Memory Hierarchy:

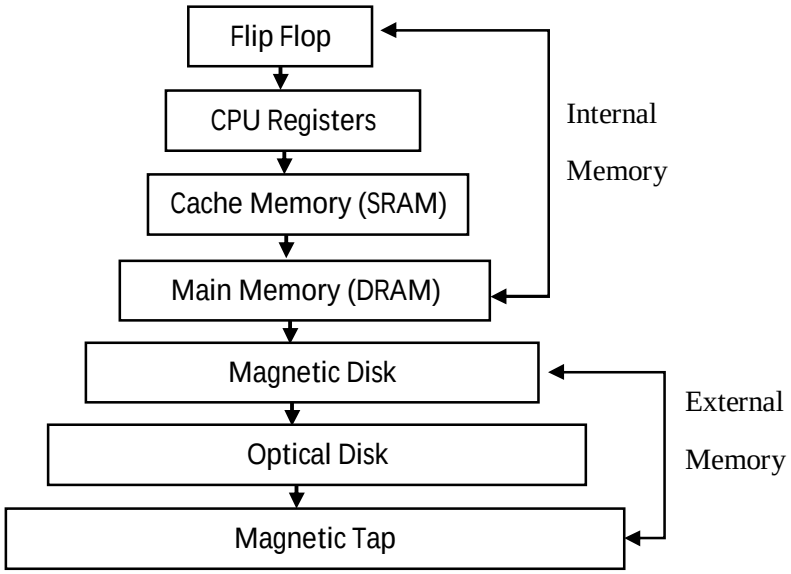


Figure 2: Memory Hierarchy

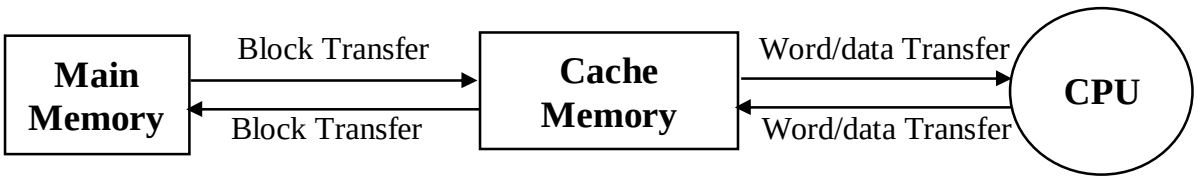
Flip Flop: Can store one bit of instruction to be added into the CPU registers sequentially.

Registers:

Registers are the fastest and smallest type of memory, located directly within the central processing unit (CPU). They store small amounts of data that are frequently accessed by the CPU. They also hold data, instructions, and control information that the CPU needs to perform operations. The storage capacity for registers ranges from 16 to 64 bits.

Cache memory:

Cache memory is a small, high-speed memory that acts as a buffer between the CPU and main memory (RAM), storing frequently accessed data and instructions for faster retrieval and improved overall system performance.



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**

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

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.

Cache memory performance is frequently measured in terms of a quantity called **Hit ratio.**

Hit ratio = hit / (hit + miss) = no. of hits/total accesses

Cache Size V/s Block Size:

In computing, cache size refers to the total amount of memory available for storing frequently accessed data, while block size (also known as cache line size) represents the amount of data transferred between the cache and main memory in a single operation.

Cache Size: The total amount of memory allocated for storing frequently accessed data and instructions. Caches are often organized in levels (L1, L2, L3), with L1 being the fastest and smallest and L3 being the slowest and largest. Larger cache size can improve performance by reducing the need to access slower main memory, but it also increases the cost and complexity of the system.

Cache sizes are typically measured in kilobytes (KB), megabytes (MB), or gigabytes (GB).

Block Size (Cache Line Size): The amount of data transferred between the cache and main memory in a single operation. A larger block size can lead to better utilization of the cache, as more data is transferred at once, but it can also increase the penalty of a cache miss (when the requested data is not in the cache).

The cache size is determined by the number of blocks it can hold and the block size. Common block sizes in modern CPUs are 64 bytes.

Main Memory (Primary Memory):

Primary memory is the volatile storing mechanism of a computer system. It may refer to random access memory (RAM), cache memory, or data buses, although it is often connected with a computer's RAM. Primary memory loads all running programmes, including the basic operating system (OS), user interface, and any user-installed and running software utility, as soon as a computer boots up. When a programme or application is launched in the main

memory, it interacts with the system processor to complete all application-specific tasks. Secondary memory is said to be slower than primary memory.

Magnetic Disks:

Magnetic disks, like HDDs, store data magnetically on spinning platters. They offer high storage capacity and support both sequential and random access. However, they are less durable and slower than solid-state drives (SSDs). Still used for bulk data storage in some systems, they provide cost-effective storage options for various applications.

Optical Disks:

Optical Disk is a storage medium that relies on laser technology to read and write data, in shape, it is a flat circular disk which is made up of polycarbonate or a similar material with a very shiny reflective layer on the surface. They are mainly used for sharing, storing and backup Data as they have a great life span and capacity compared to older technologies like floppy disks. CD-ROM, CD-R, CD-RW, DVD, and Blu-ray Disc (BD) are the examples of the optical disk.

