TOPIC1: Interleaved Memory

What is Interleaved Memory?

It is a technique for compensating the relatively slow speed of DRAM(Dynamic RAM). In this technique, the main memory is divided into memory banks which can be accessed individually without any dependency on the other.

For example: If we have 4 memory banks(4-way Interleaved memory), with each containing 256 bytes, then, the Block Oriented scheme(no interleaving), will assign virtual address 0 to 255 to the first bank, 256 to 511 to the second bank. But in Interleaved memory, virtual address 0 will be with the first bank, 1 with the second memory bank, 2 with the third bank and 3 with the fourt, and then 4 with the first memory bank again.

Hence, CPU can access alternate sections immediately without waiting for memory to be cached. There are multiple memory banks which take turns for supply of data.

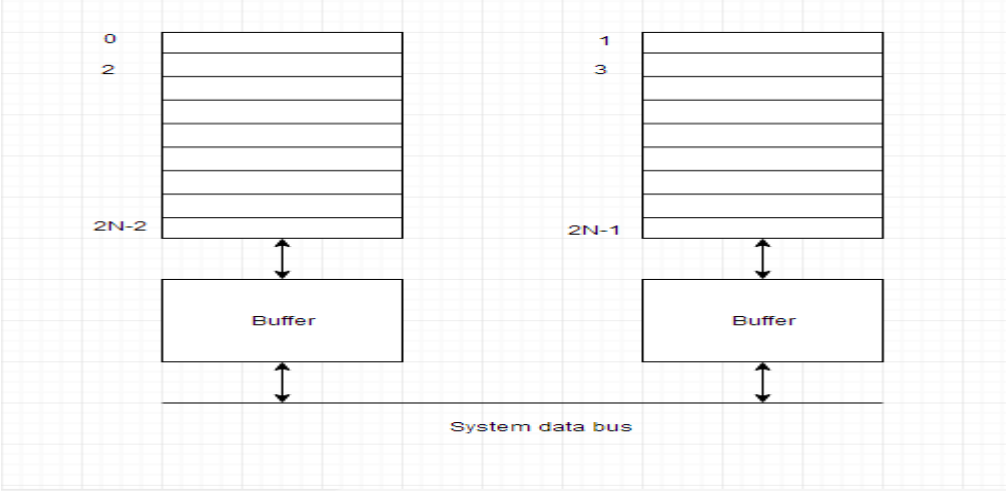
Memory interleaving is a technique for increasing memory speed. It is a process that makes the system more efficient, fast and reliable.

For example: In the above example of 4 memory banks, data with virtual address 0, 1, 2 and 3 can be accessed simultaneously as they reside in spearate memory banks, hence we do not have to wait for completion of a data fetch, to begin with the next.

An interleaved memory with n banks is said to be **n-way interleaved**. In an interleaved memory system, there are still **two banks of DRAM** but logically the system seems one bank of memory that is twice as large.

In the interleaved bank representation below with 2 memory banks, the first long word of bank 0 is floowed by that of bank 1, which is followed by the second long word of bank 0, which is followed by the second long word of bank 1 and so on.

The following figure shows the organization of two physical banks of n long words. All even long words of logical bank are located in physical bank 0 and all odd long words are located in physical bank 1.



Types of Interleaving

There are two methods for interleaving a memory:

- 2-Way Interleaved

Two memory blocks are accessed at same time for writing and reading operations.

- 4-Way Interleaved

Four memory blocks are accessed at the same time

TOPIC2: Memory Hierarchy

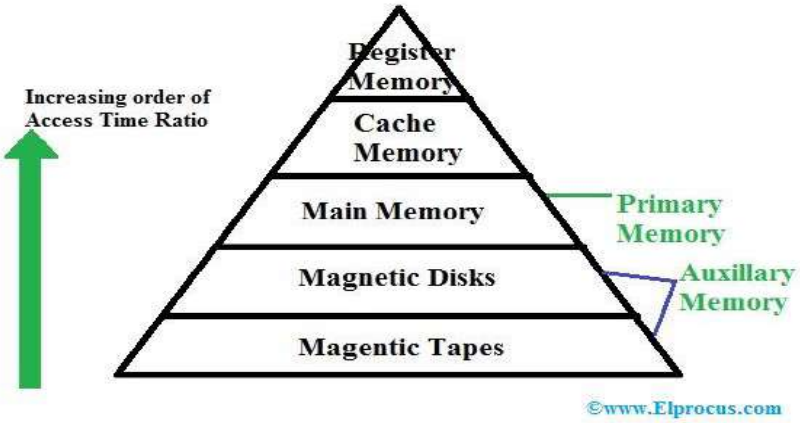
What is Memory Hierarchy?

