UNIT-4

Q-1)

a) Name the different modes of data transfer?

Three possible Data transfer modes are:
1. Programmed I/O
2. Interrupt-initiated I/O
3. Direct memory access (DMA)

b) Define Priority Interrupt.

A priority interrupt is a computer architecture which decides the priority at which various devices, which generates the interrupt signal at the same time, will be serviced by the CPU.

c) What are Auxiliary and Cache memory?

- An **Auxiliary memory** is known as the lowest-cost, highest-capacity and slowest-access storage in a computer system. It is where programs and data are kept for long-term storage or when not in immediate use. The most common examples of auxiliary memories are magnetic tapes and magnetic disks.
- **Cache memory** is a special type of high-speed memory located close to the CPU in a computer. It stores frequently used data and instructions, So that the CPU can access them quickly, improving the overall speed and efficiency of the computer. It is a faster and smaller segment of memory whose access time is as close as registers.

d) Write 2 operational differences between RAM and ROM.

- RAM is high-speed, volatile memory used to store and process temporary data. ROM, on the other hand, is non-volatile memory used to store lasting data like firmware.
- RAM is more flexible, but it costs more, while ROM is more stable and secure, but it doesn't have as much freedom.

e) Write one operational difference between the main memory and the auxiliary memory.

- Main Memory is a computer's short-term storage, whereas Auxiliary Memory is the long-term storage.
- Main Memory is faster but more expensive and has less capacity compared to Auxiliary Memory.

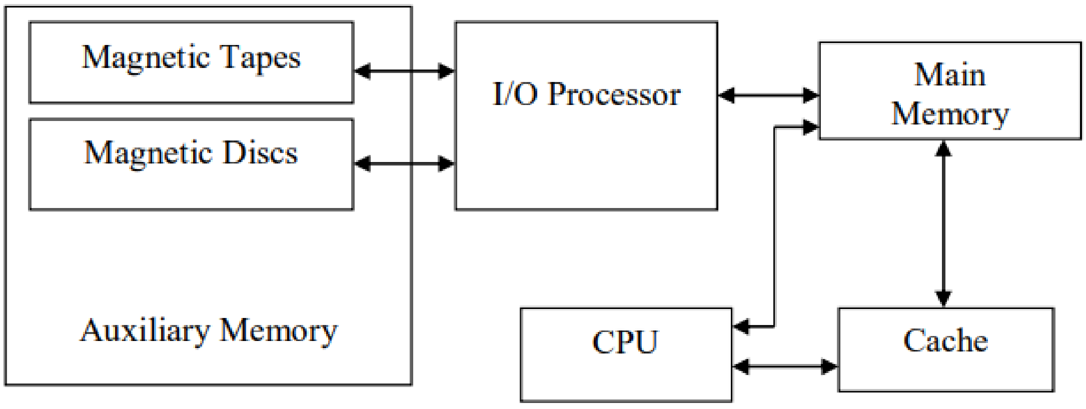
Q-2)

a) Give the memory hierarchy in a computer system.

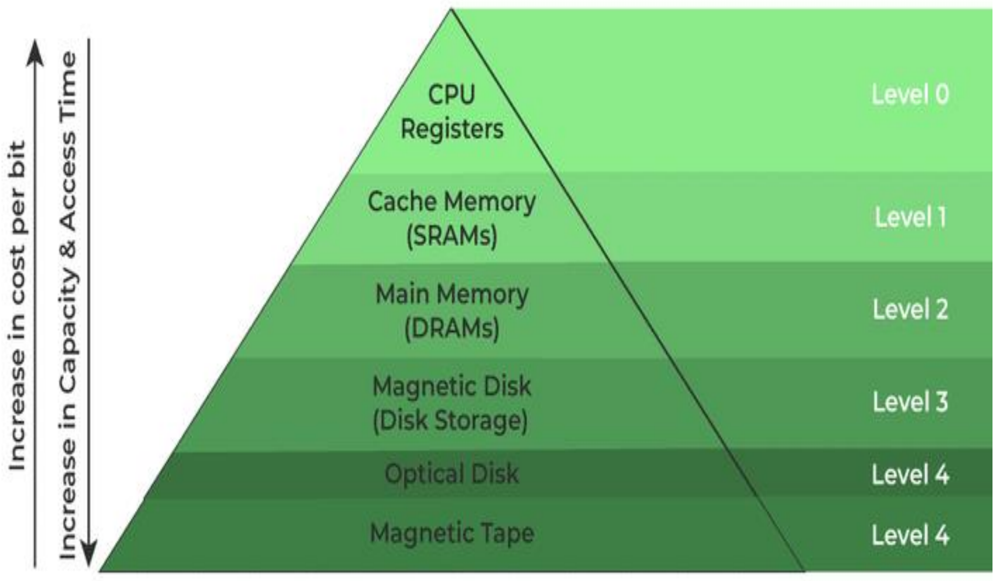
The memory unit is an essential component in any digital computer since it is needed for storing programs and data. A very small computer with a limited application may be able to fulfil its intended task without the need of additional storage capacity. The memory unit that communicates directly with the CPU is called the main memory. Devices that provide backup storage are called auxiliary memory. The most common auxiliary memory devices used in computer systems are magnetic disks and tapes. They are used for storing system programs, large data files, and other backup information. Only programs and data currently needed by the processor reside in main memory. All other information is stored in auxiliary memory and transferred to main memory when needed.

In the Computer System, Memory Hierarchy is an enhancement to organize the memory such that it can minimize the access time. The memory hierarchy includes different components, generally split into one of the two types:

- **Internal Memory (Primary Memory)**. Registers, main memory, and cache. Internal memory is directly accessible by the processor.
- **External Memory (Secondary Memory)**. Secondary storage (HDDs, SSDs) and tertiary storage (magnetic disk, magnetic tape, and optical disk. This is peripheral storage accessible by the processor via an I/O Module.



Memory hierarchy in computer system



b) What is the difference between isolated I/O and memorymapped I/O? What are the advantages and disadvantages of each?

Differences between memory mapped I/O and isolated I/O –

Isolated I/O	**Memory Mapped I/O**
Memory and I/O have separate address space	Both have same address space
All address can be used by the memory	Due to addition of I/O addressable memory become less for memory
Separate instruction control read and write operation in I/O and Memory	Same instructions can control both I/O and Memory
In this I/O address are called ports.	Normal memory address are for both
More efficient due to separate buses	Lesser efficient
Larger in size due to more buses	Smaller in size
It is complex due to separate logic is used to control both.	Simpler logic is used as I/O is also treated as memory only.

Advantages of Memory-Mapped I/O:

Faster I/O Operations: Memory-mapped I/O allows the CPU to access I/O devices at the same speed as it accesses memory. This means that I/O operations can be performed much faster compared to isolated I/O.
Simplified Programming: Memory-mapped I/O simplifies programming as the same instructions can be used to access memory and I/O devices. This means that software developers do not have to use specialized I/O instructions, which can reduce programming complexity.
Efficient Use of Memory Space: Memory-mapped I/O is more memory-efficient as I/O devices share the same address space as the memory. This means that the same memory address space can be used to access both memory and I/O devices.

