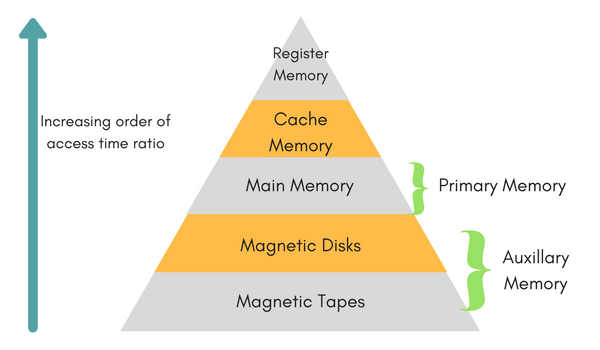
**CSC2231 COMPUTER ARCHITECTURE 4**

**Memory Organization in Computer Architecture**

* Memory Just like the human brain are used to store data's and instructions, it is the storage space in the computer, it is a collection of boxes divided into large and small parts called cells
* Memory Cell: It is a circuit that contains 1 bit of information (binary information), this can be set or Reset.
* Memory Word: This is a collection of bit to form a multi-bit unit that carries information, this can be 8 bits, 16 bits, 32 bits, 64 bits etc. but others are rare
* Memory capacity: how much memory is in the memory chip is called capacity this is expressed in byte, KB, MB, GB, TB etc
* Address: this is a memory module used to identify each location



**Memory Hierarchy**

**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.

**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**

* Performance
* Ability
* Access Time
* Cost per bit

**Performance:** 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.

**Characteristics of Memory Hierarchy**

* **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 Organization in Computer Architecture**

* Generally, memory/storage is classified into 2 categories
  + **Volatile Memory**
  + **Non-Volatile Memory**

**Volatile Memory:** This loses its data, when power is switched off

**Non-Volatile Memory:** This is a permanent storage and does not lose any data when power is switched off.

**REGISTER SET**

* Registers are essentially extremely fast memory locations within the CPU that are used to create and store the results of CPU operations and other calculations. Different computers have different register sets. They differ in the number of registers, register types, and the length of each register
* **Memory Access Registers:**
* Two registers are essential in memory write and read operations: the memory data register (MDR) and memory address register (MAR). The MDR and MAR are used exclusively by the CPU and are not directly accessible to programmers.
* In order to perform a write operation into a specified memory location, the MDR and MAR are used as follows:
* In order to perform a write operation into a specified memory location, the MDR and MAR are used as follows:

1. The word to be stored into the memory location is first loaded by the CPU into MDR.

2. The address of the location into which the word is to be stored is loaded by the CPU into a MAR.

3. A write signal is issued by the CPU.

* Similarly, to perform a memory read operation, the MDR and MAR are used as follows:

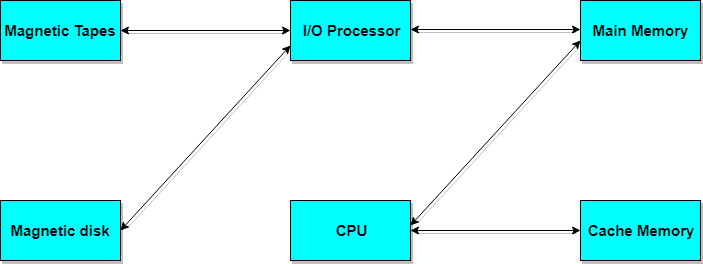
1. The address of the location from which the word is to be read is loaded into the MAR.

2. A read signal is issued by the CPU.

3. The required word will be loaded by the memory into the MDR ready for use by the CPU.

**Cache Memory**

* The **cache memory** is used to store program data which is currently being executed in the CPU. Approximate access time ratio between cache memory and main memory is about **1 to 7~10.**



**Memory Access Methods**

* Each memory type, is a collection of numerous memory locations. To access data from any memory, first it must be located and then the data is read from the memory location. Following are the methods to access information from memory locations:
* **Random Access**: Main memories are random access memories, in which each memory location has a unique address. Using this unique address any memory location can be reached in the same amount of time in any order.
* **Sequential Access**: This methods allows memory access in a sequence or in order.
* **Direct Access**: In this mode, information is stored in tracks, with each track having a separate read/write head.

**Cache Memory**

* The data or contents of the main memory that are used again and again by CPU, are stored in the cache memory so that we can easily access that data in shorter time.
* Whenever the CPU needs to access memory, it first checks the cache memory. If the data is not found in cache memory then the CPU moves onto the main memory. It also transfers block of recent data into the cache and keeps on deleting the old data in cache to accommodate the new one.