The memory in a computer can be divided into five hierarchies based on the speed as well as use. The processor can move from one level to another based on its requirements. The five hierarchies in the memory are registers, cache, main memory, magnetic discs, and magnetic tapes. The first three hierarchies are volatile memories which mean when there is no power, and then automatically they lose their stored data. Whereas the last two hierarchies are not volatile which means they store the data permanently.

A memory element is the set of storage devices which stores the binary data in the type of bits. In general, the storage of memory can be classified into two categories such as volatile as well as non- volatile.

Memory Hierarchy in Computer Architecture

The **memory hierarchy design** in a computer system mainly includes different storage devices. Most of the computers were inbuilt with extra storage to run more powerfully beyond the main memory capacity. The following **memory hierarchy diagram** is a hierarchical pyramid for computer memory. The designing of the memory hierarchy is divided into two types such as primary (Internal) memory and secondary (External) memory.



Memory Hierarchy

1.Primary Memory

The primary memory is also known as internal memory, and this is accessible by the processor straightly. This memory includes main, cache, as well as CPU registers.

2.Secondary Memory

The secondary memory is also known as external memory, and this is accessible by the processor through an input/output module. This memory includes an optical disk, magnetic disk, and magnetic tape.

Characteristics of Memory Hierarchy

The memory hierarchy characteristics mainly include the following.

- **Performance**

Previously, the designing of a computer system was done without memory hierarchy, and the speed gap among the main memory as well as the CPU registers enhances because of the huge disparity in access time, which will cause the lower performance of the system. So, the enhancement was mandatory. The enhancement of this was designed in the memory hierarchy model due to the system's performance increase.

- **Ability**

The ability of the memory hierarchy is the total amount of data the memory can store. Because whenever we shift from top to bottom inside the memory hierarchy, then the capacity will increase.

- **Access Time**

The access time in the memory hierarchy is the interval of the time among the data availability as well as request to read or write. Because whenever we shift from top to bottom inside the memory hierarchy, then the access time will increase

- **Cost per bit**

When we shift from bottom to top inside the memory hierarchy, then the cost for each bit will increase which means an internal Memory is expensive compared with external memory.

- **Memory Hierarchy Design**

The memory hierarchy in computers mainly includes the following.

A.Registers

Usually, the register is a static RAM or SRAM in the processor of the computer which is used for holding the data word which is typically 64 or 128 bits. The program counter register is the most important as well as found in all the processors. Most of the processors use a status word register as well as an accumulator. A status word register is used for decision making, and the accumulator is used to store the data like mathematical operation. Usually, computers like **complex instruction set computers** have so many registers for accepting main memory, and **RISC- reduced instruction set** computers have more registers.

B.Cache Memory

Cache memory can also be found in the processor, however rarely it may be another **IC (integrated circuit)** which is separated into levels. The cache holds the chunk of data which are frequently used from main memory. When the processor has a single core then it will have two (or) more cache levels rarely. Present multi-core processors will be having three, 2-levels for each one core, and one level is shared.

C.Main Memory

The main memory in the computer is nothing but, the memory unit in the CPU that communicates directly. It is the main storage unit of the computer. This memory is fast as well as large memory used for storing the data throughout the operations of the computer. This memory is made up of RAM as well as ROM.

D.Magnetic Disks

The magnetic disks in the computer are circular plates fabricated of plastic otherwise metal by magnetized material. Frequently, two faces of the disk are utilized as well as many disks may be stacked on one spindle by read or write heads obtainable on every plane. All the disks in computer turn jointly at high speed. The tracks in the computer are nothing but bits which are stored within the magnetized plane in spots next to concentric circles. These are usually separated into sections which are named as sectors.

E.Magnetic Tape

This tape is a normal magnetic recording which is designed with a slender magnetizable covering on an extended, plastic film of the thin strip. This is mainly used to back up huge data. Whenever the computer requires to access a strip, first it will mount to access the data. Once the data is allowed, then it will be unmounted. The access time of memory will be slower within magnetic strip as well as it will take a few minutes for accessing a strip.