Disadvantages of Memory-Mapped I/O:

Limited I/O Address Space: Memory-mapped I/O limits the I/O address space as I/O devices share the same address space as the memory. This means that there may not be enough address space available to address all I/O devices.
Slower Response Time: If an I/O device is slow to respond, it can delay the CPU's access to memory. This can lead to slower overall system performance.

Advantages of Isolated I/O:

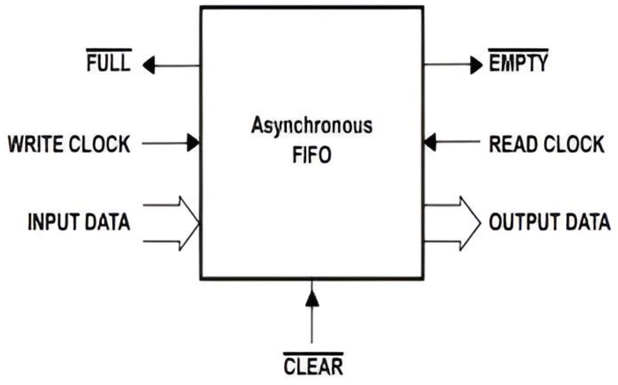
Large I/O Address Space: Isolated I/O allows for a larger I/O address space than memory-mapped I/O as I/O devices have separate address spaces.
Greater Flexibility: Isolated I/O provides greater flexibility as I/O devices can be added or removed from the system without affecting the memory address space.
Improved Reliability: Isolated I/O provides better reliability as I/O devices do not share the same address space as the memory. This means that if an I/O device fails, it does not affect the memory or other I/O devices.

Disadvantages of Isolated I/O:

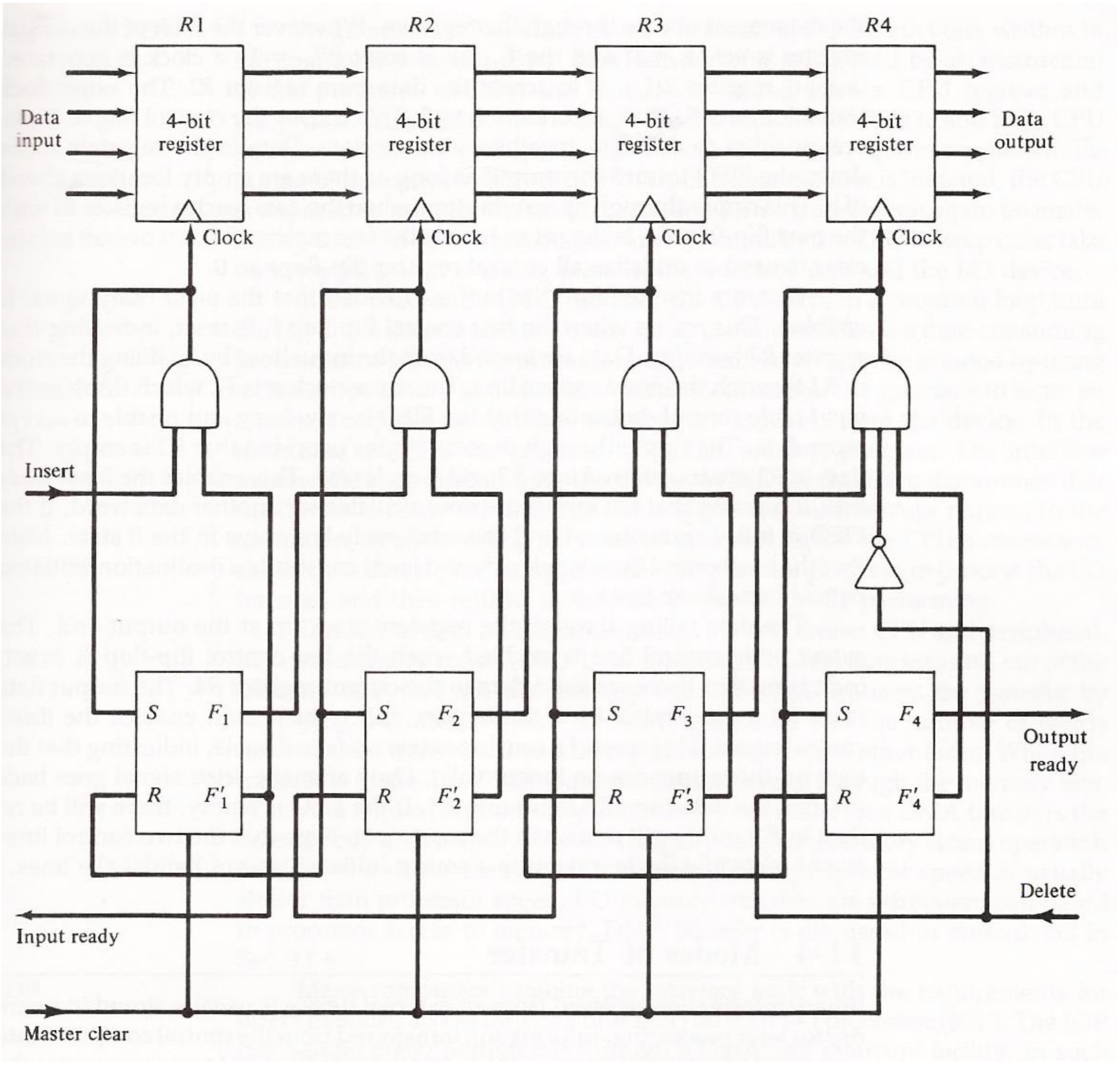
Slower I/O Operations: Isolated I/O can result in slower I/O operations than memory-mapped I/O as it requires specialized I/O instructions.
More Complex Programming: Isolated I/O requires specialized I/O instructions, which can lead to more complex programming.

c) Write about first-in and first-out buffers in asynchronous data transfer.

A FIFO (First-In, First-Out) is a type of data buffer or queue where the first piece of data entered is the first one to be removed. This order-preserving mechanism is essential in various digital systems, where data must be processed or transmitted in the exact sequence it was received. FIFOs are fundamental components in digital systems, offering a simple yet powerful mechanism for managing data flow across various applications.
An Asynchronous FIFO (First-In, First-Out) is a type of data buffer or queue used in digital systems where different clock signals control the writing and reading of data. This flexibility allows the FIFO to operate efficiently across different clock domains, making it ideal for applications where data transfer must occur between components that do not share the same clock frequency.

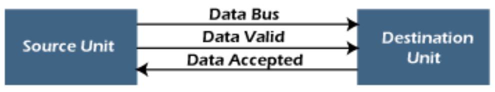


The block diagram for a 4 Register FIFO is shown next.

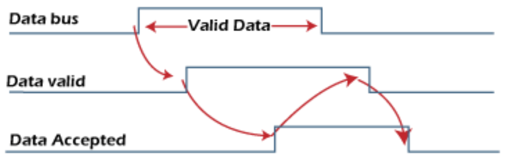


d) Define Source-initiated handshaking.