Magnetic Tap:

Magnetic tapes are used to store data that is accessed infrequently. This data is usually archived data. It is also used to store data that is too large to be stored in the computer system. Magnetic tapes are similar to magnetic disks, but they use an electric current instead of a magnet to store data stored on them.

In magnetic tape only one side of the ribbon is used for storing data. It is a sequential memory that contains a thin plastic ribbon to store data and is coated by magnetic oxide. Data read/write speed is slower because of sequential access. It is highly reliable which requires magnetic tape drive writing and reading data. For example

1. **9-Track:** Recorded 8 data bits plus one parity bit across the tape, offering densities like 800, 1600, and 6250 bpi (bytes per inch).
2. **7-Track:** A predecessor to the 9-track, using a different track layout.

Write Policies:

1. Write through Method:

The simplest method is to update the main memory as well as Cache memory with every memory write operation when the cache memory is updated in parallel when it contains the word at the specified address. This can be known as the write-through method.

2. Write Back Method:

During a write operation, only the cache location is updated in the write-back method. Then, the location is marked by a flag so that it is later copied to the main memory when the word is removed from the cache. For the write-back method, the reason is that during the time a word remains in the cache, it can be updated multiple times. Thus, as long as the word remains in the cache, it does not matter if the copy in the main cache. This is only when the word is displaced from the cache which needs an exact copy that is rewritten into main memory.

Memory Interleaving:

As you know that cache memory is used as a buffer memory between processor and the main memory to bridge the difference between the processor speed and access time of the main memory. So, when processor requests a data item, it is first looked into the cache and if data item is not present in the cache (called cache miss), only then main memory is accessed to read the data item. To further enhance the performance of the computer system and to reduce the memory access time of the main memory, in case of cache miss, the concept of memory interleaving is used. Memory interleaving is of three type, viz. lower order memory interleaving, higher order memory interleaving and hybrid memory interleaving. we will discuss the lower order memory interleaving.

In memory interleaving technique, main memory is partitioned into n number of equal sized modules called as memory banks and technique is known as n-way memory interleaving. Where each memory module has its own memory address register, base register and instruction register, thus each memory bank can be accessed individually and simultaneously. Instructions of a process are stored in successive memory banks. So, in a single memory access time n successive instructions of the process can be accessed from n memory banks. For example, suppose main memory is divided into four modules or memory banks denoted as M1, M2, M3 and M4 then first n instructions of a process will be stored as: first instruction in M1, second instruction in M2, third instruction in M3, fourth instruction in M4 and again fifth instruction in M1 and so on.

When processor issues a memory fetch command during the execution of the program, memory access system creates n consecutive memory addresses and places them in respective memory address register of all memory banks in the right order. Instructions are read from all memory modules simultaneously and loads them into n instruction registers. Thus, each fetch for a new instruction results in the loading of n consecutive instructions in n instruction registers of the CPU, in the time of a single memory access. Figure 2 shows the structure of 4-way memory interleaving. The address is resolved by interpreting the least significant bits to select the memory module, and rest of the most significant bits are the address in the memory module.

For example, in an 8-bit address and 4-way memory interleaving, two least significant bits will be used for module selection and six most significant bits will be used as an address in the module.

8-bit address in 4-way memory interleaving

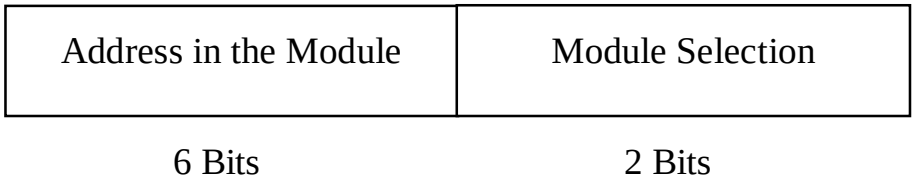


Figure 2: Address mapping for Memory interleaving

The following example demonstrates **how the main memory words be distributed to different interleaved memory modules**. For this example, only a four bit address of main memory is used.

Main Memory	
Address	**Data**
0000	10
0001	20
0010	30
0011	40
0100	50
0101	60
0110	80
0111	76
1000	46
1001	25
1010	58
1011	100
1100	23
1101	78
1110	25
1111	11

Module 00	
Address	**Data**
00	10
01	50
10	46
11	23

Module 01	
Address	**Data**
00	20
01	60
10	25
11	78

Module 10	
Address	**Data**
00	30
01	80
10	58
11	35

Module 11	
Address	**Data**
00	40
01	76
10	100
11	11

Figure 3: Example of Memory interleaving

Cache replacement algorithms:

Cache replacement algorithms, also known as cache replacement policies, are strategies used to determine which data to remove from a cache when it's full and a new piece of data needs to be stored. Common algorithms include **FIFO (First-In, First-Out), LRU (Least Recently Used), and LFU (Least Frequently Used).**