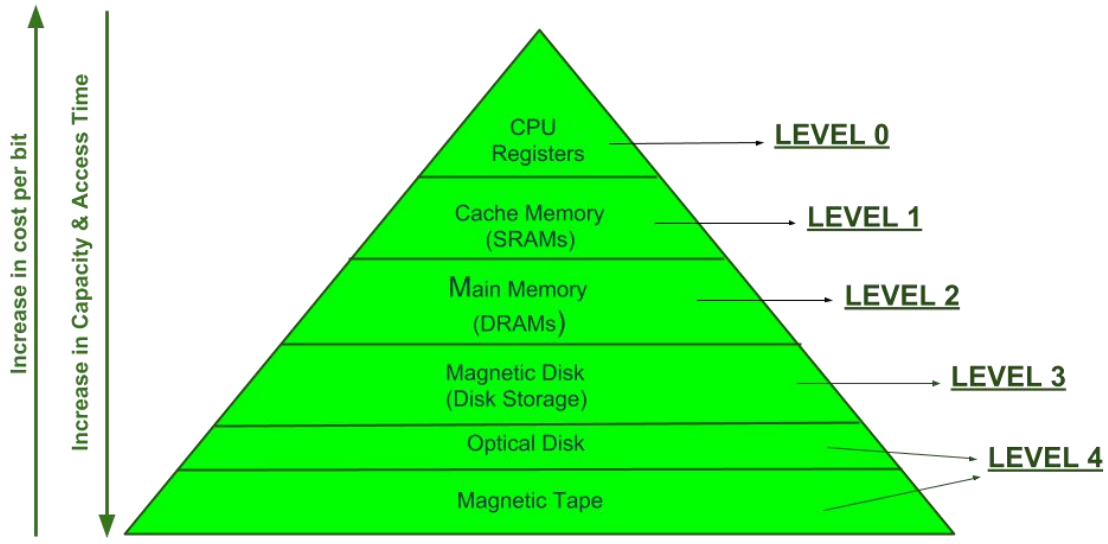
4th Semester CE/IT/CSE

CE0404 - Computer Organization & Architecture

Unit-4 Memory Organization

1. Memory Hierarchy

In the Computer System Design, Memory Hierarchy is an enhancement to organize the memory such that it can minimize the access time. The Memory Hierarchy was developed based on a program behavior known as locality of references.The figure below clearly demonstrates the different levels of memory hierarchy :



MEMORY HIERARCHY DESIGN

Hierarchy Design is divided into 2 main types:

- **External Memory or Secondary Memory** – Comprising of Magnetic Disk, Optical Disk, Magnetic Tape i.e. peripheral storage devices which are accessible by the processor via I/O Module.
- **Internal Memory or Primary Memory** – Comprising of Main Memory, Cache Memory & CPU registers. This is directly accessible by the processor.How pipelining works.

There are typically four levels of memory in a memory hierarchy:

- **Registers**: Registers are small, high-speed memory units located in the CPU. They are used to store the most frequently used data and instructions. Registers have the fastest access time and 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 access to frequently used data.
- **Main Memory**: Main memory, also known as RAM (Random Access Memory), is the primary memory of a computer system. It has a larger storage capacity than cache memory, but it is slower. Main memory is used to store data and instructions that are currently in use by the CPU.

  Types of Main memory:
  - **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.
  - **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 main memory. It is used to store data and instructions that are not currently in use by the CPU. Secondary storage has the slowest access time and is typically the least expensive type of memory in the memory hierarchy.

We can infer the following characteristics of Memory Hierarchy Design from above figure:

- **Capacity**: It is the global volume of information the memory can store. As we move from top to bottom in the Hierarchy, the capacity increases.
- **Access Time**: It is the time interval between the read/write request and the availability of the data. As we move from top to bottom in the Hierarchy, the access time increases.
- **Performance**: Earlier when the computer system was designed without Memory Hierarchy design, the speed gap increases between the CPU registers and Main Memory due to large difference in access time. This results in lower performance of the system and thus, enhancement was required. This enhancement was made

in the form of Memory Hierarchy Design because of which the performance of the system increases. One of the most significant ways to increase system performance is minimizing how far down the memory hierarchy one has to go to manipulate data.

- **Cost per bit**: As we move from bottom to top in the Hierarchy, the cost per bit increases i.e. Internal Memory is costlier than External Memory.

