Table of Contents

[What is Memory 2](#_Toc135578050)

[Characteristics of Computer Memory 2](#_Toc135578051)

[Location 2](#_Toc135578052)

[Capacity 2](#_Toc135578053)

[Unit of Transfer 2](#_Toc135578054)

[Methods of Access 2](#_Toc135578055)

[Performance 3](#_Toc135578056)

[Physical Types 3](#_Toc135578057)

[The Memory Hierarchy 4](#_Toc135578058)

[Principle of Locality of reference 4](#_Toc135578059)

[Cache Memory Principles 5](#_Toc135578060)

[Cache Memory 5](#_Toc135578061)

[What happens when a processor wants to read a word from memory! 5](#_Toc135578062)

[Structure of Main Memory System 5](#_Toc135578063)

[Structure of Cache 5](#_Toc135578064)

[Contemporary Cache Structure 6](#_Toc135578065)

[Elements of Cache Design 7](#_Toc135578066)

[Cache Addresses 7](#_Toc135578067)

[Cache Size 7](#_Toc135578068)

[Mapping Function 8](#_Toc135578069)

[Replacing Algorithms 9](#_Toc135578070)

[Write Policy 10](#_Toc135578071)

# What is Memory

Memory is an electronic holding/storage space where a computer stores information for immediate use.

# Characteristics of Computer Memory

## Location

There are two types of memory: internal and external. Internal memory is local to the processor such as registers, RAM or cache. External memory includes peripheral storage devices like hard drives, optical disks, magnetic disks, tapes.

## Capacity

The amount of data a memory can hold is called its capacity and is measured in bytes. The basic unit of storage is a bit (0 or 1) and the smallest unit is a byte. The capacity of internal memory is expressed in bytes or words (which are normally 8, 16, or 32 bits) while external memory capacity is expressed in bytes.

## Unit of Transfer

The unit of transfer refers to the number of electrical lines in and out of the memory, which is normally equal to the length of a word but can be larger. Data can be transferred in blocks of words.

## Methods of Access

Sequential Access: When an access request is made, memory is searched linearly in a sequential fashion. The search is started from the first memory location and incremented by moving one step ahead at a time until the desired location is found. This is a simple but slow technique.

Random Access: Data present at any location can be accessed directly, and the time to access any memory location is the same. This is a faster way to retrieve the data, and RAM is an example of this type of access.

Direct Access: Memory is divided into blocks (multiple memory locations) used by magnetic and optical discs. Like Random access, blocks can be accessed randomly, so the block access time is the same. However, unlike Random Access, the time taken to access a memory location inside Block-I would be different compared to accessing a memory location inside Block-II because sequential access is required inside the blocks.

## Performance

Performance of a memory is a critical factor for users. The parameters for performance include access time (latency), memory cycle time, and transfer rate.

Access Time: Access time refers to the time taken by a computer's memory to retrieve or store data in response to a request from the processor. For random access memory (RAM), it is the time taken for a read or write operation, which starts from the moment an address is presented to the memory, and ends when the requested data is available for use. For non-Random Access Memory, it is the time taken to position the read-write mechanism at the desired location.

Memory cycle time is the time taken for an access time plus the wait time. Time taken by the memory to complete an operation in a cycle.

Transfer rate is the rate at which data is transferred in or out of the memory. For Random-access memory, it is calculated as 1/cycle time.

## Physical Types

Semiconductor memory: Volatile/Non-Volatile

* RAM (Random Access Memory) 🡪 Volatile
* ROM (Read-Only Memory) 🡪 Non-Volatile

Magnetic memory:

* Hard disk drive
* Magnetic tape
* Floppy disk

Optical memory:

* CD (Compact Disc)
* DVD (Digital Versatile Disc)
* Blu-ray Disc

# The Memory Hierarchy

## Principle of Locality of reference

The principle of locality of reference states that a processor tends to access the same set of memory locations repeatedly over a short period of time when executing a program that has arrays, loops, and instructions that repeat. This principle ensures that frequently accessed instructions are stored in faster memory, improving the hit ratio.

# Cache Memory Principles

## Cache Memory

Cache Memory is a type of computer memory that combines the advantages of two separate types of memory. It allows for the high speed accessed of data storage found in high-speed memory, while also providing the large memory size found in lower speed, less expensive memory.

Or

Cache memory is a fast and small memory that stores frequently accessed data to speed up the performance of a computer system by reducing the time needed to retrieve information from the main memory.

## What happens when a processor wants to read a word from memory!

When a processor needs to read memory, it first checks the cache to see if the data is available there. If the word is available, it is delivered to the processor immediately. If not, a block of words is read from the main memory and the required word is delivered to the processor from that block.

Cache Memory uses block of words instead of a single word because of the principle of locality of reference. This principle states that the same word may be referenced again, or a word adjacent to the current word may be referenced. So, having a block of words in the cache ensures that if one word is needed, other words in the block may also be needed, making retrieval much faster.

## Structure of Main Memory System

The main memory system is structured in a way that allows it to store a large amount of data. Each piece of data, called a "word," has a specific length measured in bits. The total number of words that can be stored is. The memory can also be organized into "blocks," where each block contains multiple words, typically K words, that is, there are blocks in main memory. This helps with efficiency and makes it easier for the processor to access the data it needs.

