Chapter - 07

MEMORY ORGANIZATION

Memory refers to any medium for data storage. It stores binary informations of program instructions as well as data.

7.1. Memory Hierarchy

The purpose of all memory systems is storage and retrival of data. But a wide variety of memories exists with contrasting performance and different characteristics. Some of the fundamental characteristics of memory are:

* read - write ability
* access time
* storage permanance and
* storage capability.

The memory hierarchy summarizes the memory types and their performance using pyramidal structure. The fig below shows the memory hierarchy.

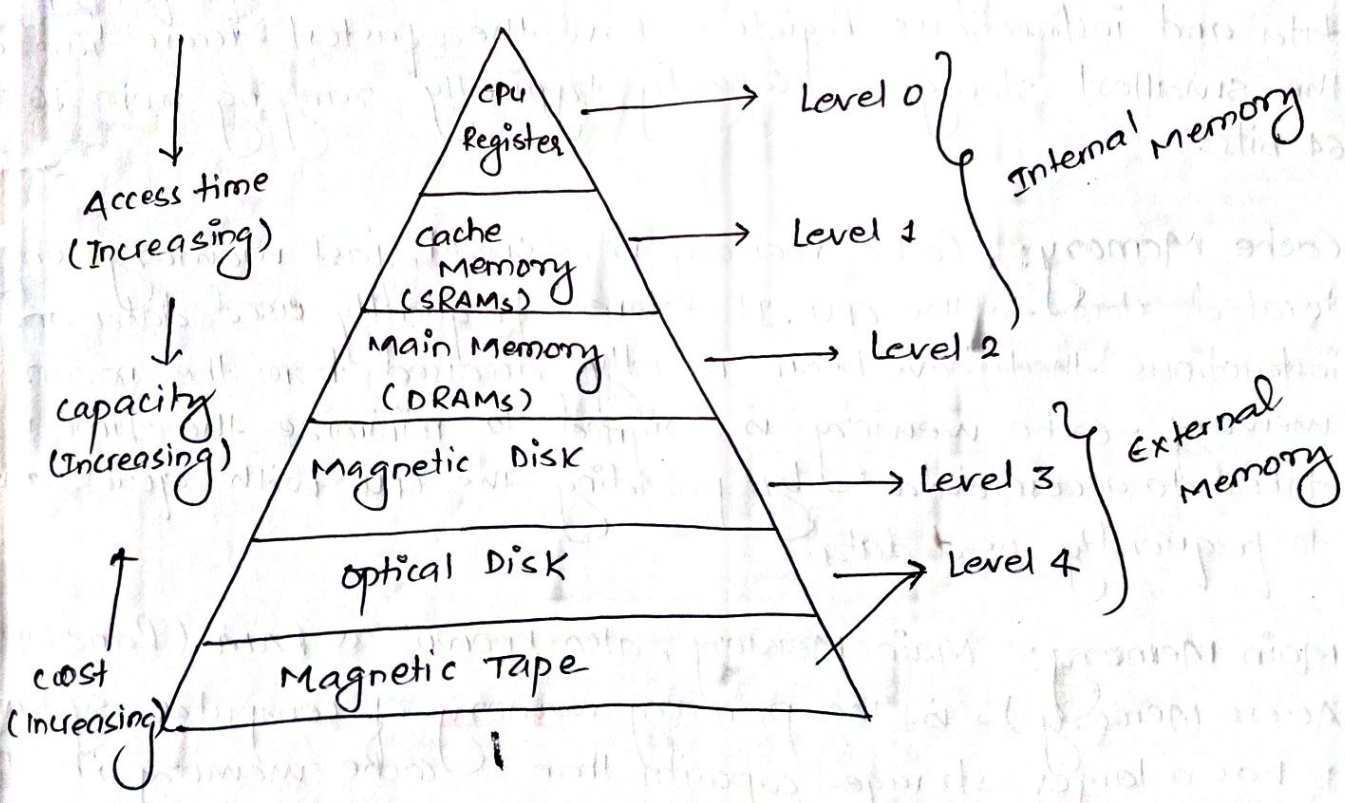


fig: Memory Hierarchy Design.

As one goes down the hierarchy

1. Distance from processor increases
2. Access time increases
3. Memory capacity increases
4. cost per bit decreases.

Hierarchy design is divided into 2 main types:

a. External Memory or secondary memory: comprising of Magnetic disk, optical Disk, Magnetic Tape ie. peripheral storage devices which are accessible by the processor via I/O modules.

b. Internal Memory or Primary Memory: comprising of main Memory, cache Memory and CPU registers. This is directly acc ible by the processor.