According to the memory Hierarchy, the system supported memory standards are defined below:

Level	1	2	3	4
Name	Register	Cache	Main Memory	Secondary Memory
Size	<1 KB	less than 16 MB	<16GB	>100 GB
Implementation	Multi-ports	On-chip/SRAM	DRAM (capacitor memory)	Magnetic
Access Time	0.25ns to 0.5ns	0.5 to 25ns	80ns to 250ns	50 lakh ns
Bandwidth	20000 to 1 lakh MBytes	5000 to 15000	1000 to 5000	20 to 150
Managed by	Compiler	Hardware	Operating System	Operating System
Backing Mechanism	From cache	from Main Memory	from Secondary Memory	from ie

2. Types of Memory

In general, memory can be divided into primary and secondary memory; moreover, there are numerous types of memory when discussing just primary memory. Some types of primary memory include the following

- **Cache memory**. This temporary storage area, known as a cache, is more readily available to the processor than the computer's main memory source. It is also called CPU memory because it is typically integrated directly into the CPU chip or placed on a separate chip with a bus interconnect with the CPU.
- **RAM.** The term is based on the fact that any storage location can be accessed directly by the processor.

- **Dynamic RAM.** DRAM is a type of semiconductor memory that is typically used by the data or program code needed by a computer processor to function.
- **Static RAM.** SRAM retains data bits in its memory for as long as power is supplied to it. Unlike DRAM, which stores bits in cells consisting of a capacitor and a transistor, SRAM does not have to be periodically refreshed.
- **Double Data Rate SDRAM.** DDR SRAM is SDRAM that can theoretically improve memory clock speed to at least 200 MHz.
- **Double Data Rate 4 Synchronous Dynamic RAM.** DDR4 RAM is a type of DRAM that has a high-bandwidth interface and is the successor to its previous DDR2 and DDR3 versions. DDR4 RAM allows for lower voltage requirements and higher module density. It is coupled with higher data rate transfer speeds and allows for dual in-line memory modules (DIMMS) up to 64 GB.
- **Rambus Dynamic RAM.** DRDRAM is a memory subsystem that promised to transfer up to 1.6 billion bytes per second. The subsystem consists of RAM, the RAM controller, the bus that connects RAM to the microprocessor and devices in the computer that use it.
- **Read-only memory.** ROM is a type of computer storage containing nonvolatile, permanent data that, normally, can only be read and not written to. ROM contains the programming that enables a computer to start up or regenerate each time it is turned on.
- **Programmable ROM.** PROM is ROM that can be modified once by a user. It enables a user to tailor a microcode program using a special machine called a PROM programmer.
- **Erasable PROM.** EPROM is programmable read-only memory PROM that can be erased and re-used. Erasure is caused by shining an intense ultraviolet light through a window designed into the memory chip.
- **Electrically erasable PROM.** EEPROM is a user-modifiable ROM that can be erased and reprogrammed repeatedly through the application of higher than normal electrical voltage. Unlike EPROM chips, EEPROMs do not need to be removed from the computer to be modified. However, an EEPROM chip must be erased and reprogrammed in its entirety, not selectively.
- **Virtual memory**. A memory management technique where secondary memory can be used as if it were a part of the main memory. Virtual memory uses

hardware and software to enable a computer to compensate for physical memory shortages by temporarily transferring data from RAM to disk storage.

3. Auxiliary Memory

An Auxiliary memory is referred to as the lowest-cost, highest-space, and slowest-approach storage in a computer system. It is where programs and information are preserved for long-term storage or when not in direct use. The most typical auxiliary memory devices used in computer systems are magnetic disks and tapes.

3.1. Magnetic Disks

A magnetic disk is a round plate generated of metal or plastic coated with magnetized material. There are both sides of the disk are used and multiple disks can be stacked on one spindle with read/write heads accessible on each surface.

All disks revolve together at high speed and are not stopped or initiated for access purposes. Bits are saved in the magnetized surface in marks along concentric circles known as tracks. The tracks are frequently divided into areas known as sectors.

In this system, the lowest quantity of data that can be sent is a sector. The subdivision of one disk surface into tracks and sectors is displayed in the figure.