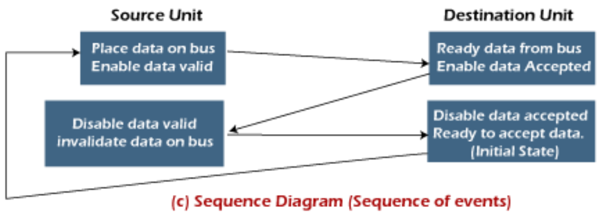
Source-initiated handshaking: In the source Initiated the handshaking process after sending the valid data, the receiver sends the acknowledgment that the data has been received. Hence, the signal 'DATA VALID' is sent by the sender before sending the data, and the signal 'DATA ACCEPTED' is sent by the receiver after getting the data.



(a) Block Diagram



(b) Timing Diagram



(c) Sequence Diagram (Sequence of events)

The timing diagram shows the timing relationship of the exchange of signals between the two units. The source initiates a transfer by placing data on the bus and enabling its data valid signal. The destination unit then activates the data accepted signal after it accepts the data from the bus.

Q-3)

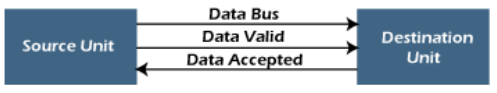
a) What is meant by handshaking? Explain with a neat diagram.

Handshaking is an I/O control approach to synchronize I/O devices with the microprocessor. As several I/O devices accept or release data at a much lower cost than the microprocessor, this technique is used to control the microprocessor to operate with an I/O device at the I/O devices data transfer rate.

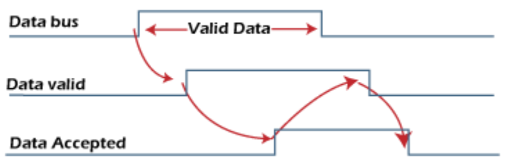
The strobe method has the disadvantage that the source unit that initiates the transfer has no way of knowing whether the destination has received the data that was placed in the bus. Similarly, a destination unit that initiates the transfer has no way of knowing whether the source unit has placed data on the bus. This problem is solved by the handshaking method. The handshaking method introduces a second control signal line that replays the unit that initiates the transfer.

In this method, one control line is in the same direction as the data flow in the bus from the source to the destination. The source unit uses it to inform the destination unit whether there are valid data in the bus. The other control line is in the other direction from the destination to the source. This is because the destination unit uses it to inform the source whether it can accept data. And in it also, the sequence of control depends on the unit that initiates the transfer. So it means the sequence of control depends on whether the transfer is initiated by source and destination.

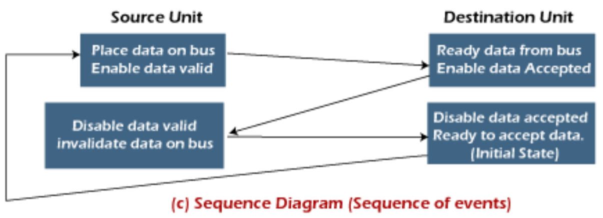
- **Source initiated handshaking:** In the source Initiated the handshaking process after sending the valid data, the receiver sends the acknowledgment that the data has been received. Hence, the signals 'DATA VALID' is sent by the sender before sending the data and the signal 'DATA ACCEPTED' is sent by the receiver after getting the data.



(a) Block Diagram



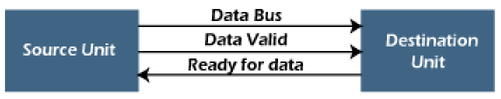
(b) Timing Diagram



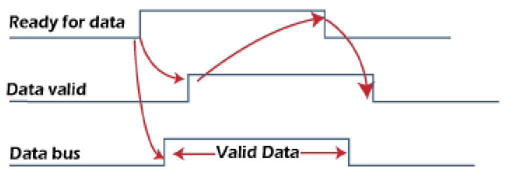
(c) Sequence Diagram (Sequence of events)

The timing diagram shows the timing relationship of the exchange of signals between the two units. The source initiates a transfer by placing data on the bus and enabling its data valid signal. The destination unit then activates the data accepted signal after it accepts the data from the bus.

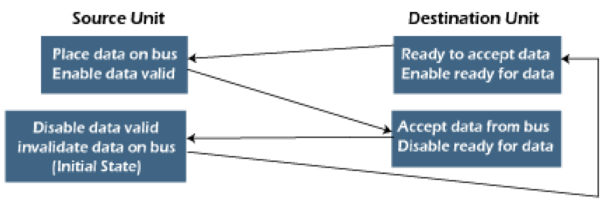
- **Destination initiated handshaking:** In this process at first destination unit generate "**ready for data**" signal after that "**data valid**", generated by the source unit, Hence, the signals "**data valid**" is sent by the sender after sending the data into data bus.



(a) Block Diagram



(b) Timing Diagram



(c) Sequence Diagram (Sequence of events)

b) Explain Auxiliary Memory.

An Auxiliary memory is known as the lowest-cost, highest-capacity and slowest-access storage in a computer system. It is a non-volatile memory and used to store a large amount of data or information. The data or information stored in secondary memory is permanent, and it is slower than primary memory. A CPU cannot access secondary memory directly. The data/information from the auxiliary memory is first transferred to the main memory, and then the CPU can access it.

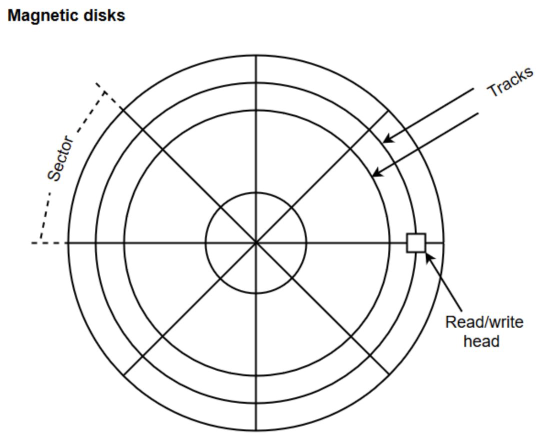
Characteristics of Auxiliary Memory

- It is a slow memory but reusable.
- It is a reliable and non-volatile memory.
- It is cheaper than primary memory.
- The storage capacity of secondary memory is large.
- A computer system can run without secondary memory.
- In secondary memory, data is stored permanently even when the power is off.

Types of Secondary Memory

I. **Magnetic Disks:** A magnetic disk is a type of memory constructed using a circular plate of metal or plastic coated with magnetized materials. Usually, both sides of the disks are used to carry out read/write operations. However, several disks may be stacked on one spindle with read/write head available on each surface.

The following image shows the structural representation for a magnetic disk.