Advantages of Memory Hierarchy

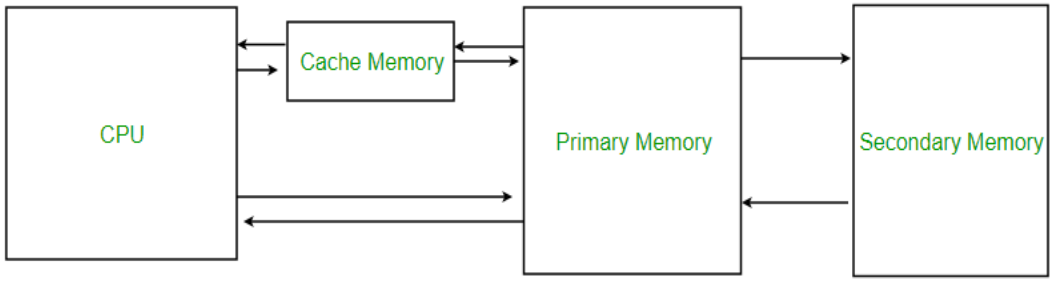
The need for a memory hierarchy includes the following.

- Memory distributing is simple and economical
- Removes external destruction
- Data can be spread all over
- Permits demand paging & pre-paging
- Swapping will be more proficient

TOPIC3: Cache Memory

Cache Memory in Computer Organization

Cache Memory is a special very high-speed memory. It is used to speed up and synchronizing with high-speed CPU. Cache memory is costlier than main memory or disk memory but 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 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.



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

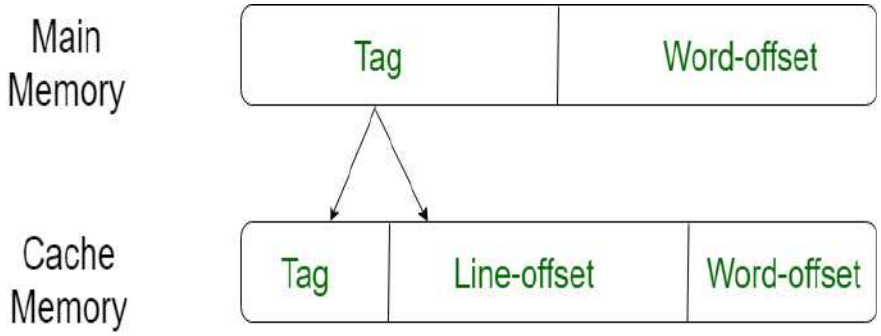
We can improve Cache performance using higher cache block size, higher associativity, reduce miss rate, reduce miss penalty, and reduce Reduce the time to hit in the cache.

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

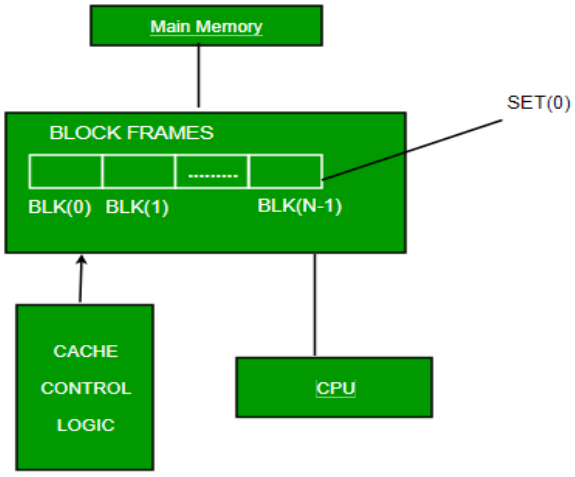
1. **Direct Mapping –**
   The simplest technique, known as direct mapping, maps each block of main memory into only one possible cache line. or
   In Direct mapping, assigne each memory block to a specific line in the cache. If a line is previously taken up by a memory block when a new block needs to 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.
2. i = j modulo m
3. where
4. i=cache line number
5. j= main memory block number

   m=number of lines in the cache

   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 2^s 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 $m=2^r$ lines of the cache.



6. **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.



7. **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 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