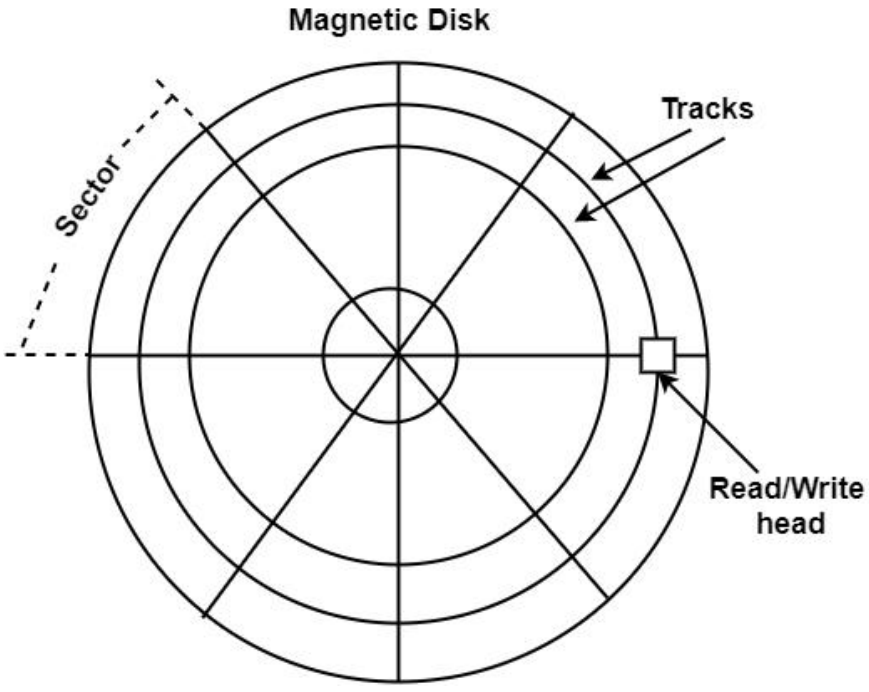


Figure 3.1. Magnetic Disk

3.2. Magnetic Tape

Magnetic tape transport includes the robotic, mechanical, and electronic components to support the methods and control structure for a magnetic tape unit. The tape is a layer of plastic coated with a magnetic documentation medium.

Bits are listed as a magnetic stain on the tape along various tracks. There are seven or nine bits are recorded together to form a character together with a parity bit. Read/write heads are mounted one in each track therefore that information can be recorded and read as a series of characters.

Magnetic tape units can be stopped, initiated to move forward, or in the opposite, or it can be reversed. However, they cannot be initiated or stopped fast enough between single characters. For this reason, data is recorded in blocks defined as records. Gaps of unrecorded tape are added between records where the tape can be stopped.

The tape begins affecting while in a gap and achieves its permanent speed by the time it arrives at the next record. Each record on tape has a recognition bit design at the starting and end. By reading the bit design at the starting, the tape control recognizes the data number.

4. Properties of Memory Heirarchy:

There are three important properties for maintaining consistency in the memory hierarchy these three properties are Inclusion, Coherence, and Locality.

Information stored in a memory hierarchy satisfies all the above properties

A. the Inclusion property, it states as: (M1CM2C...)

The set inclusive implies that all the information's are originally stored in the outermost level Mi and the information word found in Mn then copies of the same word can also be found in all the upper levels Mi+1, Mi+2.

- Access by word(4 bytes) from a cache block of 32 bytes, such as block A
- Access by block (32 bytes) from a memory page of 32 blocks or 1 KB such as block B from page B.
- Access by page 1 KB from a file consisting of many pages like A & B in segment F
- Segment Transfer with different No. of pages

B. Coherence Property

The coherence property requires that copies of the same information item at successive memory level should be consistent. If a word is modified in the cache copies of that word must be updated immediately at the higher levels.

There are two strategies for maintaining coherence in a memory hierarchy. The first method is,

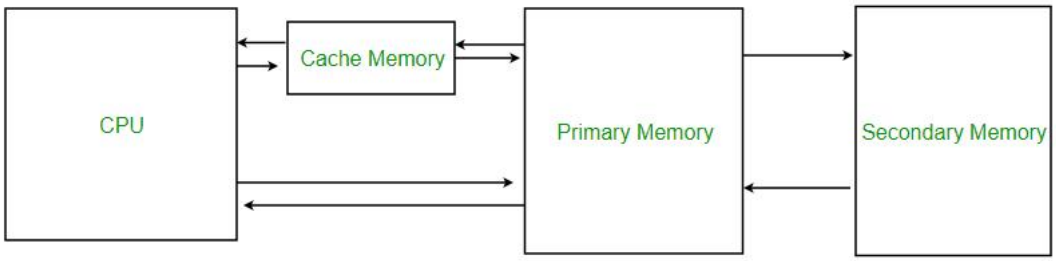
- Write through which demand immediate update through broadcasting Mi+1 level of memory if a word is modified in Mi.
- Write back – the second method is written which delays the update in Mi.

C. Locality of references

- Temporal Locality - Recently referenced items like instructions or data are likely to be referred again shortly such as iterative loops process stacks, temporary variables or subroutines.
- Spatial Locality - This refers to the tendency for a process to access items whose addresses are nearer to another for example operations on tables or arrays involve access of a certain clustered area in the address space.
- Sequential Locality - In typical programs the execution of instructions follows a sequential order unless branch instructions create out of order execution. The ratio of in order execution to out of order execution is roughly 5:1 in ordinary programs.

5. Cache Memory & Cache Memory Organization

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 more economical than CPU registers. 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 that 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.