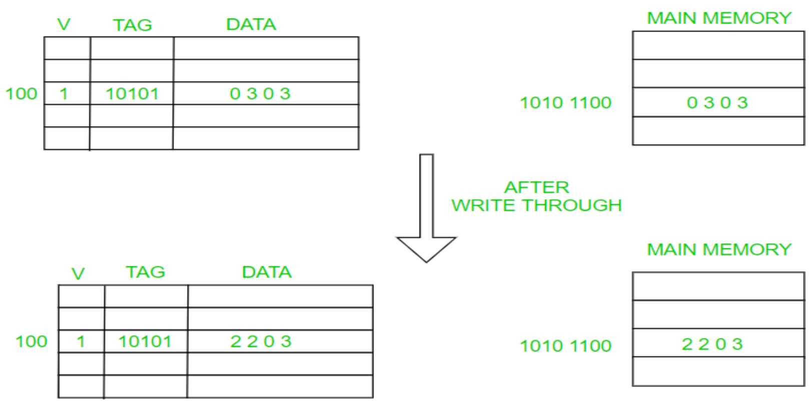
1. The memory bits are stored in the magnetized surface in spots along the concentric circles called tracks.
2. The concentric circles (tracks) are commonly divided into sections called sectors.

a. **Magnetic Tapes:** Magnetic tape is a long, narrow strip of plastic film with a thin, magnetic coating on it that is used for magnetic recording. Bits are recorded on tape as magnetic patches called RECORDS that run along many tracks. Typically, 7 or 9 bits are recorded concurrently. Each track has one read/write head, which allows data to be recorded and read as a sequence of characters. It can be stopped, started moving forward or backward, or rewound.

c) How do you write into a cache?

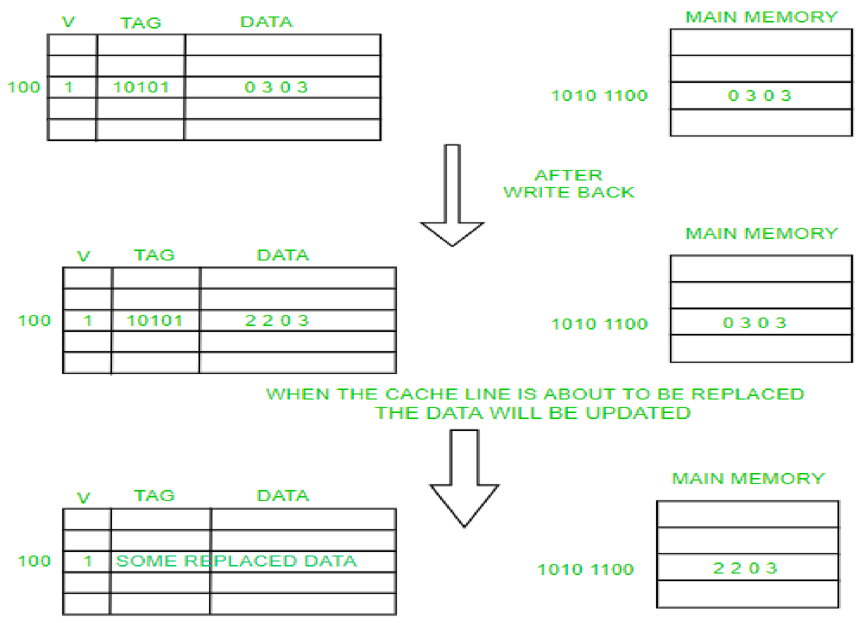
Cache is a technique of storing a copy of data temporarily in rapidly accessible storage memory. Cache stores most recently used words in small memory to increase the speed at which data is accessed.
Whenever a Processor wants to write a word, it checks that the address is present in the cache or not. If the address is present in the cache, then this two approach is used to write the data into cache, i.e., **Write Through** and **Write Back**.

Write Through:



In write-through, data is **simultaneously updated to cache and memory**. This process is simpler and more reliable. This is used when there are no frequent writes to the cache (The number of write operations is less). It helps in data recovery (In case of a power outage or system failure).

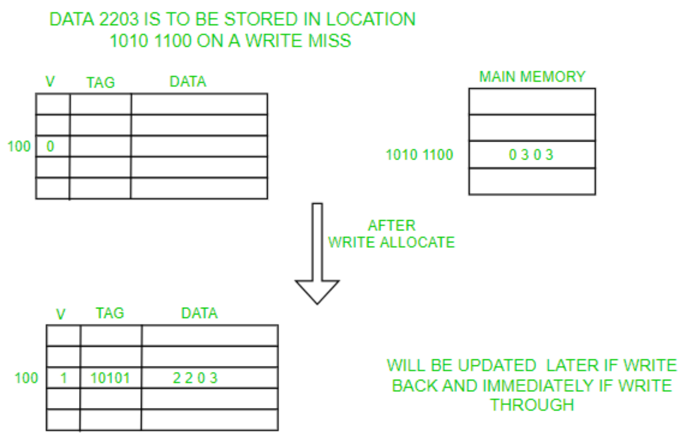
Write Back:



In write-back method only the cache location is updated during a write operation. The location is then marked by a flag so that later when the words are removed from the cache it is copied into main memory. The reason for the write-back method is that during the time a word resides in the cache, it may be updated several times; however, as long as the word remains in the cache, it does not matter whether the copy in main memory is out of date, since requests from the word are filled from the cache. It is only when the word is displaced from the cache that an accurate copy need be rewritten into main memory.

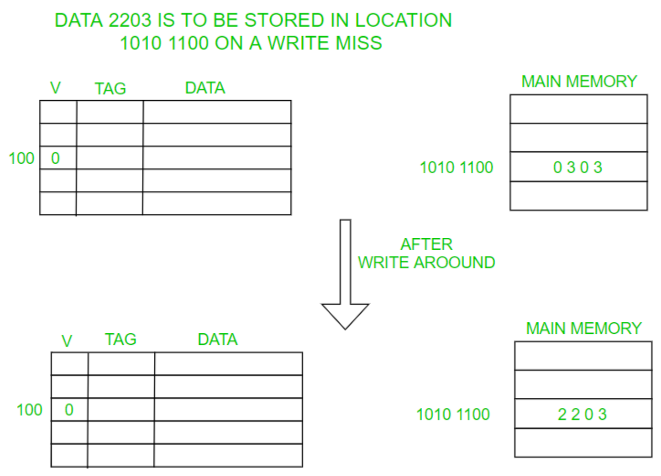
If write occurs to a location that is not present in the Cache(Write Miss), we use two options, **Write Allocation** and **Write Around**.

Write Allocation:


DATA 2203 IS TO BE STORED IN LOCATION
1010 1100 ON A WRITE MISS
V
TAG
DATA
100
0
MAIN MEMORY
1010 1100
0 3 0 3
AFTER
WRITE ALLOCATE
V
TAG
DATA
100
1
10101
2 2 0 3
WILL BE UPDATED LATER IF WRITE
BACK AND IMMEDIATELY IF WRITE
THROUGH

In Write Allocation data is loaded from the memory into cache and then updated. Write allocation works with both Write back and Write through. But it is generally used with Write Back because it is unnecessary to bring data from the memory to cache and then updating the data in both cache and main memory. Thus Write Through is often used with No write Allocate.

Write Around:



Here data is Directly written/updated to the main memory without disturbing the cache. It is better to use this when the data is not immediately used again.