There are typically four levels of memory in a memory hierarc

Registers: Registers are small, high speed memory units locate in the CPU. They are used to store the most frequently us data and instructions. Registers have the fastest access time an the smallest storage capacity, typically ranging from 16 to 64 bits.

Cache Memory: Cache memory is a small, fast memory unit located close to the CPU. It stores frequently used data and instructions that have been recently accessed from the main memory. cache memory is designed to minimize the time it takes to access data by providing the CPU with quick acc to frequently used data.

Main Memory: Main Memory, also known as RAM (Random Access Memory), is the primary memory of computer systen It has a larger storage capacity than cache memory, but i is slower. Main memory is used to store data and instruct that are currently in use by the CPU.

Types of Main Memory:

i) Static RAM: It stores the binary information in flip flops and information remains valid until power is supplied. It has faster access time and is used in implementing cache memory.

ii) Dynamic RAM: It stores the binary information as a charge on the capacitor. It requires refreshing circuitry to maintain the charge on the capacitors after few milliseconds. It contains more memory cells per unit area as compared to SRAM.

Secondary storage:

Secondary storage, such as hard disk drives (HDD) and solid state drives (SSD), is a non-volatile memory unit that has a larger storage capacity than the main memory. It is used to store data and instructions that are not currently in used by the CPU. Secondary storage has the lowest access time and is typically the least expensive type of memory in the memory hierarchy.

7.2 Associative Memory:

An associative memory can be treated as memory unit whose saved information can be recognized for approach by content of the information itself instead of by an address or memory location.

Associative memory is also known as **Content Addressable Memory (CAM).**

Associative memory is accessed simultaneously and in parallel on the basic of data content rather by specific address or location.

The block diagram of associative memory is shown in fig. below:

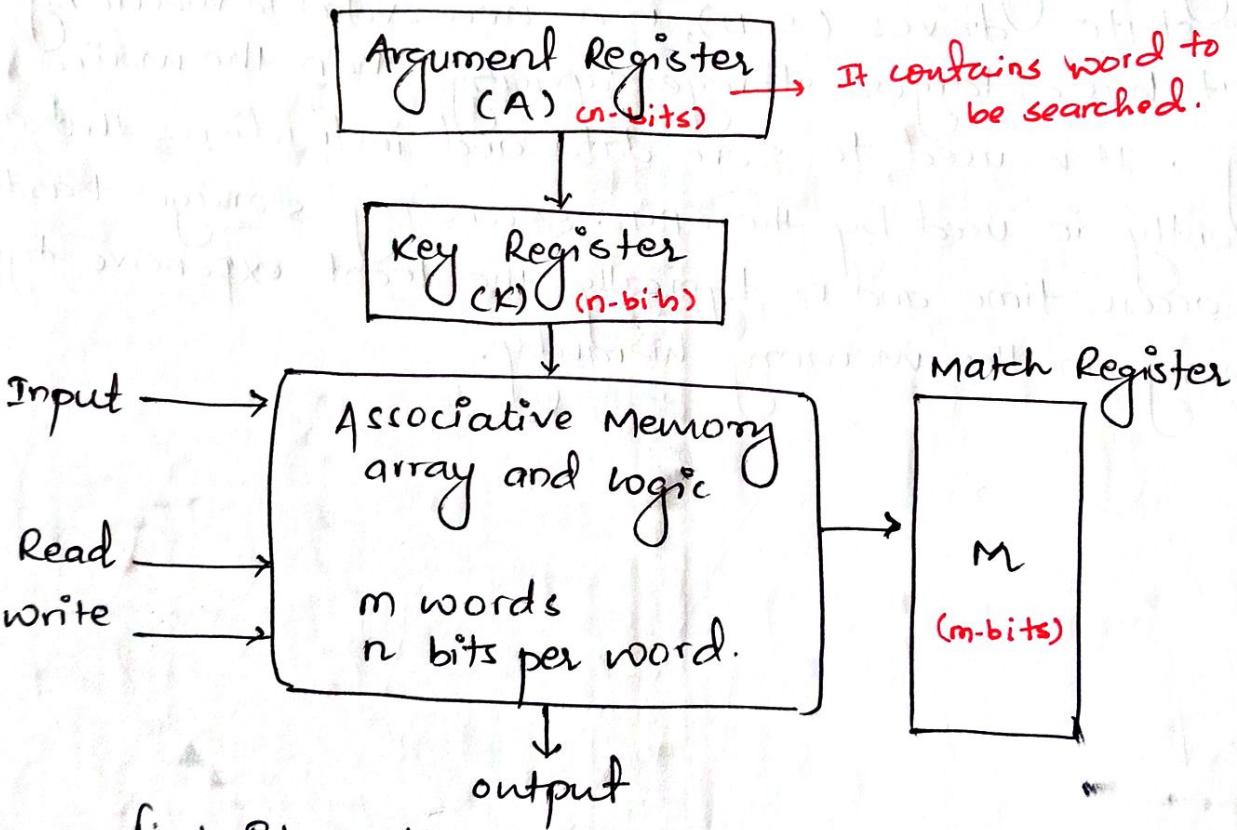


fig: Block diagram of Associative Memory.