6. Cache Memory 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 the 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(H) = hit / (hit + miss) = no. of hits/total accesses

Miss ratio = miss / (hit + miss) = no. of miss/total accesses = 1 - hit ratio(H)

We can improve Cache performance using higher cache block size, and higher associativity, reduce miss rate, reduce miss penalty, and reduce the time to hit in the cache.

7. Cache Memory Mapping Algorithms:

Cache Mapping: There are three different types of mapping used for the purpose of cache memory which is as follows: Direct mapping, Associative mapping, and Set-Associative mapping. These are explained below.

7.1. 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 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 modulo m

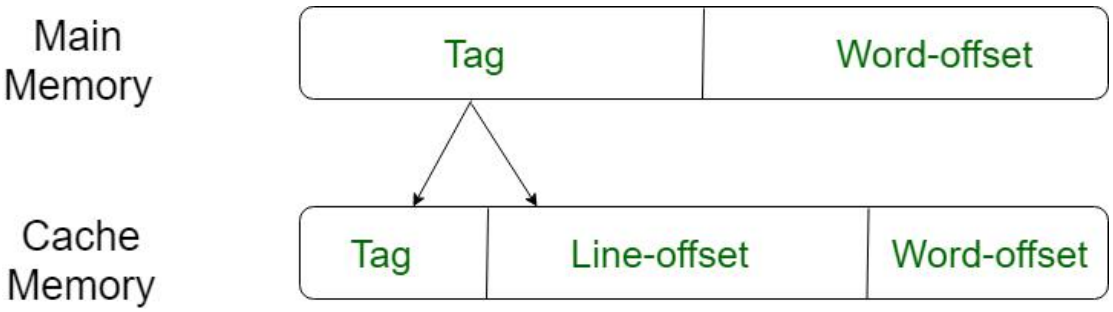
where

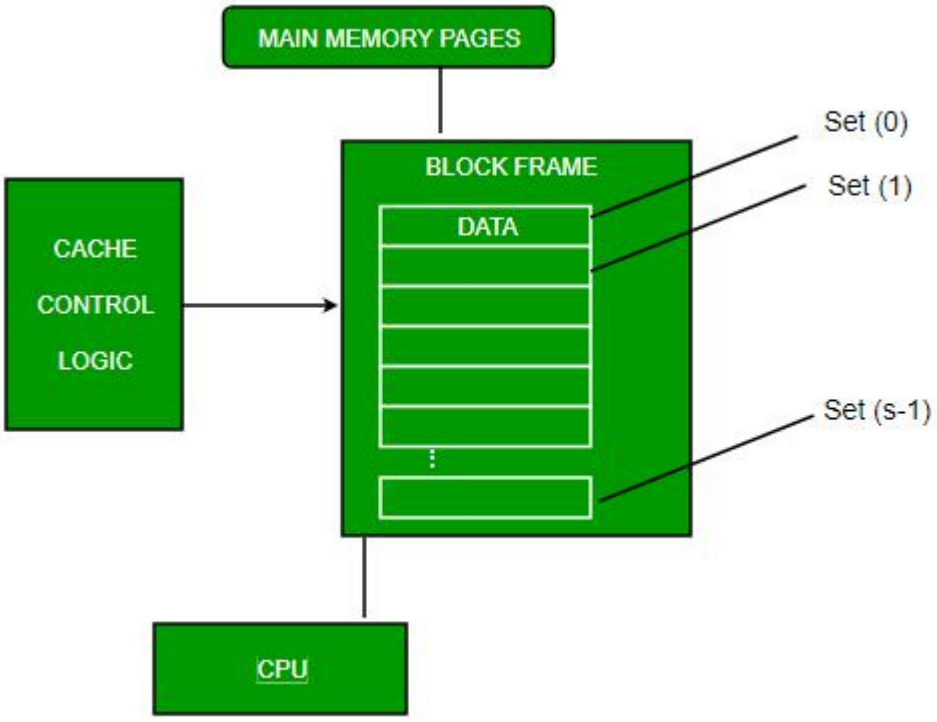
i=cache line number

j= main memory block number

m=number of lines in the cache

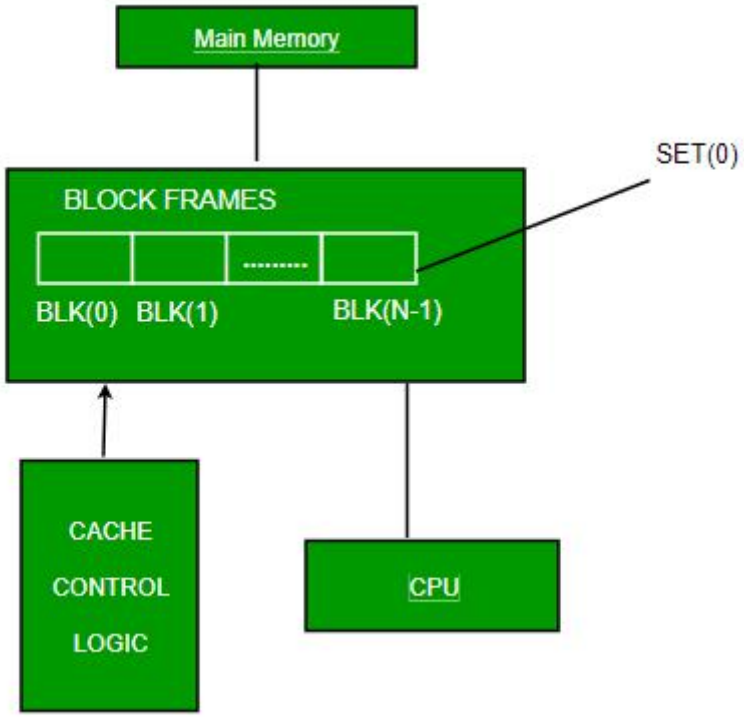
1) For purposes of cache access, each main memory address can be viewed as consisting of three fields. The least significant w bits identify a unique word or byte within a block of main memory. In most contemporary machines, the address is at the byte level. The remaining s bits specify one of the 2s blocks of main memory. The cache logic interprets these s bits as a tag of s-r bits (most significant portion) and a line field of r bits. This latter field identifies one of the m=2r lines of the cache. Line offset is index bits in the direct mapping.





7.2. 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. In associative mapping the index bits are zero.



7.3. 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 set associative mapping the index bits are given by the set offset bits. 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 mod v