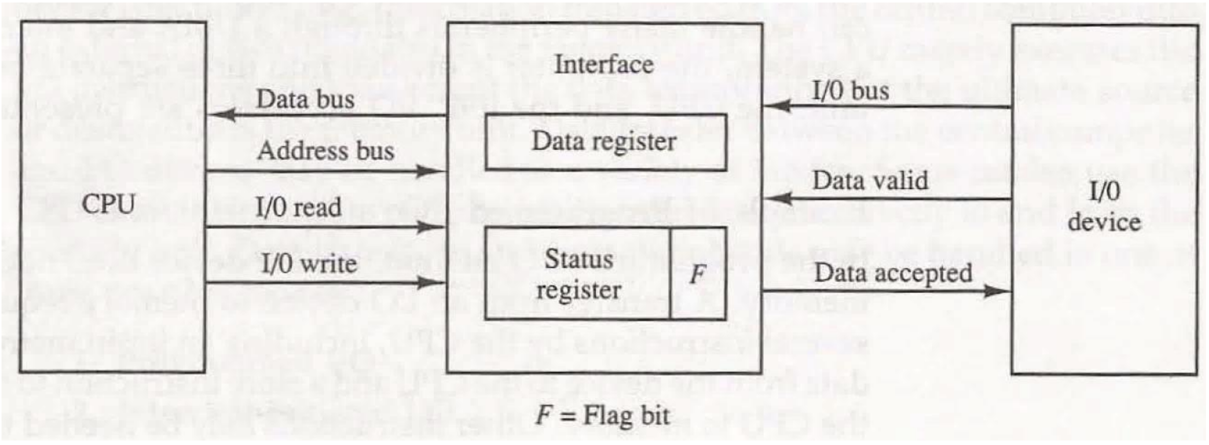
d) What is programmed I/O? Discuss the data transfer from an I/O device through an interface into the CPU.

Programmed I/O operations are the result of I/O instructions written in the computer program. Each data item transfer is initiated by an instruction in the program. Usually, the transfer is to and from a CPU register and peripheral. Other instructions are needed to transfer the data to and from CPU and memory. Transferring data under program control requires constant monitoring of the peripheral by the CPU. Once a data transfer is initiated, the CPU is required to monitor the interface to see when a transfer can again be made. It is up to the programmed instructions executed in the CPU to keep close tabs on everything that is taking place in the interface unit and the I/O device.

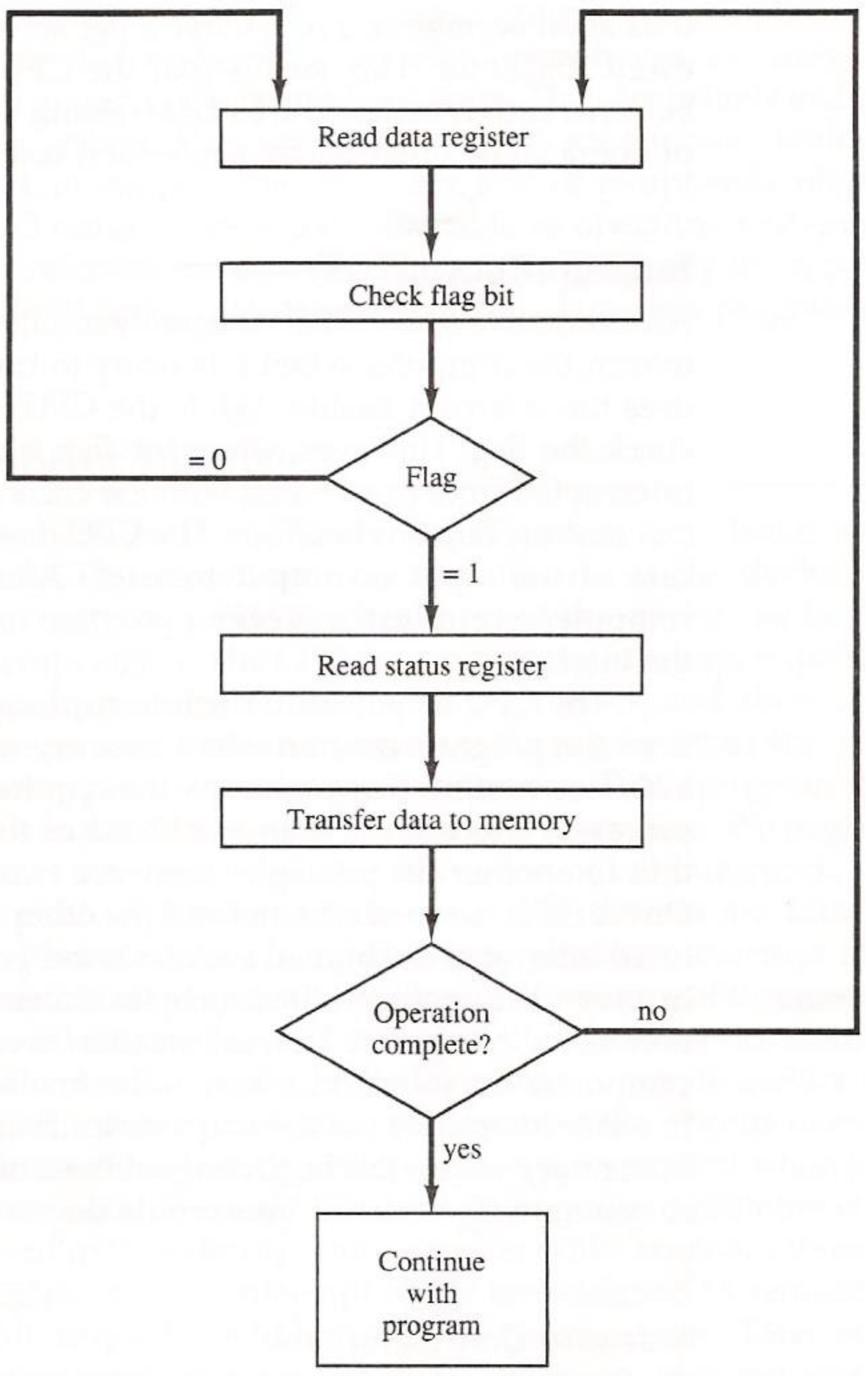
In the programmed I/O method, the CPU stays in a program loop until the I/O unit indicates that it is ready for data transfer. This is a time-consuming process since it keeps the processor busy needlessly. It can be avoided by using an interrupt facility and special commands to inform the interface to issue an interrupt request signal when the data are available from the device. In the meantime, the CU can proceed to execute another program. The interface meanwhile keeps monitoring the device. When the interface determines that the device is ready for data transfer, it generates an interrupt request to the computer. Upon detecting the external interrupt signal, the CPU momentarily stops the task it is processing, branches to a service program to process the I/O transfer, and then returns to the task it was originally performing Transfer of data under programmed I/O is between CPU and peripheral.

In the programmed I/O method, the I/O device does not have direct access to memory. A transfer from an I/O device to memory requires the execution of several instructions by the CPU, including an input instruction to transfer the data from the device to the CPU, and a store instruction to transfer the data from the CPU to memory. Other instructions may be needed to verify that the data are available from the device and to count the numbers of words transferred.

An example of data transfer from an I/O device through an interface into the CPU is shown in Fig.



A flowchart of the program that must be written for the CPU is shown in Fig.


