Unit 3 Memory & I/O

Characteristics of Computer Memory Systems:-

Location	**Performance**
Internal (e.g., processor registers, cache, main memory)	Access time
External (e.g., optical disks, magnetic disks, tapes)	Cycle time
Capacity	Transfer rate
Number of words	**Physical Type**
Number of bytes	Semiconductor
Unit of Transfer	Magnetic
Word	Optical
Block	Magneto-optical
Access Method	**Physical Characteristics**
Sequential	Volatile/nonvolatile
Direct	Erasable/nonerasable
Random	**Organization**
Associative	Memory modules

Types is the method of accessing units of data. These include the following:

■ **Sequential access:** Memory is organized into units of data, called records. Access must be made in a specific linear sequence. Stored addressing information is used to separate records and assist in the retrieval process. A shared read–write mechanism is used, and this must be moved from its current location to the desired location, passing and rejecting each intermediate record. Thus, the time to access an arbitrary record is highly variable.

■ **Direct access:** it involves accessing data using a shared read-write mechanism where each block or record has a unique physical address. This method allows data to be located by directly reaching a general area, followed by sequential searching to reach the exact location. Access time varies based on the data's position, and examples include disk drives.

■ **Random access:** Each addressable location in memory has a unique, physically wired- in addressing mechanism. The time to access a given location is independent of the sequence of prior accesses and is constant. Thus, any location can be selected at random and directly addressed and accessed. Main memory and some cache systems are random access.

■ **Associative**: This is a random access type of memory that enables one to make a comparison of desired bit locations within a word for a specified match, and to do this for all words simultaneously.

User's point of view, the two most important characteristics of memory are **capacity** and **performance**

Three performance parameters are used:

■ **Access time (latency):** For random-access memory, this is the time it takes to perform a read or write operation, that is, the time from the instant that an address is presented to the memory to the instant that data have been stored or made available for use. For non- random- access memory, access time is the time it takes to position the read–write mechanism at the desired location.

■ **Memory cycle time**: This concept is primarily applied to random-access memory and consists of the access time plus any additional time required before a second access can commence. This additional time may be required for transients to die out on signal lines or to regenerate data if they are read destructively. Note that memory cycle time is concerned with the system bus, not the processor.

■ **Transfer rate**: This is the rate at which data can be transferred into or out of a memory unit. For random-access memory, it is equal to 1/(cycle time). For non-random-access memory, the following relationship holds:

$Tn = TA + n/ R$

where

Tn = Average time to read or write n bits

TA = Average access time

n = Number of bits

R = Transfer rate, in bits per second (bps)

■ **Physical types**

1) semiconductor memory,

2) magnetic surface memory

3) used for disk and tape

4) optical

5) magneto-optical

■ physical characteristics

1) **volatile/nonvolatile**:- In a volatile memory, information decays naturally or is lost when electrical power is switched off. In a nonvolatile memory, information once recorded remains without deterioration until deliberately changed; no electrical power is needed to retain information. Magnetic-surface memories are nonvolatile. Semiconductor memory (memory on integrated circuits) may be either volatile or nonvolatile.

2) **Nonerasable/erasable**:- . Nonerasable memory cannot be altered, except by destroying the storage unit. Semiconductor memory of this type is known as read-only memory (ROM). Of necessity, a practical nonerasable memory must also be nonvolatile.

The Memory Hierarchy :