where

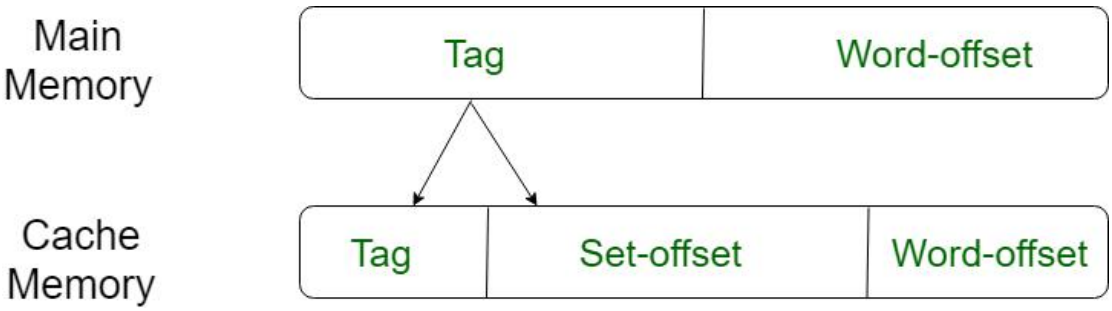
i=cache set number

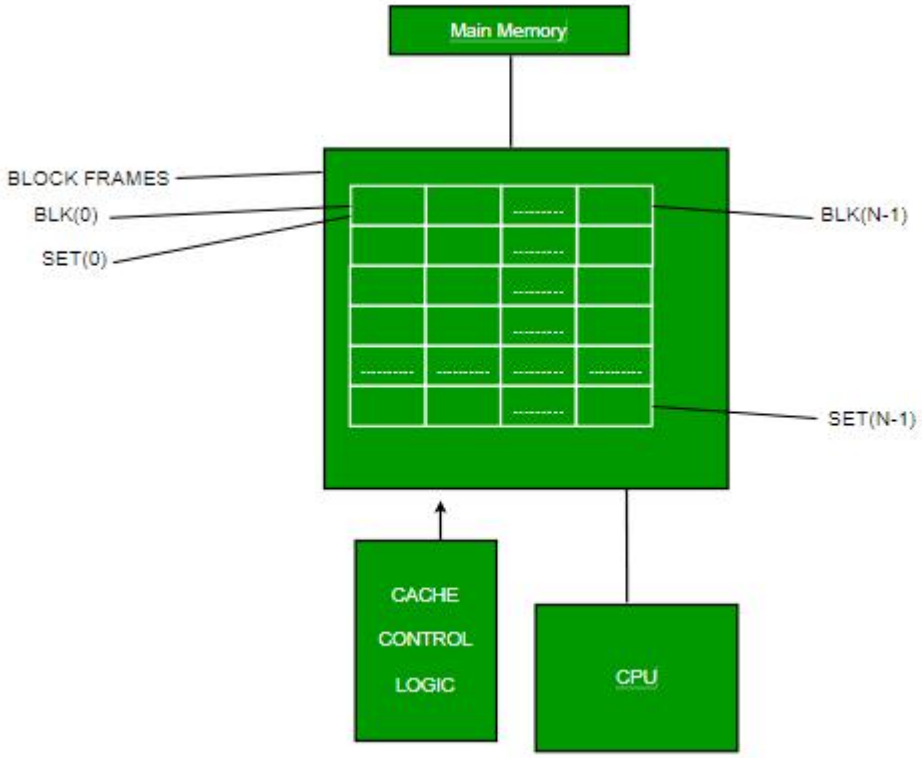
j=main memory block number

v=number of sets

m=number of lines in the cache number of sets

k=number of lines in each set





8. Cache Memory Optimization

The following are the activity for optimizing cache performance:

1) **Reducing the hit time** – Small and simple first-level caches and way-prediction. Both techniques also generally decrease power consumption.
2) **Increasing cache bandwidth** – Pipelined caches, multi-banked caches, and non-blocking caches. These techniques have varying impacts on power consumption.
3) **Reducing the miss penalty** – Critical word first and merging write buffers. These optimization have little impact on power.
4) **Reducing the miss rate** – Compiler optimization. Obviously any improvement at compile time improves power consumption.
5) **Reducing the miss penalty or miss rate via parallelism** – Hardware prefetching and compiler prefetching. These optimization generally increase power consumption, primarily due to prefetched data that are unused.

9. Virtual Memory