Read data register
Check flag bit
Flag
= 0
= 1
Read status register
Transfer data to memory
Operation complete?
no
yes
Continue with program

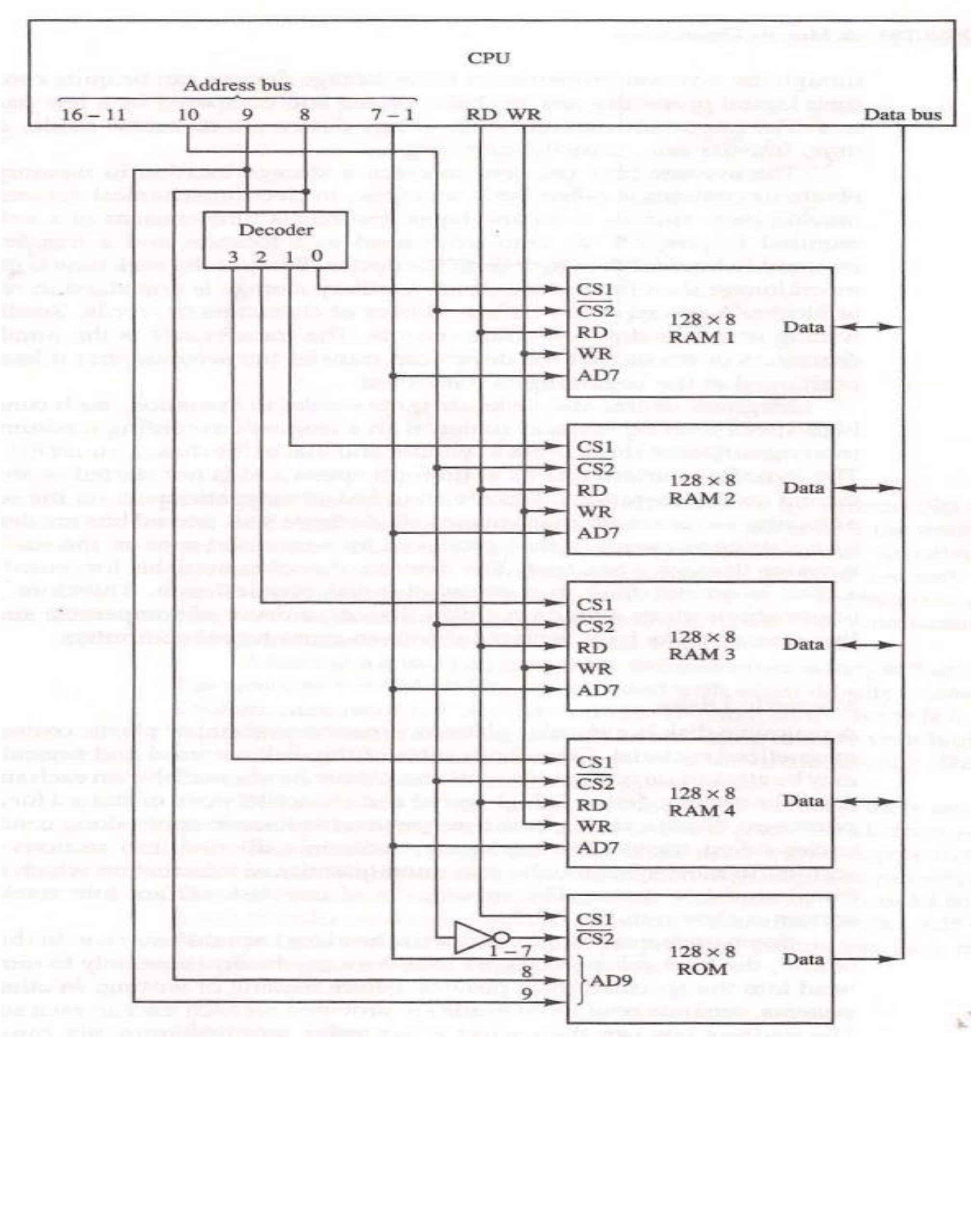
e) Calculate the number of 128x8 RAM chips and 512x8 ROM chips required to design a computer system that needs 512 bytes of RAM and 512 bytes of ROM. Also, give the memory address map.

Basic RAM size = 128 x 8
Required RAM size = 512 x 8
Basic ROM size = 512 x 8
Required ROM size is also 512 x 8

To design a computer system of 512 bytes ROM, Number of chips required is 1.

To design a RAM size of 512×8 from 128×8, Number of chips required = Desired RAM Size/ Basic RAM Size

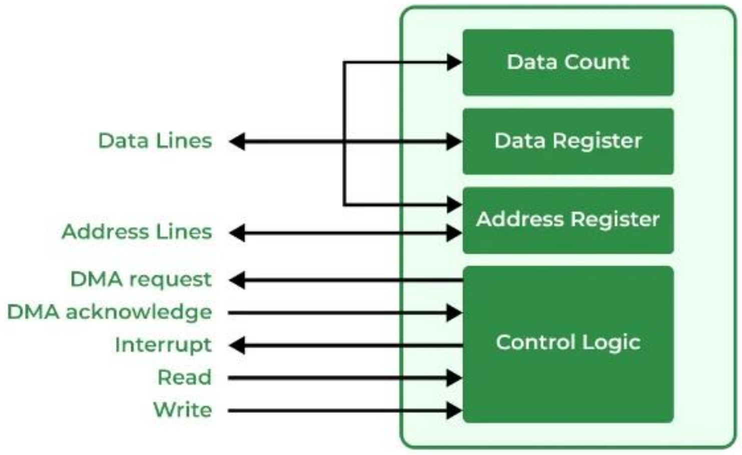
=512x8/128x8
=4 chips



Q-4)

a) Explain the internal configuration of a DMA Controller diagrammatically and it works.

In modern computer systems, transferring data between input/output devices and memory can be a slow process if the CPU is required to manage every step. To address this, a Direct Memory Access (DMA) Controller is utilized. A Direct Memory Access (DMA) Controller solves this by allowing I/O devices to transfer data directly to memory, reducing CPU involvement. This increases system efficiency and speeds up data transfers, freeing the CPU to focus on other tasks. DMA controller needs the same old circuits of an interface to communicate with the CPU and Input/Output devices.
Direct Memory Access (**DMA)** uses hardware for accessing the memory, that hardware is called a DMA Controller. It has the work of transferring the data between Input Output devices and main memory with very less interaction with the processor. The direct Memory Access Controller is a control unit, which has the work of transferring data.
DMA Controller is a type of control unit that works as an interface for the data bus and the I/O Devices. As mentioned, DMA Controller has the work of transferring the data without the intervention of the processors, processors can control the data transfer. DMA Controller also contains an address unit, which generates the address and selects an I/O device for the transfer of data. Here we are showing the block diagram of the DMA Controller.



Block diagram of DMA Controller

Types of Direct Memory Access (DMA)
There are four popular types of DMA.

- Single-Ended DMA
- Dual-Ended DMA
- Arbitrated-Ended DMA
- Interleaved DMA

Single-Ended DMA: Single-Ended DMA Controllers operate by reading and writing from a single memory address. They are the simplest DMA.
Dual-Ended DMA: Dual-Ended DMA controllers can read and write from two memory addresses. Dual-ended DMA is more advanced than single-ended DMA.
Arbitrated-Ended DMA: Arbitrated-Ended DMA works by reading and writing to several memory addresses. It is more advanced than Dual-Ended DMA.