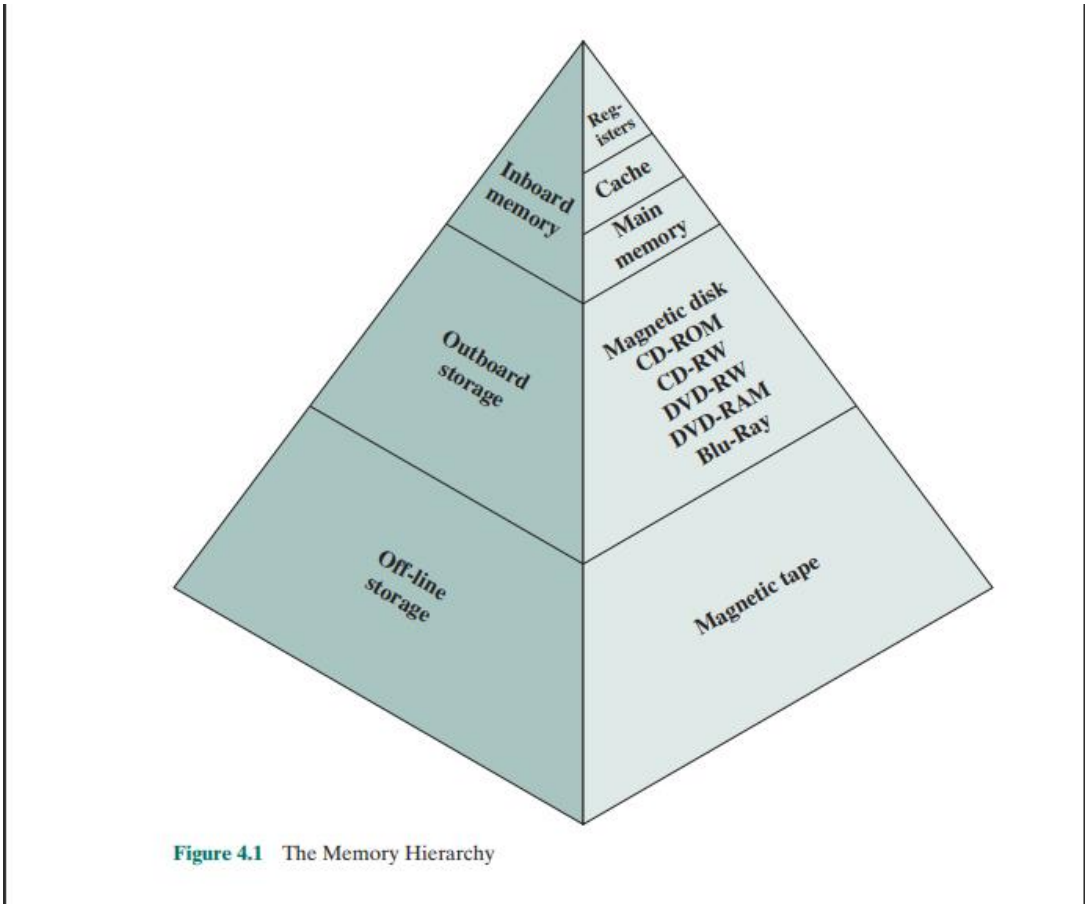


Figure 4.1 The Memory Hierarchy

The way out of this dilemma is not to rely on a single memory component or technology, but to employ a memory hierarchy. A typical hierarchy is illustrated in Figure 4.1. As one goes down the hierarchy, the following occur:

a. Decreasing cost per bit;

b. Increasing capacity;

c. Increasing access time;

d. Decreasing frequency of access of the memory by the processor.

Thus, smaller, more expensive, faster memories are supplemented by larger, cheaper, slower memories. The key to the success of this organization.

Inboard Memory

- **Registers:** The fastest type of memory, directly accessible by the CPU. They hold data and instructions currently being processed.
- **Cache:** A small, high-speed memory that stores frequently accessed data and instructions from main memory. It improves performance by reducing the time it takes to fetch data.
- **Main Memory (RAM):** The primary storage area for data and programs. It's faster than secondary storage but slower than cache.

Outboard Storage

- **Magnetic Disk:** A secondary storage device that stores data magnetically on a rotating disk. Examples include hard disk drives (HDDs).
- **Optical Discs:** Storage devices that use laser technology to read and write data. Examples include CD-ROM, CD-RW, DVD-RW, DVD-RAM, and Blu-ray discs.

Off-line Storage

- **Magnetic Tape:** A sequential access storage medium used for long-term data storage and backup.

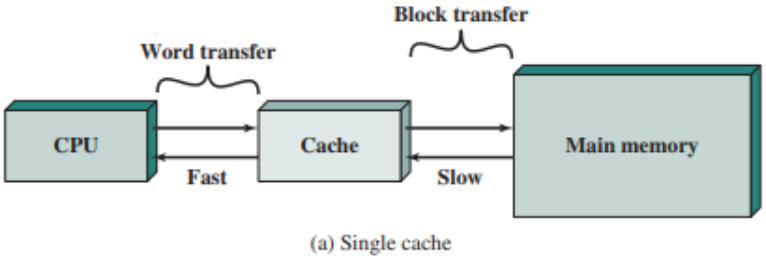
Key points about the memory hierarchy:

- **Speed:** The higher in the hierarchy, the faster the memory.
- **Cost:** The higher in the hierarchy, the more expensive the memory.
- **Capacity:** The lower in the hierarchy, the higher the capacity.

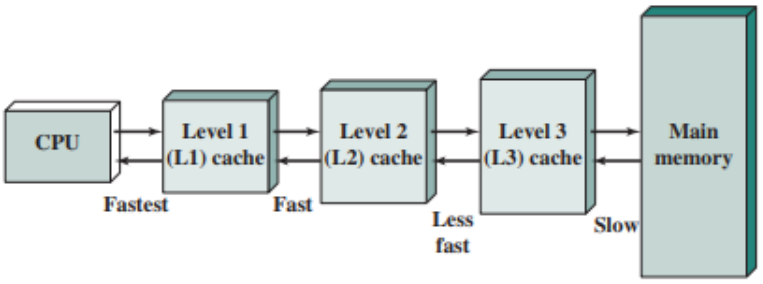
- **Access Time:** The time it takes to access data from a particular level of the hierarchy.
- **Data Transfer Rate:** The speed at which data can be transferred between levels of the hierarchy.