Virtual Memory is a storage scheme that provides user an illusion of having a very big main memory. This is done by treating a part of secondary memory as the main memory.

In this scheme, User can load the bigger size processes than the available main memory by having the illusion that the memory is available to load the process.

Instead of loading one big process in the main memory, the Operating System loads the different parts of more than one process in the main memory.

By doing this, the degree of multiprogramming will be increased and therefore, the CPU utilization will also be increased.

9.1. How Virtual Memory Works?

In modern word, virtual memory has become quite common these days. In this scheme, whenever some pages needs to be loaded in the main memory for the execution and the memory is not available for those many pages, then in that case, instead of stopping the pages from entering in the main memory, the OS search for the RAM area that are least used in the recent times or that are not referenced and copy that into the secondary memory to make the space for the new pages in the main memory.

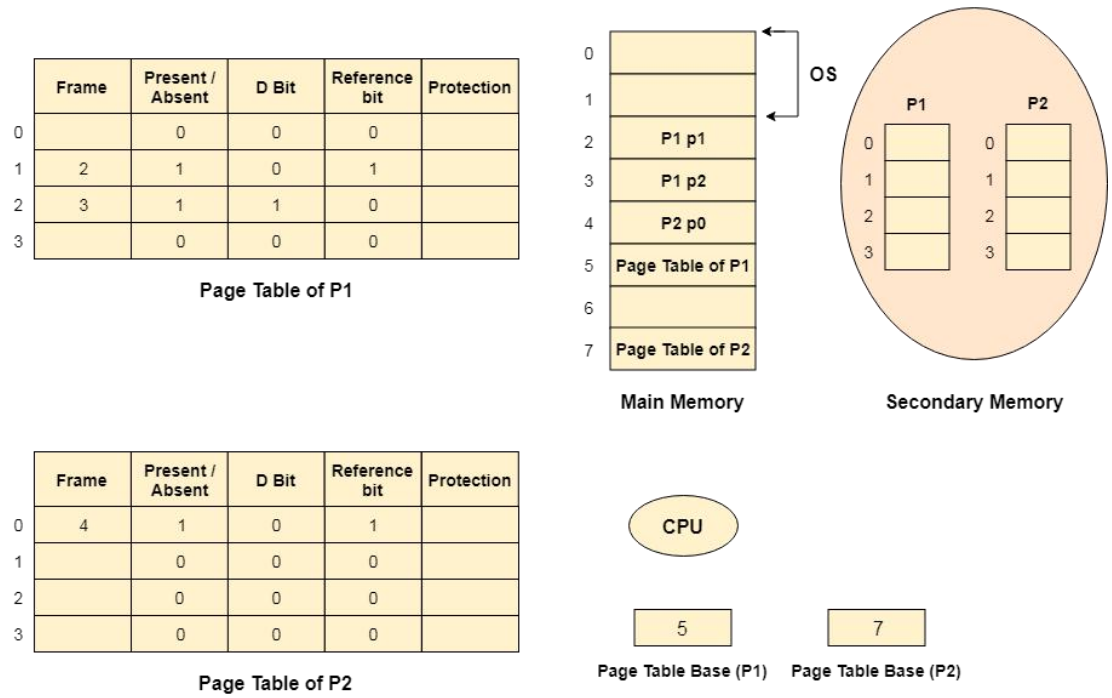
Since all this procedure happens automatically, therefore it makes the computer feel like it is having the unlimited RAM.