↳ It includes a memory array and logic for m words with n bits per word. The argument register A and key register K each have n bits, one for each bit of a word.

↳ The match register M has m bits, one for each memory word. Each word in memory is related in parallel to the content of the argument register.

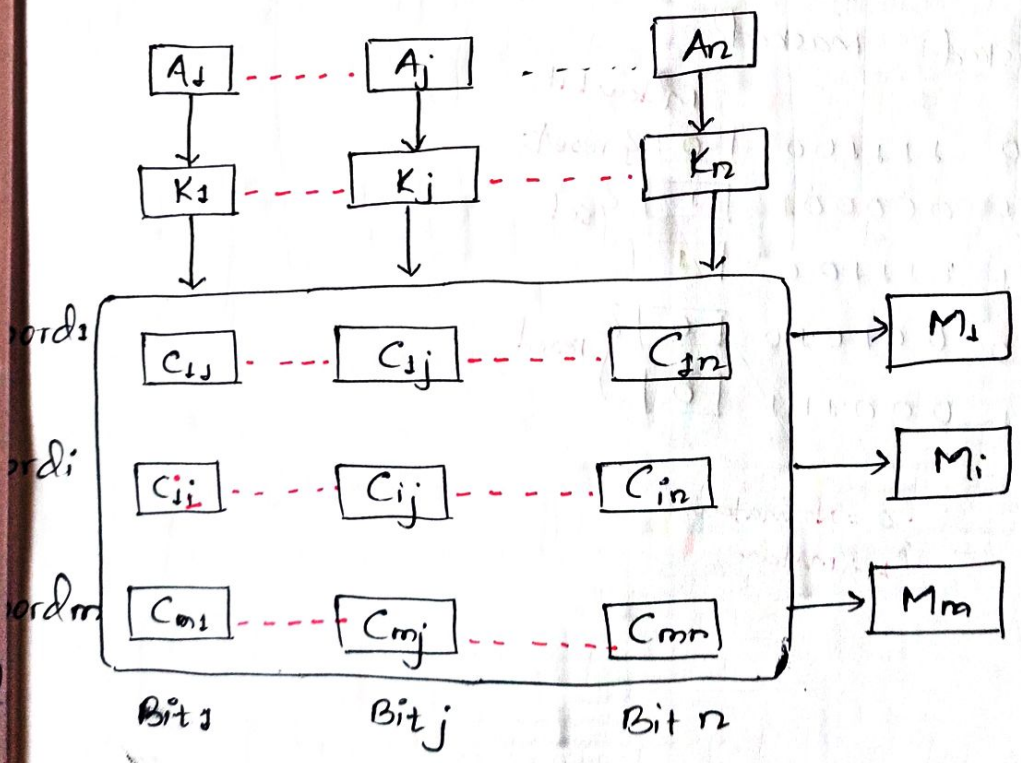
The words that connect the bits of the argument register set an equivalent bit in the match register. After the matching process, those bits in the match register that have been set denote the fact that their equivalent words have been connected.

The key register supports a mask for selecting a specific field or key in the argument word. The whole argument is distinguished with each memory word if the key register includes all 1's.

Hence, there are only those bits in the argument that have 1's in their equivalent position of the key register are compared. Therefore, the key gives a mask or ~~rege~~ recognizing a piece of data that determines how the reference to memory is created.

The following figure can define the relation between the memory array and the external registers in associative memory.

Associative Memory of m word, n cells per word



The cells in the array are considered by the letter C with two subscripts. The first subscript provides the word number and the second determines the bit position in the word. Therefore cell C_{ij} is the cell for bit j in word i.

A bit in the argument register is compared with all the bits in column j of the array supported that $K_j = 1$. ~~Therefore cell Cij is the cell for bit j in word. Then~~ This is completed for all columns $j = 1, 2, \ldots n$.

If a match appears between all the unmasked bits of the argument and the bits in word i, the equivalent bit M_i in the match register is set to 1. If one or more unmasked bits of the argument and the word do not match, M_i is cleared to 0.

eg:

A 101 111100

K 111 000000

↑ unmasked ↑ masked.