CACHE MEMORY PRINCIPLES :

Because of the phenomenon of locality of reference, when a block of data is fetched into the cache to satisfy a single memory reference, it is likely that there will be future references to that same memory location or to other words in the block.



(a) Single cache



(b) Three-level cache organization

Figure 4.3 Cache and Main Memory

a) Single Cache:

- **Simple structure:** The CPU directly interacts with a single cache level.
- **Data transfer:** Data is transferred between the CPU and cache in units of words or blocks.
- **Performance:** The cache's speed significantly affects the overall system performance.

b) Three-Level Cache Organization:

- **Multiple cache levels:** The CPU interacts with multiple cache levels (L1, L2, L3).
- **Hierarchy:** Each level is smaller and faster than the previous one.
- **Data transfer:** Data is transferred between levels based on proximity and access patterns.

- **Improved performance:** The three-level hierarchy can improve performance by reducing the average access time for frequently used data.

Cache/Main Memory Structure

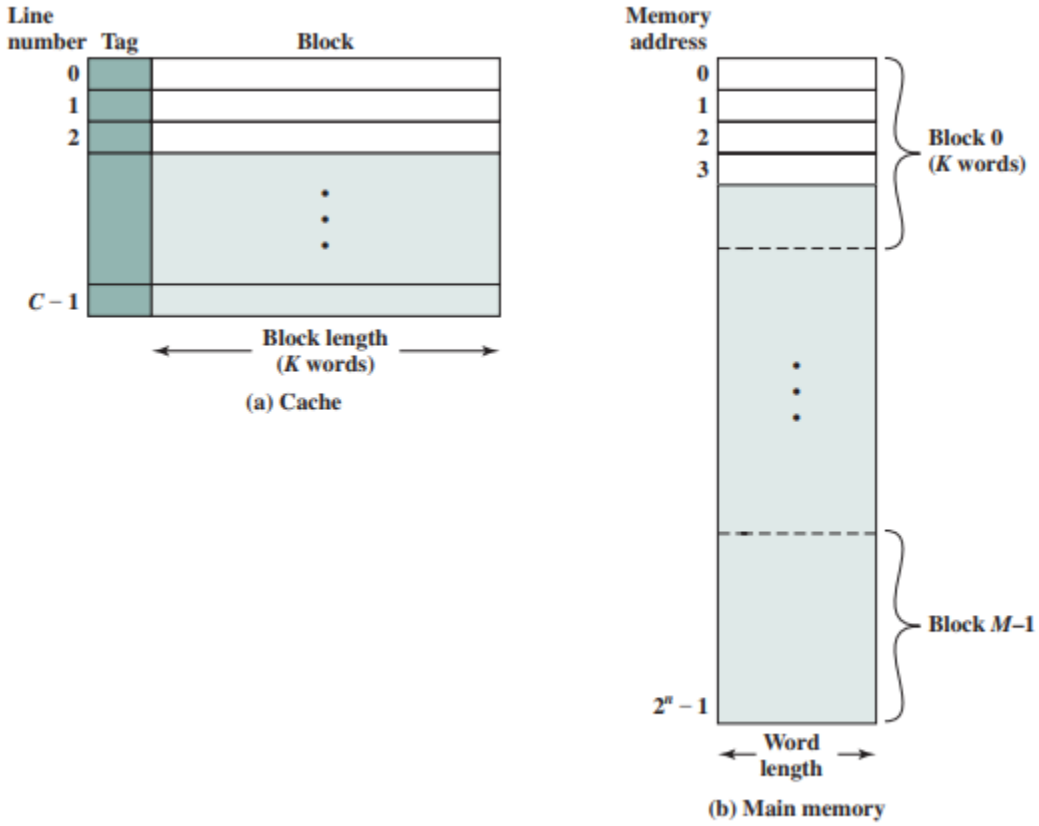


Figure 4.4 Cache/Main Memory Structure

a) Cache:

- **Line number:** Each cache line has a unique number.
- **Tag:** A portion of the memory address used to identify the corresponding block in main memory.
- **Block:** A contiguous block of data stored in the cache.
- **Block length:** The size of each cache block, typically measured in words.

b) Main Memory:

- **Memory address:** Each memory location has a unique address.

- **Word length:** The size of a single word in memory.
- **Block M-1:** The last block in main memory.