9.2. Snapshot of a virtual memory management system

Let us assume 2 processes, P1 and P2, contains 4 pages each. Each page size is 1 KB. The main memory contains 8 frame of 1 KB each. The OS resides in the first two partitions. In the third partition, 1st page of P1 is stored and the other frames are also shown as filled with the different pages of processes in the main memory.

The page tables of both the pages are 1 KB size each and therefore they can be fit in one frame each. The page tables of both the processes contain various information that is also shown in the image.

The CPU contains a register which contains the base address of page table that is 5 in the case of P1 and 7 in the case of P2. This page table base address will be added to the page number of the Logical address when it comes to accessing the actual corresponding entry.



9.3. Advantages of Virtual Memory

- The degree of Multiprogramming will be increased.
- User can run large application with less real RAM.
- There is no need to buy more memory RAMs.

9.4. Disadvantages of Virtual Memory

- The system becomes slower since swapping takes time.
- It takes more time in switching between applications.
- The user will have the lesser hard disk space for its use.

10. Demand Paging

Demand Paging is a popular method of virtual memory management. In demand paging, the pages of a process which are least used, get stored in the secondary memory.

A page is copied to the main memory when its demand is made or page fault occurs. There are various page replacement algorithms which are used to determine the pages which will be replaced. We will discuss each one of them later in detail.

11. Page Replacement Algorithms

A virtual memory organization is a consolidation of hardware and software systems. It can make efficient utilization of memory space all the software operations are handled by the memory management software.

The hardware mapping system and the memory management software together form the structure of virtual memory.

When the program implementation starts, one or more pages are transferred into the main memory and the page table is set to denote their location. The program is implemented from the main memory just before a reference is created for a page that is not in memory. This event is defined as a page fault.

When a page fault appears, the program that is directly in execution is stopped just before the required page is transferred into the main memory. Because the act of loading a page from auxiliary memory to main memory is an I/O operation, the operating framework creates this function for the I/O processor.

In this interval, control is moved to the next program in the main memory that is waiting to be prepared in the CPU. Soon after the memory block is assigned and then moved, the suspended program can resume execution.

If the main memory is full, a new page cannot be moved in. Therefore, it is important to remove a page from a memory block to hold the new page. The decision of removing specific pages from memory is determined by the replacement algorithm.

There are two common replacement algorithms used are the first-in, first-out (FIFO) and least recently used (LRU).

The FIFO algorithm chooses to replace the page that has been in memory for the highest time. Every time a page is weighted into memory, its identification number is pushed into a FIFO stack.

FIFO will be complete whenever memory has no more null blocks. When a new page should be loaded, the page least currently transports in is removed. The page to be removed is simply determined because its identification number is at the high of the FIFO stack.

The FIFO replacement policy has the benefit of being simple to execute. It has the drawback that under specific circumstances pages are removed and loaded from memory too frequently.

The LRU policy is more complex to execute but has been more interesting on the presumption that the least recently used page is an excellent applicant for removal than the least recently loaded page as in FIFO. The LRU algorithm can be executed by relating a counter with each page that is in the main memory.

When a page is referenced, its associated counter is set to zero. At permanent intervals of time, the counters related to all pages directly in memory are incremented by 1.

The least recently used page is the page with the largest count. The counters are known as aging registers, as their count denotes their age, that is, how long ago their related pages have been referenced.

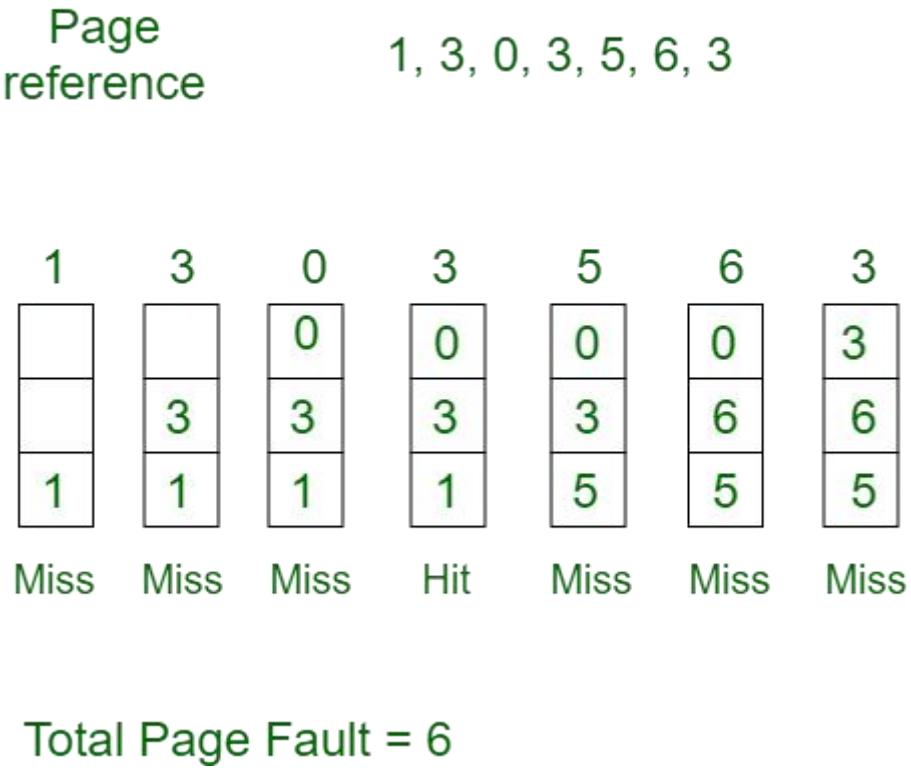
In an operating system that uses paging for memory management, a page replacement algorithm is needed to decide which page needs to be replaced when a new page comes in.