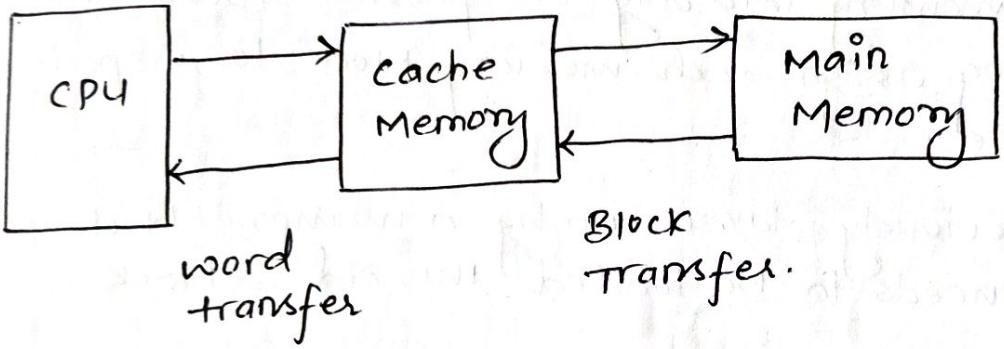
			Match bit	
word 1	100	111100	0	} reset
word 2	101	000001	1	} set
word 3	101	111100	1	
word 4	110	001010	0	} reset
word 5	111	000111	0	

0 = No match
1 = Match

7.3 Cache Memory and Cache Mapping Technique.

The main memory is slower than processor. A designer would like to build a main memory as fast as the processor but that would significantly increase the cost. A direct and frequent access to the main memory would slow down the processor operation. To address this problem, a faster memory whose speed is same as that of the processor is employed which is known as cache.

This memory resides within the processor chip and contains a copy of block of main memory so that processor can access contents faster. Cache memory comprises of SRAM cells which are expensive but faster to access.



When CPU needs to access some memory, cache is examined. If the desired word is available in the cache it indicates cache hit and processing continues. If it is not available in cache, it indicates cache miss and block of words including the desired word is transfered into cache from main memory.

Cache connects to the processor via address, data and control bus. The address and data bus are connected to the address and data buffer as well.

$$\text{Hit ratio (H)} = \frac{\text{hit}}{(\text{hit} + \text{miss})} = \frac{\text{no. of hits}}{\text{total accesses}}$$

$$\text{Miss ratio (M)} = \frac{\text{miss}}{(\text{hit} + \text{miss})} = \frac{\text{no. of miss}}{\text{Total accesses}} = 1 - \text{hit ratio(H)}$$

Cache Mapping Technique

Basically cache mapping refers to a technique using which the content present in the main memory is brought into the memory of the cache.

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

1. Direct Mapping
2. Associative Mapping
3. set-Associative Mapping.

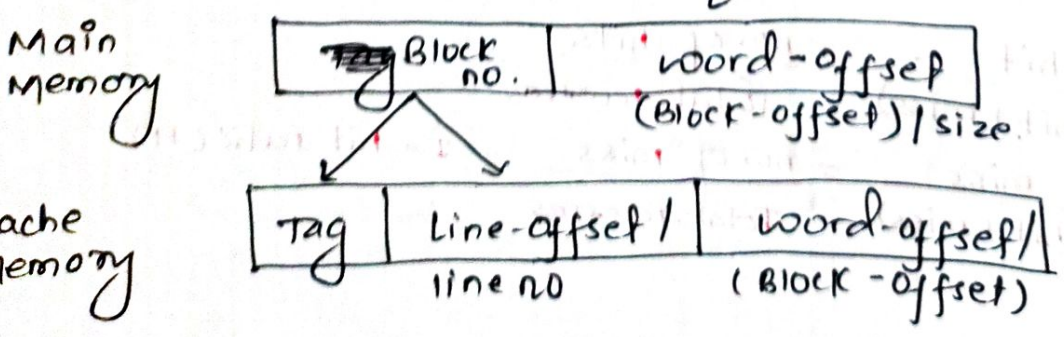
1. Direct Mapping:

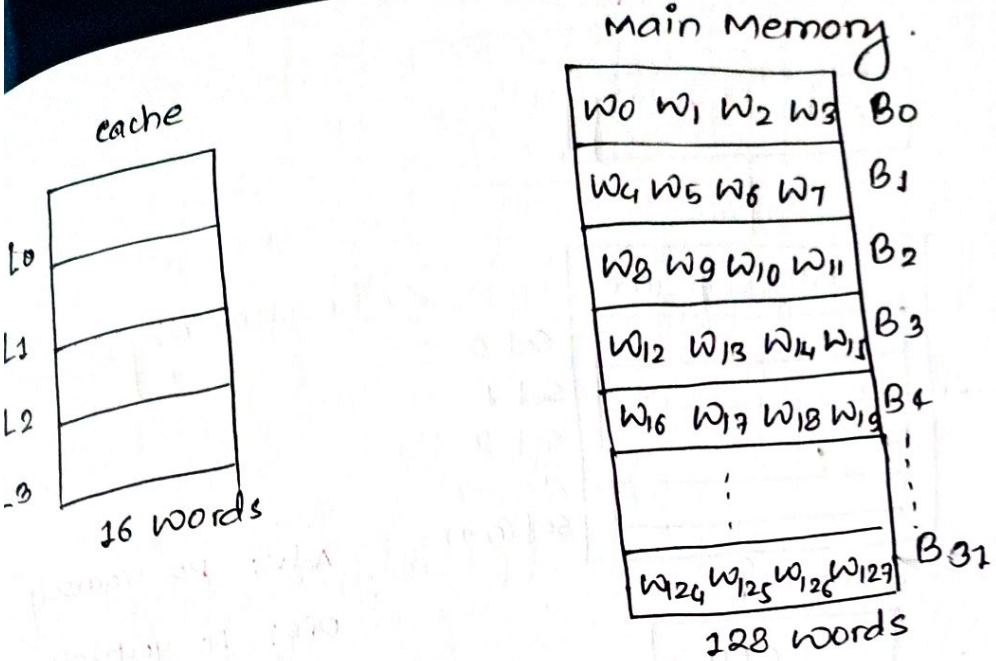
- The simplest technique, known as direct mapping, maps each block of main memory into only one possible cache line or In direct mapping, assign each memory block to a specific line in the cache.
- If a line is previously taken up by a memory block wh a new block needs to be loaded, the old block is trashed.
- An address is split into two parts index field and tag field. The cache is used to store the tag field whes the rest is stored in the main memory. Direct mapping performance is directly proportional to the Hit ratio

$$i = j \bmod m$$

where

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





(1 byte = 1 word)

One block contains 4 words.

Total No of blocks = $\frac{128}{4}$ = 32 Blocks.

Lines size = Block size

No. of lines = $\frac{16}{4}$ = 4 lines.

We have,

$i = j \bmod m$

Let $j = 0$ and $m = 4$ Then, $i = 0 \bmod 4 = 0$ (ie. Line no zero)

$j = 1$ and $m = 4$ Then, $i = 1 \bmod 4 = 1$ (ie. L_1)

$j = 2$ and $m = 4$ Then, $i = 2 \bmod 4 = 2$ (ie. L_2)

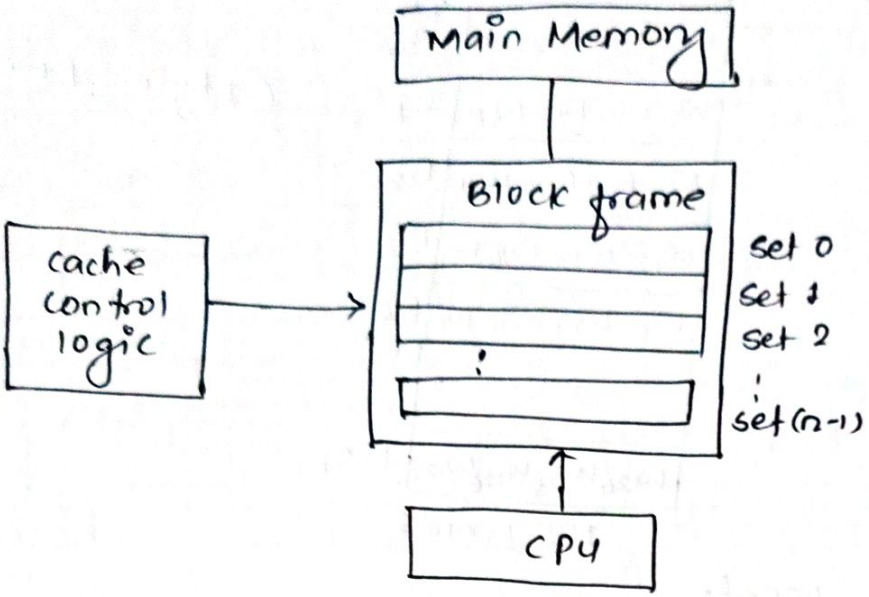
$j = 3$ and $m = 4$ Then, $i = 3 \bmod 4 = 3$ (ie. L_3)

$j = 4$ and $m = 4$ Then, $i = 4 \bmod 4 = 0$ (ie. L_0)

Cache Memory.

L_0	B_0	B_4	B_8	B_{12}	---- B_{28}
L_1	B_1	B_5	B_9	B_{13}	--- B_{29}
L_2	B_2	B_6	B_{10}	B_{14}	--- B_{30}
L_3	B_3	B_7	B_{11}	B_{15}	---- B_{31}

When we store main memory address content in cache, we also store the tag. To determine if desired main memory content is available in the cache or not, we go to the cache address indicated by the index and we compare the tag available there with the desired tag.



cache memory.