Cache Read Operation :-

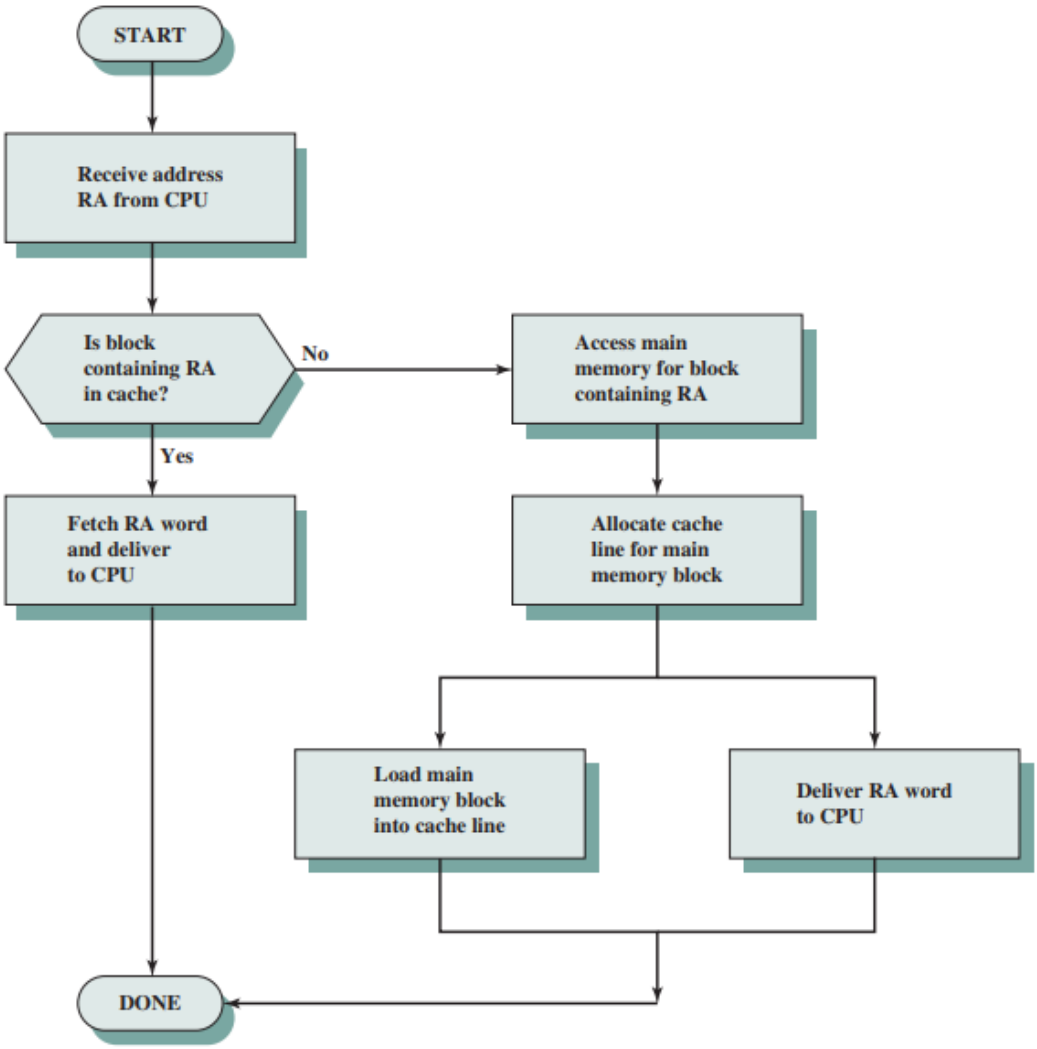


Figure 4.5 Cache Read Operation

 Receive address RA from CPU: The CPU sends a memory address (RA) to the cache.

 Is block containing RA in cache?

- **Yes:** If the cache block containing the requested address is already present in the cache, the operation proceeds to step 3.
- **No:** If the block is not in the cache, the operation proceeds to step 4.

 Fetch RA word and deliver to CPU: The requested word is retrieved from the cache and delivered to the CPU.
 Access main memory for block containing RA: The main memory is accessed to retrieve the entire block containing the requested address.
 Allocate cache line for main memory block: A cache line is allocated to store the retrieved block from main memory.
 Load main memory block into cache line: The retrieved block is loaded into the allocated cache line.
 Deliver RA word to CPU: The requested word is extracted from the newly loaded block and delivered to the CPU.

