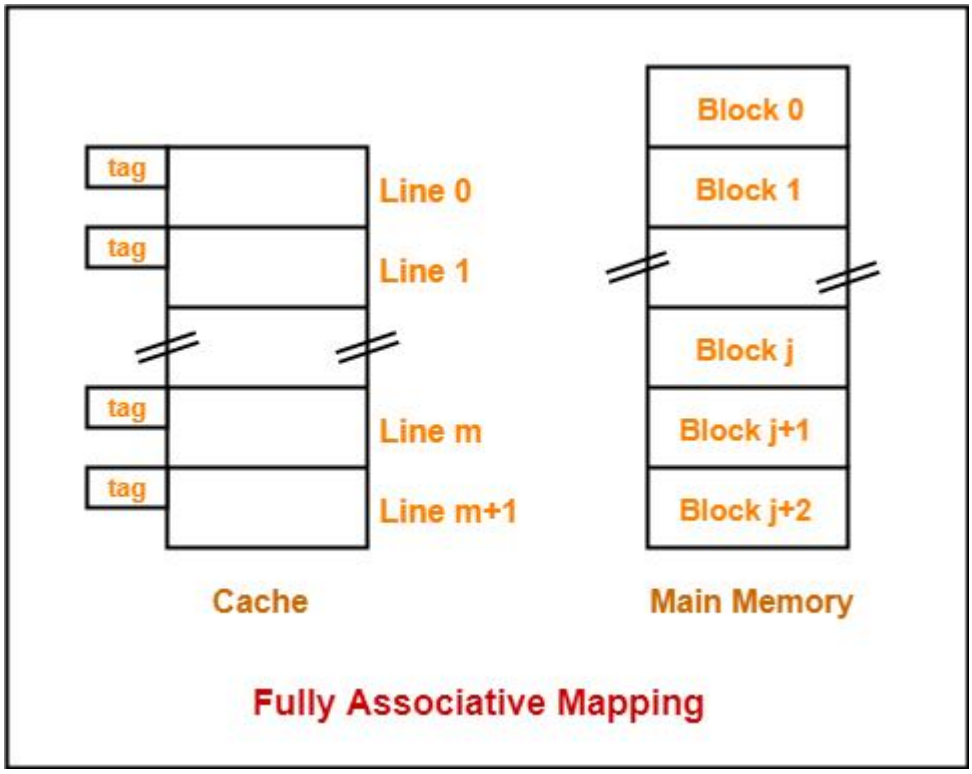
2. Fully Associative Mapping-

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-



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