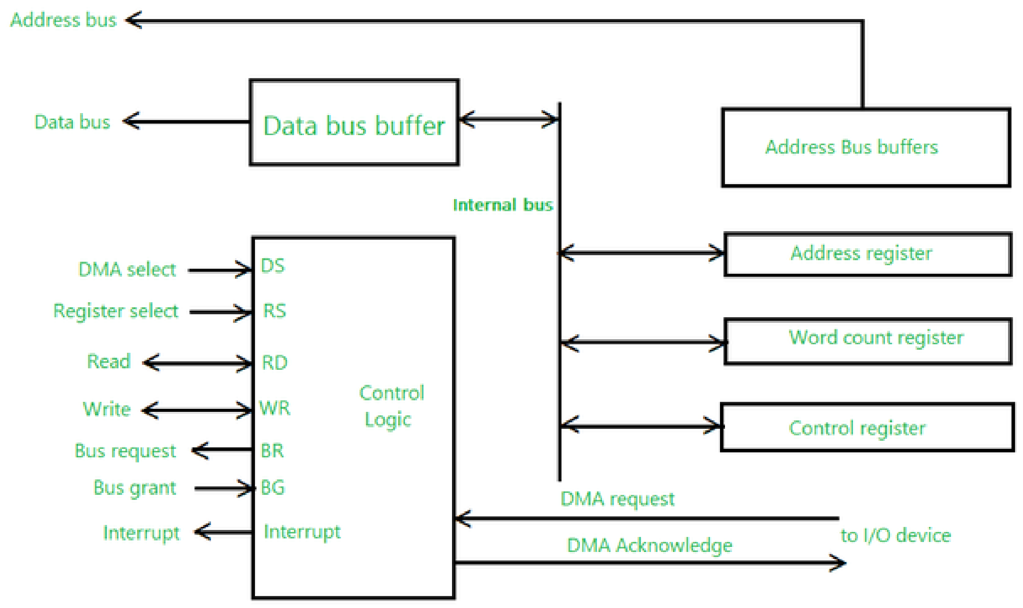
Interleaved DMA: Interleaved DMA are those DMA that read from one memory address and write from another memory address.

Working of DMA Controller

The DMA controller registers have three registers as follows.

- **Address register –** It contains the address to specify the desired location in memory.
- **Word count register –** It contains the number of words to be transferred.
- **Control register –** It specifies the transfer mode.

The figure below shows the block diagram of the DMA controller. The unit communicates with the CPU through the data bus and control lines. Through the use of the address bus and allowing the DMA and RS register to select inputs, the register within the DMA is chosen by the CPU. RD and WR are two-way inputs. When BG (bus grant) input is 0, the CPU can communicate with DMA registers. When BG (bus grant) input is 1, the CPU has relinquished the buses and DMA can communicate directly with the memory.



Working Diagram of DMA Controller

b) Explain the hardware organization of the associative memory in detail.

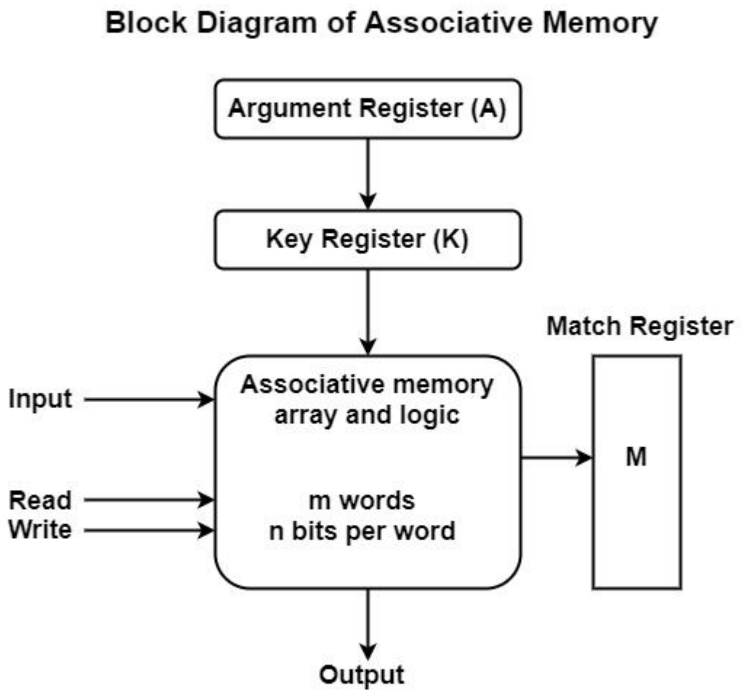
An associative memory can be treated as a memory unit whose saved information can be recognized for approach by the content of the information itself instead of by an address or memory location. Associative memory is also known as **Content Addressable Memory (CAM).**

The block diagram of associative memory is shown in the figure. It includes a memory array and logic for m words with n bits per word. The argument register A and key register K each have n bits, one for each bit of a word.

The match register M has m bits, one for each memory word. Each word in memory is related in parallel with the content of the argument register.

The words that connect the bits of the argument register set an equivalent bit in the match register. After the matching process, those bits in the match register that have been set denote the fact that their equivalent words have been connected.

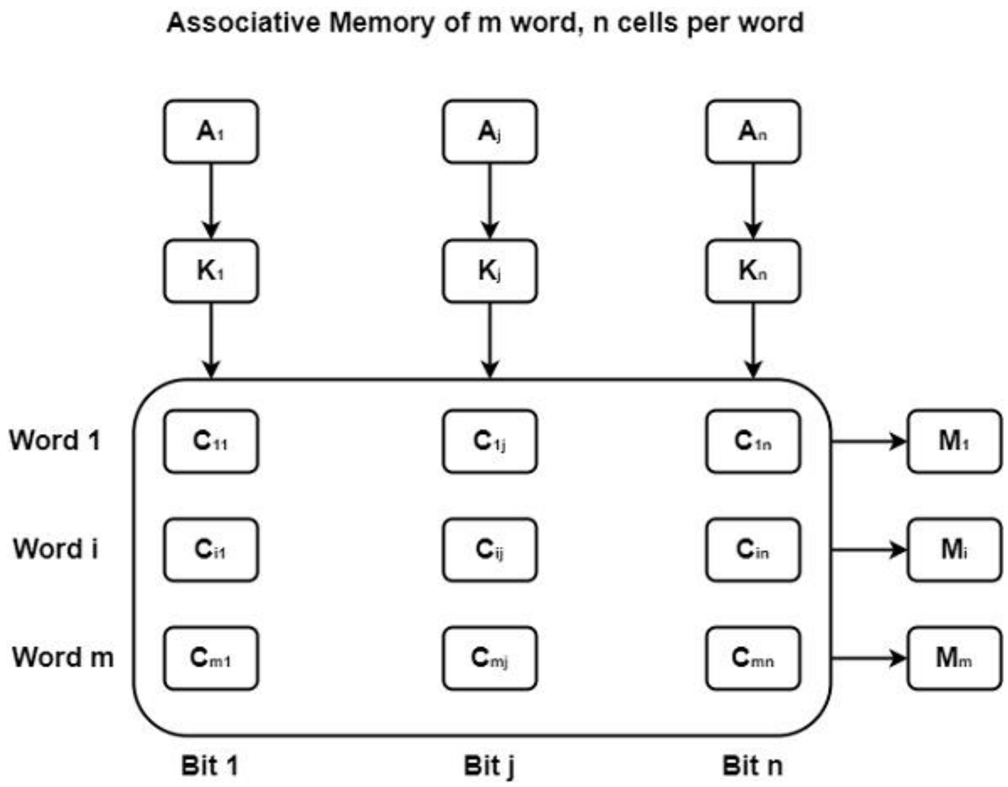
Reading is proficient through sequential access to memory for those words whose equivalent bits in the match register have been set.