Typical Cache Organization :-

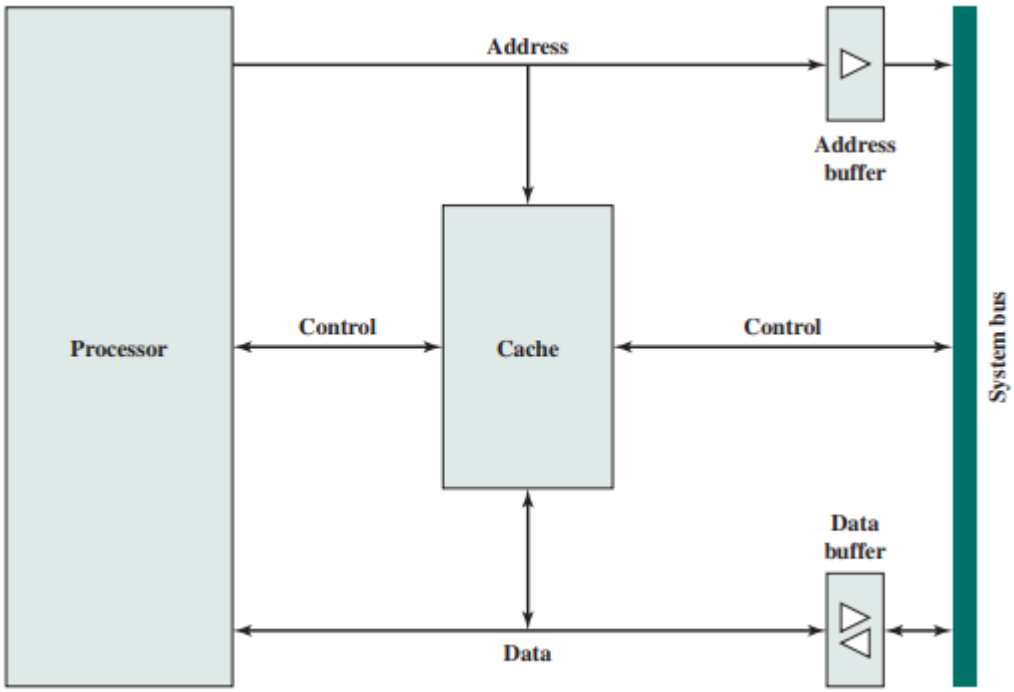


Figure 4.6 Typical Cache Organization

- **Processor:** The central processing unit of the system.
- **Control:** The control unit responsible for managing the cache operations.
- **Cache:** The high-speed memory that stores frequently accessed data.
- **Address buffer:** A buffer that holds the memory address being accessed.
- **Data buffer:** A buffer that holds the data being transferred between the processor and the cache.

- **System bus:** The communication channel connecting the processor, cache, and other components of the system.

ELEMENTS OF CACHE DESIGN :

Cache Size :- We would like the size of the cache to be small enough so that the overall average cost per bit is close to that of main memory alone and large enough so that the overall average access time is close to that of the cache alone.

Mapping Function :- Because there are fewer cache lines than main memory blocks, an algorithm is needed for mapping main memory blocks into cache lines.

Direct mapping:- The simplest technique, known as direct mapping, maps each block