Page Fault: A page fault happens when a running program accesses a memory page that is mapped into the virtual address space but not loaded in physical memory. Since actual physical memory is much smaller than virtual memory, page faults happen. In case of a page fault, Operating System might have to replace one of the existing pages with the newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce the number of page faults.

Page Replacement Algorithms:

1) **First In First Out (FIFO)**: This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

Example 1: Consider page reference string 1, 3, 0, 3, 5, 6, 3 with 3 page frames.Find the number of page faults.



Initially, all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots —> **3 Page Faults**.

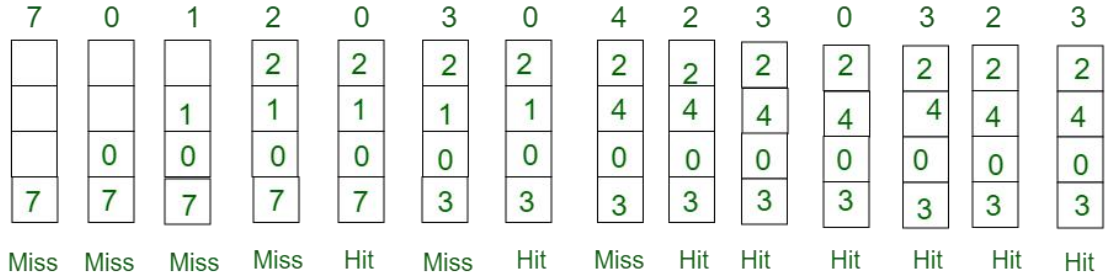
when 3 comes, it is already in memory so —> **0 Page Faults**. Then 5 comes, it is not available in memory so it replaces the oldest page slot i.e 1. —>**1 Page Fault**. 6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —>**1 Page Fault**. Finally, when 3 come it is not available so it replaces 0 **1 page fault**.

Belady's anomaly proves that it is possible to have more page faults when increasing the number of page frames while using the First in First Out (FIFO) page replacement algorithm. For example, if we consider reference strings 3, 2, 1, 0, 3, 2, 4, 3, 2, 1, 0, 4, and 3 slots, we get 9 total page faults, but if we increase slots to 4, we get 10-page faults.

2) **Optimal Page replacement**: In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.

Example-2: Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3 with 4 page frame. Find number of page fault.

Page reference 7,0,1,2,0,3,0,4,2,3,0,3,2,3 No. of Page frame - 4



Total Page Fault = 6

Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**. 0 is already there so —> **0 Page fault**. when 3 came it will take the place of 7 because it is not used for the longest duration of time in the future.—>**1 Page fault**. 0 is already there so —> **0 Page fault**. 4 will takes place of 1 —> **1 Page Fault**. Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

Optimal page replacement is perfect, but not possible in practice as the operating system cannot know future requests. The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analyzed against it.

3) **Least Recently Used**: In this algorithm, page will be replaced which is least recently used.

Example-3: Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3 with 4 page frames. Find number of page faults.

Page reference 7,0,1,2,0,3,0,4,2,3,0,3,2,3 No. of Page frame - 4

Total Page Fault = 6

Here LRU has same number of page fault as optimal but it may differ according to question.

Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> 4 Page faults

0 is already their so —> **0 Page fault**. when 3 came it will take the place of 7 because it is least recently used —>**1 Page fault**

0 is already in memory so —> **0 Page fault**.

4 will takes place of 1 —> **1 Page Fault**

Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

4) **Most Recently Used (MRU)**: In this algorithm, page will be replaced which has been used recently. Belady's anomaly can occur in this algorithm.