Adv: Do yourself

Dis: Do yourself.

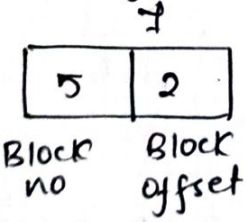
fig: Direct Mapping.

2. Associative Mapping

↳ In this type of mapping, associative memory is used to store the content and addresses of the memory word. Any block can go to 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 most flexible mapping forms. In associative mapping, the index bits are zero.

↳ we have, the physical address (PA) is of 7 bit (ie. $2^7 = 128$ no



↳ In cache memory, there is no need of line-offset block. Therefore our Tag is the block no.

ie. Tag = Block no = 5 bit

This shows that $2^5 = 32$ (0-32) blocks can appear in any line in cache.

L_0	B_0	B_1	B_2	B_3	B_4	- - - - -	B_{31}
L_1	B_0	B_1	B_2	B_3	B_4	- - - -	B_{31}
L_2	B_0	B_1	B_2	B_3	B_4	- - - - -	B_{31}
L_3	B_0	B_1	B_2	B_3	B_4	- - - - -	B_{31}

Advantage:

1. Conflict miss issue is resolved
2. Hit ratio/rate is increased

Disadvantage:

1. Primary disadvantage is that the comparison time is increased.

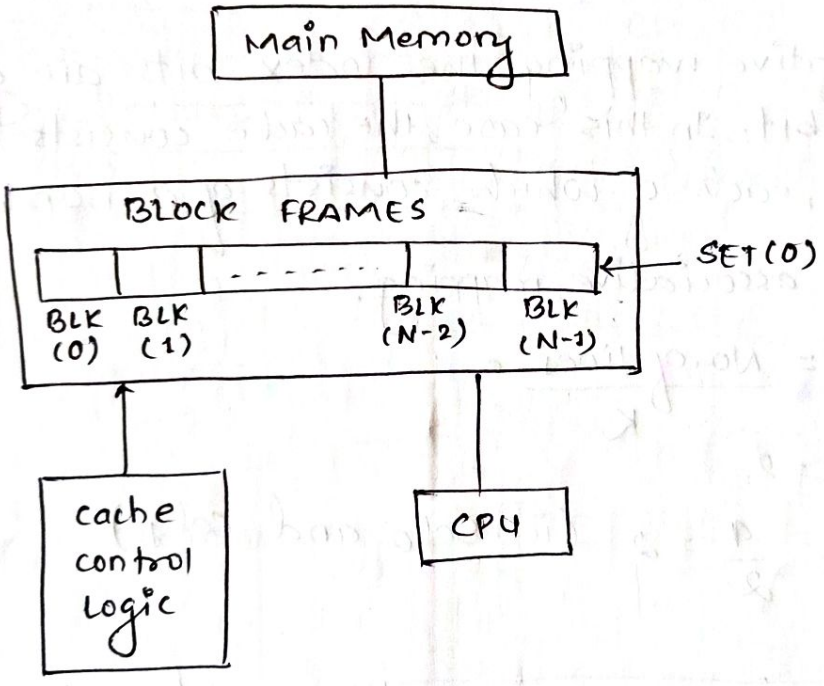


fig: Associative Mapping structure.

3. Set-Associative Mapping

This form of mapping is an enhanced form of direct mapping where the drawbacks of direct mapping are removed. Set asso e addresses the problem of possible trashing in the direct mapping method. It does this by saying that instead of ha exactly one line that a block can map to in the cache. we will group a few lines together creating a set.

↳ also known as **k-way** set associative.

↳ Then a block in memory can map to any one of the lines of a specific set.

↳ set-associative mapping allows each word that is present in the cache can have two or more words in the main memory for the same index address.

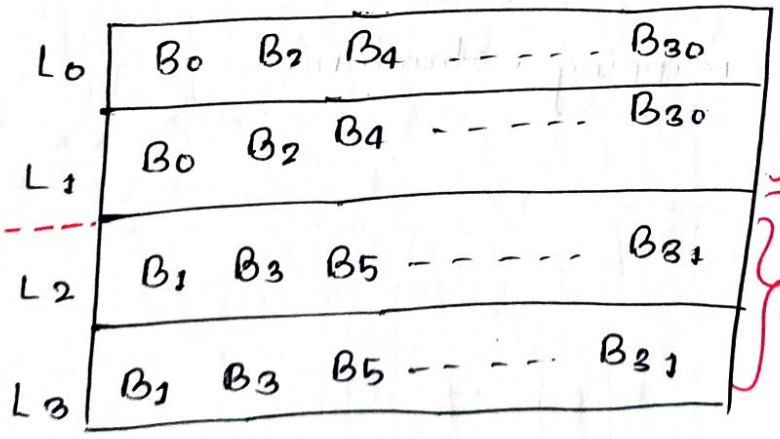
↳ In set associative mapping the index bits are given by the set offset bit. In this case, the cache consists of a number of sets, each of which consists of a number of lines.