**Hit Ratio**

* The performance of cache memory is measured in terms of a quantity called **hit ratio**. When the CPU refers to memory and finds the word in cache it is said to produce a **hit**. If the word is not found in cache, it is in main memory then it counts as a **miss**.
* The ratio of the number of hits to the total CPU references to memory is called hit ratio
* Hit ratio = hit / (hit+miss)

**Characteristic of Cache Memory**

The basic is its fast access time,

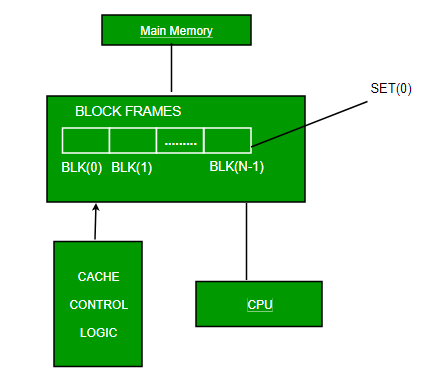
* Therefore, very little or no time must be wasted when searching the words in the cache
* The transformation of data from main memory to cache memory is referred to as a mapping process, there are three types of mapping:
* Associative mapping
* Direct mapping
* Set-associative mapping

**Associative mapping**

* Associative mapping: The fastest and most flexible cache organization uses an associative memory, the associative memory stores both the address and data of the memory word. This permits any location in cache to store any word from main memory
* The address value of 15 bits is shown as a five-digit octal number and its corresponding 12-bit word is shown as a four-digit octal number

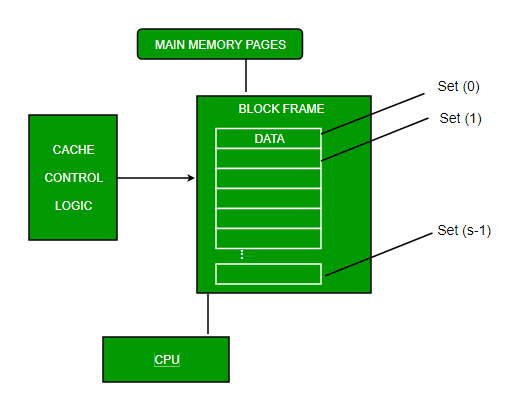


* A CPU address of 15 bits is places in the argument register and the associative memory is searched for a matching address
* If the address is found, the corresponding 12-bits data is read and sent to the CPU
* If not, the main memory is accessed for the word
* If the cache is full, an address-data pair must be displaced to make room for a pair that is needed and not presently in the cache



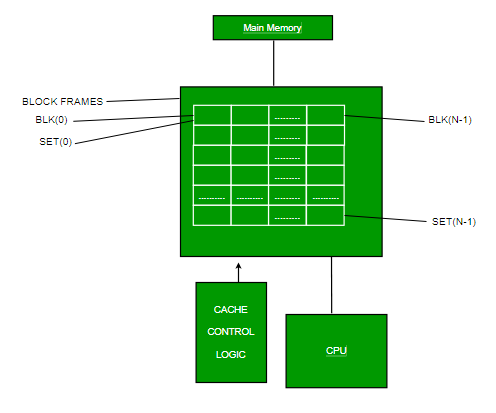
**Direct mapping**

* The simplest technique, known as direct mapping, maps each block of main memory into only one possible cache line.
* 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.



**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.



**Average memory access time**

* Average memory access time = % instructions \* (Hit\_time + instruction miss rate\*miss\_penality) + % data \* (Hit\_time + data miss rate\*miss\_penality)
* Assume 40% of the instructions are data accessing instruction.
* •Let a hit take 1 clock cycle and the miss penalty is 100 clock cycle
* •Assume instruction miss rate is 4% and data access miss rate is 12%, what is the average memory access time?

60% \* (1 + 4% \* 100) +

40% \* (1 + 12% \* 100)

= 0.6 \* (5) + 0.4 \* (13)

= 8.2 (clock cycle)

**Cache Operation:**

* It is based on the principle of locality of reference. There are two ways with which data or instruction is fetched from main memory and get stored in cache memory. These two ways are the following:
* **Temporal Locality –**  
  Temporal locality means current data or instruction that is being fetched may be needed soon. So we should store that data or instruction in the cache memory so that we can avoid again searching in main memory for the same data.
* When CPU accesses the current main memory location for reading required data or instruction, it also gets stored in the cache memory which is based on the fact that same data or instruction may be needed in near future. This is known as temporal locality. If some data is referenced, then there is a high probability that it will be referenced again in the near future.
* Spatial locality means instruction or data near to the current memory location that is being fetched, may be needed soon in the near future. This is slightly different from the temporal locality. Here we are talking about nearly located memory locations while in temporal locality we were talking about the actual memory location that was being fetched.
* **Temporal** **locality:** states that recently accessed items are likely to be accessed in the near future.
* **Spatial locality:** says that items whose addresses are near one another tend to be referenced close together in time