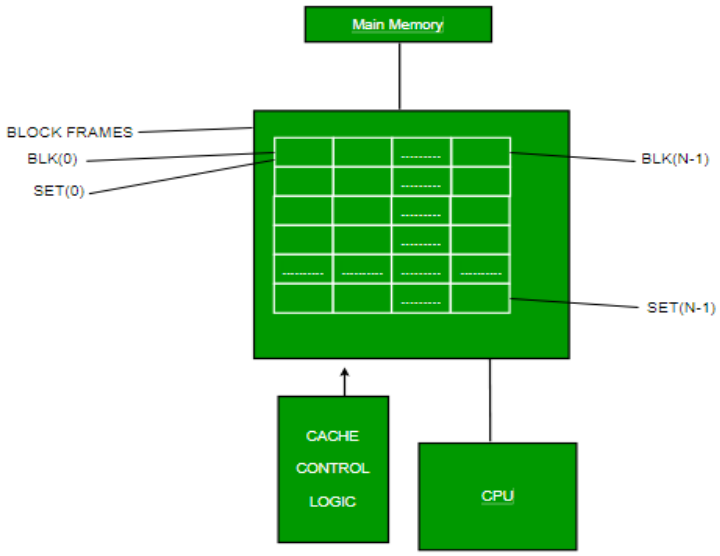
   i=cache set number

   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



Application of Cache Memory –

1. Usually, the cache memory can store a reasonable number of blocks at any given time, but this number is small compared to the total number of blocks in the main memory.
2. The correspondence between the main memory blocks and those in the cache is specified by a mapping function.

Types of Cache –

- **Primary Cache –**
  A primary cache is always located on the processor chip. This cache is small and its access time is comparable to that of processor registers.
- **Secondary Cache –**
  Secondary cache is placed between the primary cache and the rest of the memory. It is referred to as the level 2 (L2) cache. Often, the Level 2 cache is also housed on the processor chip.

Locality of reference –

Since size of cache memory is less as compared to main memory. So to check which part of main memory should be given priority and loaded in cache is decided based on locality of reference.

Types of Locality of reference

5. **Spatial Locality of reference**
   This says that there is a chance that element will be present in the close proximity to the reference point and next time if again searched then more close proximity to the point of reference.
6. **Temporal Locality of reference**
   In this Least recently used algorithm will be used. Whenever there is page fault occurs within a word will not only load word in main memory but complete page fault will be loaded because spatial locality of reference rule says that if you are referring any word next word will be referred in its register that's why we load complete page table so the complete block will be loaded.

TOPIC4 : replacement algorithms

In computing, cache algorithms (also frequently called cache replacement algorithms or cache replacement policies) are optimizing instructions, or algorithms, that a computer program or a hardware-maintained structure can utilize in order to manage a cache of information stored on the computer. Caching improves performance by keeping recent or often-used data items in memory locations that are faster or computationally cheaper to access than normal memory stores. When the cache is full, the algorithm must choose which items to discard to make room for the new ones.

First in first out (FIFO)

Using this algorithm the cache behaves in the same way as a FIFO queue. The cache evicts the blocks in the order they were added, without any regard to how often or how many times they were accessed before.

Last in first out (LIFO) or First in last out (FILO)

Using this algorithm the cache behaves in the same way as a stack and exact opposite way as a FIFO queue. The cache evicts the block added most recently first without any regard to how often or how many times it was accessed before.

Least recently used (LRU)

Discards the least recently used items first. This algorithm requires keeping track of what was used when, which is expensive if one wants to make sure the algorithm always discards the least recently used item. General implementations of this technique require keeping "age bits" for cache-lines and track the "Least Recently Used" cache-line based on age-bits. In such an implementation, every time a cache-line is used, the age of all other cache-lines changes.

Time aware least recently used (TLRU)

The Time aware Least Recently Used (TLRU) is a variant of LRU designed for the situation where the stored contents in cache have a valid life time. The algorithm is suitable in network cache applications, such as Information-centric networking (ICN), Content Delivery Networks (CDNs) and distributed networks in general. TLRU introduces a new term: TTU (Time to Use). TTU is a time stamp of a content/page which stipulates the usability time for the content based on the locality of the content and the content publisher announcement. Owing to this locality based time stamp, TTU provides more control to the local administrator to regulate in network storage. In the TLRU algorithm, when a piece of content arrives, a cache node calculates the local TTU value based on the TTU value assigned by the content publisher. The local TTU value is calculated by using a locally defined function.

Once the local TTU value is calculated the replacement of content is performed on a subset of the total content stored in cache node. The TLRU ensures that less popular and small life content should be replaced with the incoming content.

Most recently used (MRU)

Discards, in contrast to LRU, the most recently used items first. In findings presented at the 11th VLDB conference, Chou and DeWitt noted that "When a file is being repeatedly scanned in a [Looping Sequential] reference pattern, MRU is the best replacement algorithm."[6] Subsequently, other researchers presenting at the 22nd VLDB conference noted that for random access patterns and repeated scans over large datasets (sometimes known as cyclic access patterns) MRU cache algorithms have more hits than LRU due to their tendency to retain older data.[7] MRU algorithms are most useful in situations where the older an item is, the more likely it is to be accessed.