↳ for 2-way set associative mapping.

$$\text{No of set} = \frac{\text{No. of lines}}{K}$$

for $K = 2$,

$$\text{No of set} = \frac{4}{2} = 2 \quad (\text{ie. Set}_0 \text{ and Set}_1)$$



formula:

$i = j \bmod n$

i = Line number
j = Block number
n = no. of set.

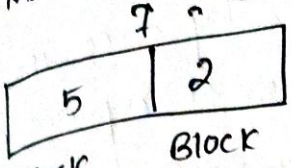
eg:

0 mod 2 = 0 (Set0)
1 mod 2 = 1 (Set1)
2 mod 2 = 0 (Set0)
3 mod 2 = 1 (Set1)

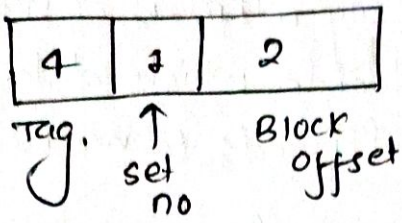
NOTE: $B_0, B_2, B_4, \ldots B_{30}$ line 0 or line 1 मध्ये कुठल्याही line मा जाऊन बसू शकतात.

$B_1, B_3, \ldots B_3$ line 2 or line 3 मा map होऊ शकतात.

Main Memory



cache memory.



(ie. $2^1 = 2 =$ S0 and S1)

Here, Tag bit = 4 = 2^4 = 16 combination.

ie. $B_0, B_2, B_4 ---- B_{30}$ (16 Blocks)

OR,

$B_1, B_3, B_5, ---- B_{31}$ (16 BLOCKS)

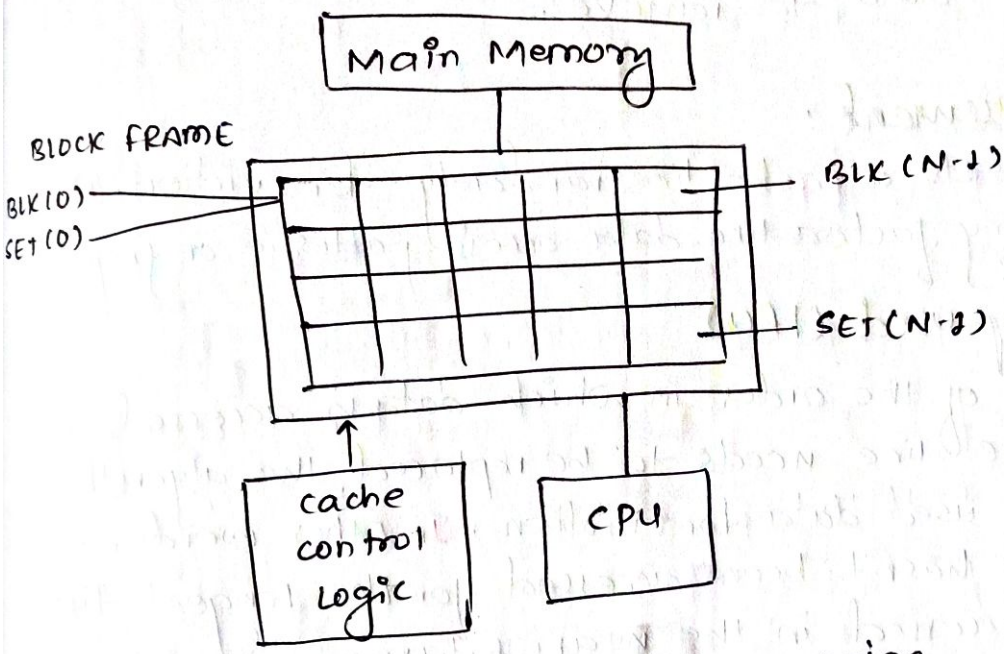


fig: set-Associative mapping.

Advantage:

The comparison time is reduced in the K-way set associative mapping as compared to the fully associative mapping.

Disadvantage:

1. It involves the factor of conflict miss.

7.4. Cache Replacement Algorithm:

Cache replacement algorithm is a method used by a computer's cache memory to decide which data to remove or replace when the cache is full and needs to make room for new data.

Cache replacement algorithms help determine which data should be evicted from the cache to make space for new data. These algorithms aims to maximize the cache hit rate and minimize cache misses, as cache hits result in faster access time.