## Structure of Cache

The cache system is made up of "lines," which are blocks of data that can hold multiple words, typically K words. Each line also has a tag, which helps identify which block of memory the line is holding. Additionally, each line may include control bits that indicate whether the line has been modified since the data was loaded into it. The size of a line is determined by the number of words it can hold excluding the control and tag bits, and it can be as small as 32 bits. Since there are typically fewer lines in the cache than there are blocks in main memory, the tag serves as a way to identify which block of memory is currently stored in a particular line of the cache.

## Contemporary Cache Structure

In a contemporary cache structure, the cache is connected to the processor through data, address, and control lines. The data and address lines also connect to data and address buffers, which in turn connect to the system bus that reaches the main memory. When a cache hit occurs, meaning the requested data is found in the cache, the data and address buffers are disabled, and communication between the processor and cache takes place directly. However, when a cache miss occurs, meaning the requested data is not found in the cache, the data is loaded from the main memory onto the data buffers available on the system bus. The word is then transferred from the system bus to the cache and to the processor for further processing.

# Elements of Cache Design

## Cache Addresses

Cache addresses refer to how the cache is positioned in a system that uses virtual memory. Virtual memory is a facility that allows programs to address memory using logical addresses, regardless of the amount of physical memory available. When virtual addresses are used, the cache can be placed between the processor and the memory management unit (MMU), or between the MMU and main memory. A logical (virtual) cache, which stores data using virtual addresses, can be directly accessed by the processor without needing to go through the MMU, making it faster. On the other hand, a physical cache stores data using physical addresses, which can be slower since it requires additional translation through the MMU before accessing the data. The choice between virtual and physical cache design depends on factors such as performance requirements, system architecture, and implementation considerations.

A hardware memory management unit (MMU) translates each virtual address into a physical address in main memory.

## Cache Size

Cache size refers to the amount of storage capacity in the cache. Ideally, the cache size should be kept as small as possible to minimize the circuit size and access time. Larger caches require more circuitry and may result in slower access times due to increased complexity. Therefore, it's important to carefully consider the appropriate cache size for a system to ensure optimal performance.

(Memory Size) TOTAL NO OF WORDS IN MEMORY: WM = 2^N

(Block Size) no of words in a block: WB = WM / BM

No of blocks in memory: BM = 2^N / WB

No of bits in a word: N = Log2 WM

## Mapping Function

When designing a cache system, a mapping function is required to determine how main memory blocks are assigned to cache lines. This mapping function also helps identify which main memory block is currently stored in a cache line. There are three common techniques for this mapping: direct, associative, and set associative.

1. Direct Mapping: In this technique, each main memory block is mapped to a specific cache line. The mapping is done using a simple formula that assigns each block to a unique line based on its address. This provides a straightforward and efficient mapping but may lead to conflicts when multiple blocks try to occupy the same cache line.

2. Associative Mapping: Associative mapping is a type of cache mapping in which any block of main memory can be stored in any cache line. This means that the cache control logic only needs to store the tag and the block/line offset, and no line number is required. However, this also means that it is not possible to determine which block is loaded in which line.

To implement associative mapping, a six bit comparator is used with each of the cache lines. Data is fed parallel to all the comparators, and all of these are connected to a multi-input OR gate. If the OR gate produces a 1, then it indicates that the desired block is present in one of the cache lines.

3. Set Associative Mapping: In set associative cache memory, the cache is divided into sets of the same size. Each set consists of a certain number of lines. Like direct mapping, any memory block can be assigned to any line but within a specific set. This arrangement makes retrieval easier because we only need to search within a particular set instead of searching the entire cache. Moreover, this approach helps reduce costs because we only need a set of comparators for each block, rather than having a comparator for each line individually. Set associative mapping simplifies retrieval and lowers the cost of cache design.

## Replacing Algorithms

Replacement algorithms come into play when the cache is full and a new block needs to be brought in. In such cases, an existing block must be replaced. There are different replacement algorithms to decide which block should be replaced.

1. LRU (least recently used) replacement algorithm replaces the block that has not been used in the longest time. This is implemented with the help of a USE bit, which is set to 1 when the block is used and reset to 0 when the block is not used. The block with the USE bit set to 0 for the longest time is the least recently used block and is replaced.
2. FIFO (First In First Out): The block that has been in the cache for the longest time, meaning the one that arrived first, is replaced. FIFO is easily implemented as a round-robin technique, where blocks are replaced in the order they were added to the cache.
3. LFU (least frequently used) replacement algorithm replaces the block that has been used the fewest times. This is implemented by associating a counter with each block in the cache. The counter is incremented each time the block is used and decremented each time the block is not used. The block with the lowest counter value is the least frequently used block and is replaced.

The choice of replacement algorithm depends on the specific application.

## Write Policy

In cache memory, the write policy determines what happens when a block needs to be replaced. There are two common techniques:

1. Write Through: When a block is replaced, if no changes have been made to the block, it is simply overridden with the new block. However, if even a single change has been made to the block, it must also be updated in the main memory. This ensures that both the cache and main memory are always in sync. The main disadvantage of this approach is that it increases memory traffic since every write operation involves updating both the cache and main memory.

2. Write Back: With write back, updates are only made in the cache when changes occur. When a block is updated, a "dirty bit" associated with the cache line is set to 1, indicating that the block in the cache is different from the corresponding block in the main memory. When a block needs to be replaced, the dirty bit is checked. If it is 1, meaning changes were made, the block is written back to the main memory. If the dirty bit is 0, indicating no changes, the block is not written back. This approach reduces memory traffic compared to write through.

These write policies determine how write operations are handled in cache memory, considering the need for updating main memory and managing data consistency.