of main memory into only one possible cache line. The mapping is expressed as

i = j modulo m

where

i = cache line number

j = main memory block number

m = number of lines in the cache

Direct-Mapping Cache Organization :-

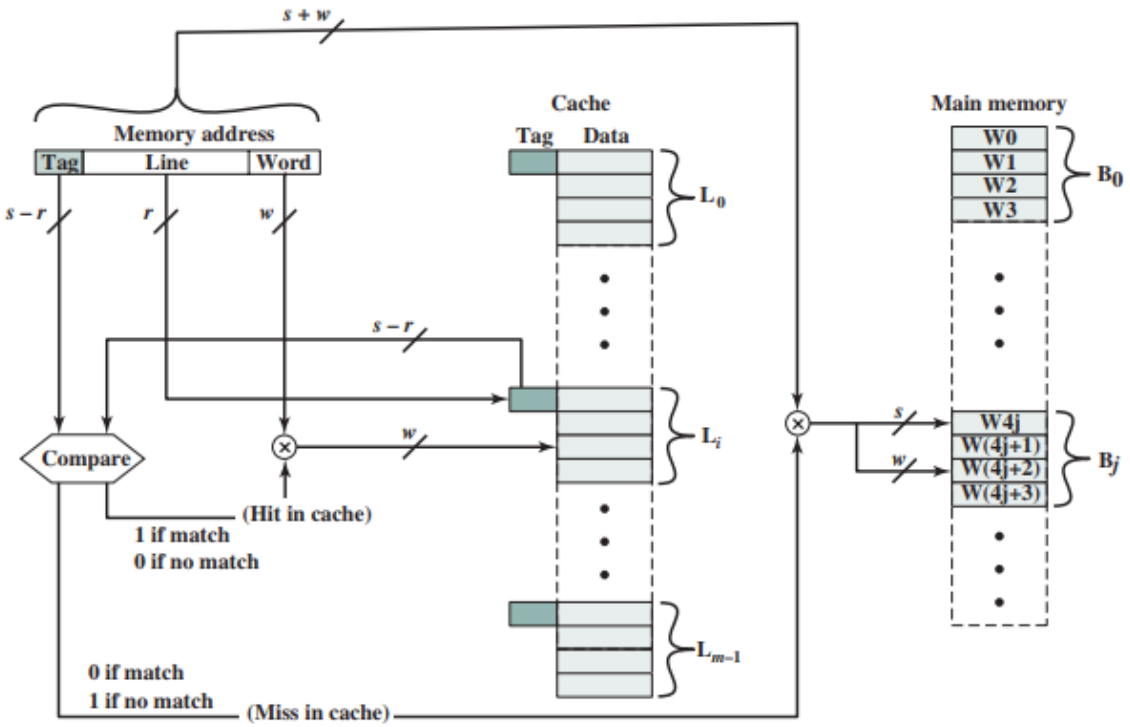


Figure 4.9 Direct-Mapping Cache Organization

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 2s 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 = 2r lines of the cache. To summarize.

■ Address length = (s + w) bits

■ Number of addressable units = 2s+w words or bytes

■ Block size = line size = 2w words or bytes

■ Number of blocks in main memory = 2s+w

2w = 2s

■ Number of lines in cache = m = 2r

■ Size of cache = 2r+w words or bytes

■ Size of tag = (s - r) bits

Components:

- **Memory address:** The address of the data being accessed.
- **Tag:** A portion of the memory address used to identify the corresponding cache line.
- **Line:** A unit of storage within the cache.
- **Word:** A unit of data (e.g., a byte or a group of bytes).
- **Cache:** The high-speed memory that stores frequently accessed data.
- **Main memory:** The primary storage device for data.

Operation:

1. **Receive memory address:** The cache receives the memory address of the data to be accessed.
2. **Extract tag and line number:** The memory address is divided into a tag and a line number.
3. **Compare tag:** The tag is compared with the tags of all cache lines.
4. **Hit/miss:**
    - **Hit:** If a match is found, the requested word is retrieved from the corresponding cache line and delivered to the processor.
    - **Miss:** If no match is found, a cache miss occurs, and the requested word is fetched from main memory.
5. **Load block:** If a miss occurs, the entire block containing the requested word is loaded from main memory into the cache line determined by the line number.

EXAMPLE 4.2a Figure 4.10 shows our example system using direct mapping.[5] In the example, $m = 16K = 2^{14}$ and $i = j$ modulo 2^{14}. The mapping becomes

Cache Line	Starting Memory Address of Block
0	000000, 010000, …, FF0000
1	000004, 010004, …, FF0004
⋮	⋮
$2^{14} - 1$	00FFFC, 01FFFC, …, FFFFFC

Note that no two blocks that map into the same line number have the same tag number. Thus, blocks with starting addresses 000000, 010000, …, FF0000 have tag numbers 00, 01, …, FF, respectively.

Referring back to Figure 4.5, a read operation works as follows. The cache system is presented with a 24-bit address. The 14-bit line number is used as an index into the cache to access a particular line. If the 8-bit tag number matches the tag number currently stored in that line, then the 2-bit word number is used to select one of the 4 bytes in that line. Otherwise, the 22-bit tag-plus-line field is used to fetch a block from main memory. The actual address that is used for the fetch is the 22-bit tag-plus-line concatenated with two 0 bits, so that 4 bytes are fetched starting on a block boundary.

Fully Associative Cache Organization :

Associative mapping :- Associative mapping overcomes the disadvantage of direct mapping by permitting each main memory block to be loaded into any line of the cache