↳ Different cache replacement algorithms use various strategies to decide which data to remove.

a) Random Replacement:

This algorithm selects a cache line randomly for eviction. It does not consider any factors like data access patterns or frequency.

b) Least Recently used (LRU)

LRU keeps track of the order in which data is accessed in the cache. When cache line needs to be replaced, the algorithm sel the least recently used data for eviction. In other words, it assu that the data that hasn't been accessed for the longest time is less likely to be accessed in the near future.

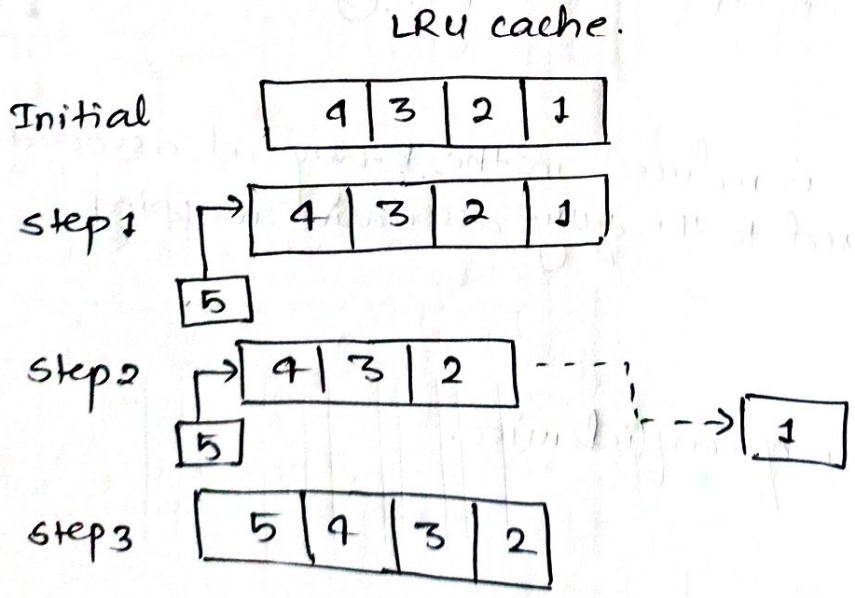


fig: LRU cache

c) Least frequently used (LFU)

The LFU cache replacement algorithm is designed to remove or replace data from a cache based on the frequency of its access. It assumes that data items with lower access frequencies are less likely to be accessed in the near future, so it prioritizes evicting those items.

The LFU algorithm maintains a counter for each data items in the cache, which keeps track of how many times that item has been accessed. When the cache becomes full and needs to make rooms for new data, the LFU algorithm identifies the data tems with the lowest access frequency and removes it from the cache.

d) FIFO (First In first out)

FIFO replaces the data that has been in the cache the longest. It works on the principle of a queue, where the first data to enter the cache is the first one to be evicted when necessary.

objects are added to the queue and are evicted with the same order. Even though it provides a simple and low-cost method to manage the cache but even the must used objects are eventually evicted when they're old enough.

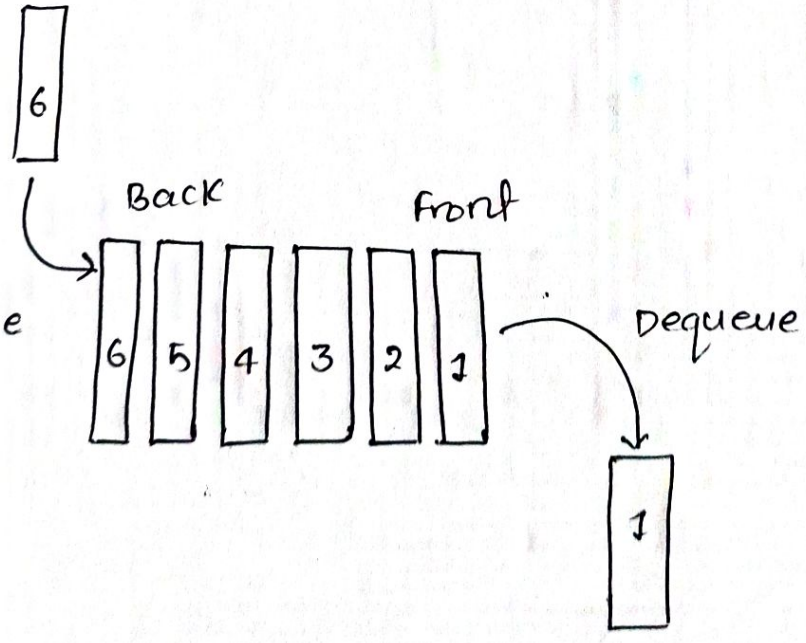


fig: FIFO Technique.