The key register supports a mask for selecting a specific field or key in the argument word. The whole argument is distinguished with each memory word if the key register includes all 1's.

Hence, there are only those bits in the argument that have 1's in their equivalent position of the key register are compared. Therefore, the key gives a mask or recognizing a piece of data that determines how the reference to memory is created.

The following figure can define the relation between the memory array and the external registers in associative memory.



The cells in the array are considered by the letter C with two subscripts. The first subscript provides the word number and the second determines the bit position in the word. Therefore cell C_{ij} is the cell for bit j in word i.

A bit in the argument register is compared with all the bits in column j of the array supported that $K_j = 1$. This is completed for all columns j = 1, 2 . . . , n.

If a match appears between all the unmasked bits of the argument and the bits in word i, the equivalent bit M_i in the match register is set to 1. If one or more unmasked bits of the argument and the word do not match, M_i is cleared to 0.

c) Explain the cache memory mapping techniques.

Cache mapping is the procedure in to decide in which cache line the main memory block will be mapped. In other words, the pattern used to copy the required main memory content to the specific location of cache memory is called cache mapping. The process of extracting the cache memory location and other related information in which the required content is present from the main memory address is called as cache mapping. The cache mapping is done on the collection of bytes called blocks. In the mapping, the block of main memory is moved to the line of the cache memory. Cache mapping is needed to identify where the cache memory is present in cache memory. Mapping provides the cache line number where the content is present in the case of cache hit or where to bring the content from the main memory in the case of cache miss.

There are three types of cache mappings namely:

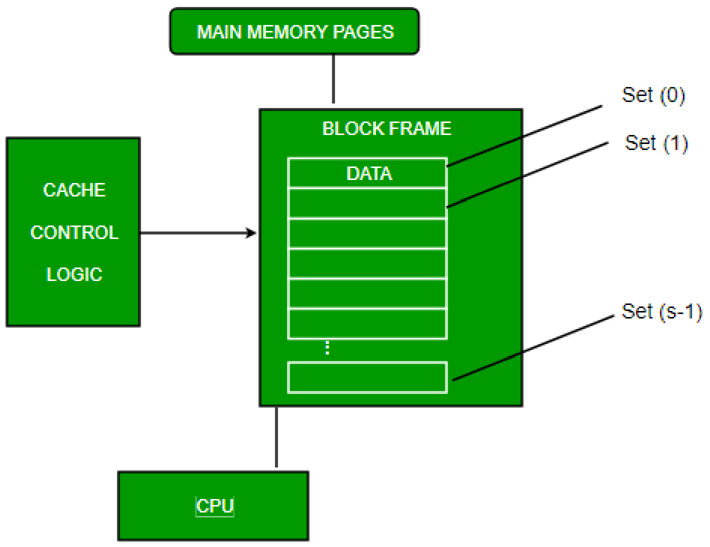
- Direct Mapping
- Fully Associative Mapping
- Set Associative Mapping

Direct Mapping

In direct mapping physical address is divided into three parts i.e., Tag bits, Cache Line Number and Byte offset. The bits in the cache line number represents the cache line in which the content is present whereas the bits in tag are the identification bits that represents which block of main memory is present in cache. The bits in the byte offset decides in which byte of the identified block the required content is present.

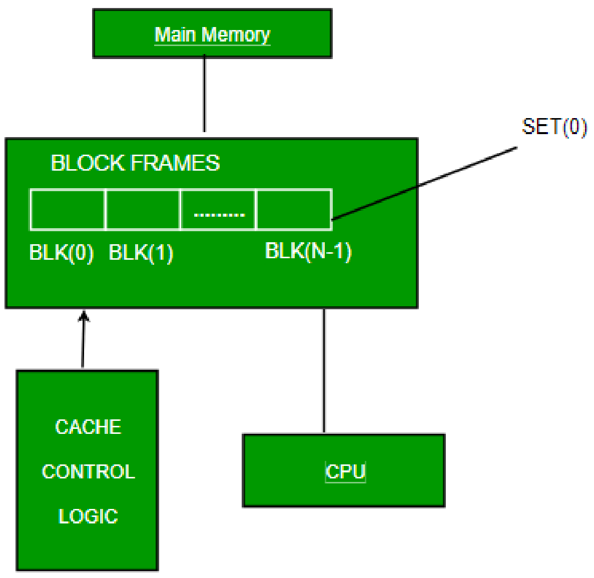
Tag	Number of Cache Lines	Byte Offset

Cache Line Number = Main Memory block Number % Number of Blocks in Cache



Fully Associative Mapping: In fully associative mapping address is divided into two parts i.e., Tag bits and Byte offset. The tag bits identify that which memory block is present and bits in the byte offset field decides in which byte of the block the required content is present.

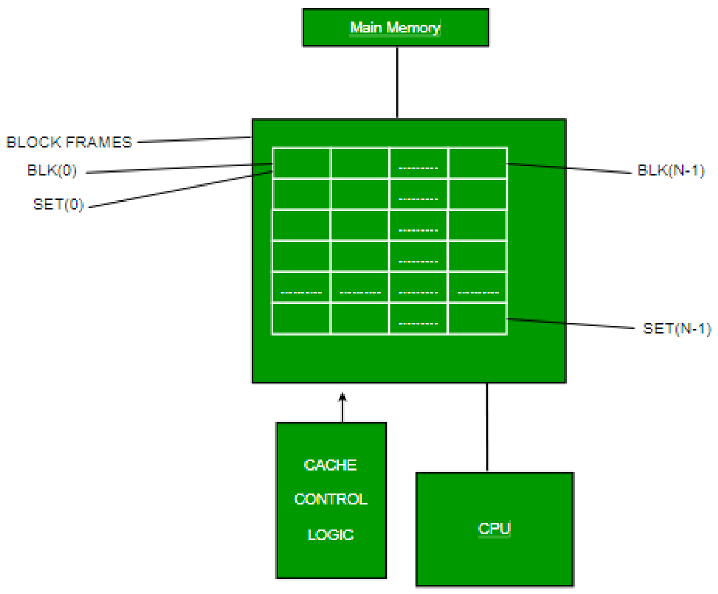
Tag	Byte Offset



Set Associative Mapping
In set associative mapping the cache blocks are divided in sets. It divides address into three parts i.e., Tag bits, set number and byte offset. The bits in set number decides that in which set of the cache the required block is present and tag bits identify which block of the main memory is present. The bits in the byte offset field gives us the byte of the block in which the content is present.

Tag	Set Number	Byte Offset

Cache Set Number = Main Memory block number % Number of sets in cache


Main Memory
BLOCK FRAMES
BLK(0)
SET(0)
BLK(N-1)
SET(N-1)
CACHE CONTROL LOGIC
CPU