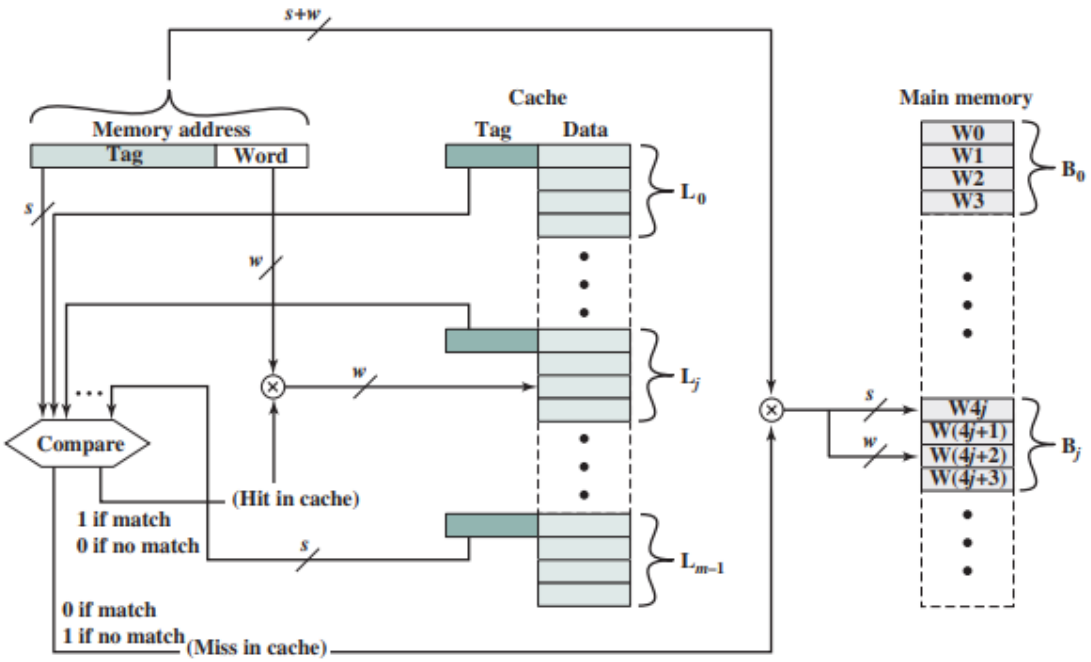


Figure 4.11 Fully Associative Cache Organization

Note that no field in the address corresponds to the line number, so that the number of lines in the cache is not determined by the address format. To summarize,

- Address length $= (s + w)$ bits
- Number of addressable units $= 2^{s+w}$ words or bytes
- Block size = line size $= 2^w$ words or bytes
- Number of blocks in main memory $= \frac{2^{s+w}}{2^w} = 2^s$
- Number of lines in cache = undetermined
- Size of tag $= s$ bits

Components:

- **Memory address:** The address of the data being accessed.
- **Tag:** A portion of the memory address used to identify the corresponding cache line.
- **Word:** A unit of data (e.g., a byte or a group of bytes).

- **Cache:** The high-speed memory that stores frequently accessed data.
- **Main memory:** The primary storage device for data.

Operation:

1. **Receive memory address:** The cache receives the memory address of the data to be accessed.
2. **Extract tag:** The tag is extracted from the memory address.
3. **Compare tag:** The tag is compared with the tags of all cache lines.
4. **Hit/miss:**
    - **Hit:** If a match is found, the requested word is retrieved from the corresponding cache line and delivered to the processor.
    - **Miss:** If no match is found, a cache miss occurs, and the requested word is fetched from main memory.
5. **Load block:** If a miss occurs, the entire block containing the requested word is loaded from main memory into a vacant cache line.

SET-ASSOCIATIVE MAPPING Set-associative mapping is a compromise that exhibits the strengths of both the direct and associative approaches while reducing their disadvantages.

In this case, the cache consists of number sets, each of which consists of a number of lines. The relationships are

$$m = v \times k$$
$$i = j \text{ modulo } v$$

where

i = cache set number
j = main memory block number
m = number of lines in the cache
v = number of sets
k = number of lines in each set

k-Way Set-Associative Cache Organization

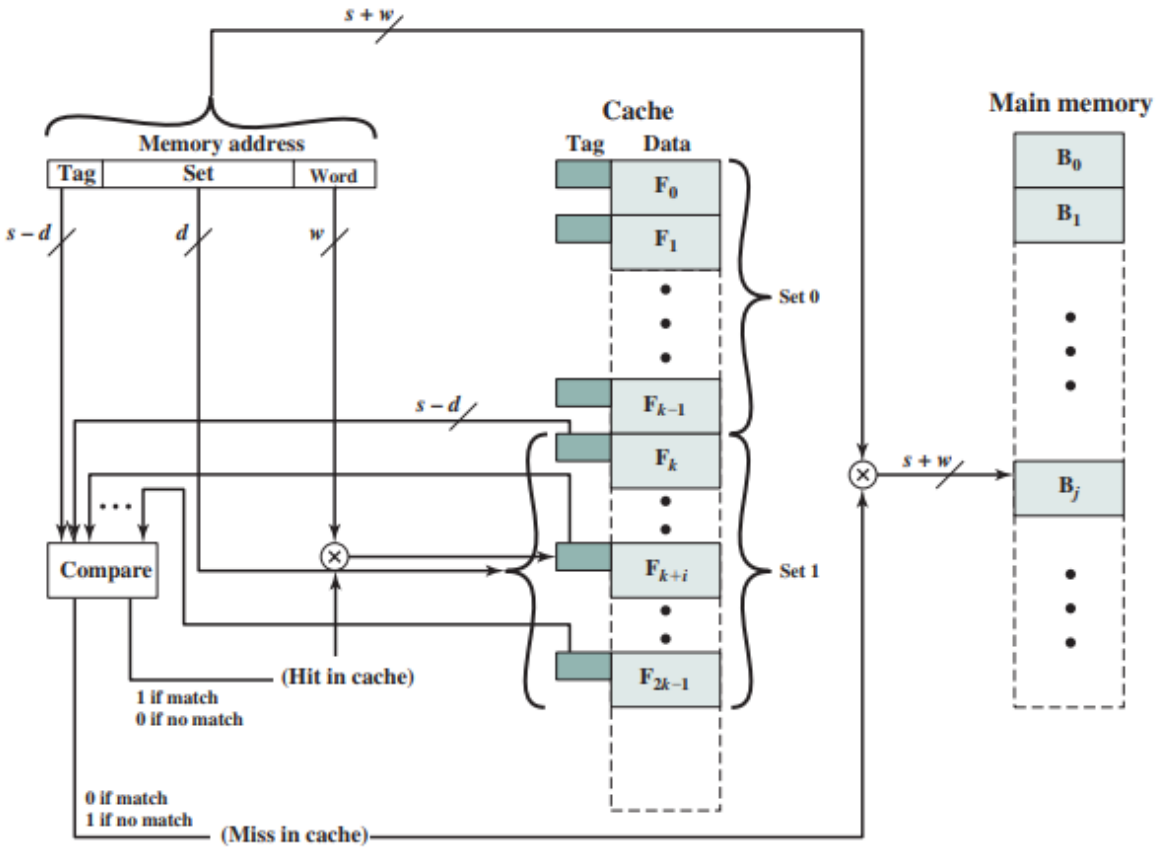


Figure 4.14 *k*-Way Set-Associative Cache Organization

Components:

- **Memory address:** The address of the data being accessed.
- **Tag:** A portion of the memory address used to identify the corresponding cache set.
- **Set:** A group of k cache lines.
- **Word:** A unit of data (e.g., a byte or a group of bytes).
- **Cache:** The high-speed memory that stores frequently accessed data.
- **Main memory:** The primary storage device for data.

Operation:

1. **Receive memory address:** The cache receives the memory address of the data to be accessed.
2. **Extract tag and set index:** The memory address is divided into a tag and a set index.
3. **Compare tag:** The tag is compared with the tags of all cache lines within the designated set.
4. **Hit/miss:**
    - **Hit:** If a match is found, the requested word is retrieved from the corresponding cache line and delivered to the processor.

- **Miss:** If no match is found, a cache miss occurs, and the requested word is fetched from main memory.

5. **Load block:** If a miss occurs, the entire block containing the requested word is loaded from main memory into a vacant cache line within the designated set.

- Block size = line size = 2^w words or bytes
- Number of blocks in main memory = $\frac{2^{s+w}}{2^w} = 2^s$
- Number of lines in set = k
- Number of sets = $v = 2^d$
- Number of lines in cache = $m = kv = k \times 2^d$
- Size of cache = $k \times 2^{d+w}$ words or bytes
- Size of tag = $(s - d)$ bits

EXAMPLE 4.2c Figure 4.15 shows our example using set-associative mapping with two lines in each set, referred to as two-way set-associative. The 13-bit set number identifies a unique set of two lines within the cache. It also gives the number of the block in main memory, modulo 2^{13}. This determines the mapping of blocks into lines. Thus, blocks 000000, 008000, …, FF8000 of main memory map into cache set 0. Any of those blocks can be loaded into either of the two lines in the set. Note that no two blocks that map into the same cache set have the same tag number. For a read operation, the 13-bit set number is used to determine which set of two lines is to be examined. Both lines in the set are examined for a match with the tag number of the address to be accessed.

* Replacement Algorithms

In direct mapping, no replacement algo reqd.

Reqd for associative & set associative

1) LRU
2) FIFO
3) LFU

1) LRU - Least Recently Used

e.g. i)

MM
A_1
A_2
A_3
A_4
A_5

Cache (MRU → LRU)

A_1

ii) → A_1, A_2, A_3, A_4, A_5

iii) → A_1, A_3, A_4, A_5

Cache: A_3, A_2, A_1

iv)

MM
A_1
A_2
A_3
A_4
A_5

Cache: A_4, A_3, A_2

2) FIFO - First In First Out

2, 3, 4, 7, 6, 3, 4, 7, 5, 4, 7, 8 → main memory.
M M M M M H H H M H H M

0	2	0	6	0	6	0	6
1	3	1	3	1	5	1	5
2	4	2	4	2	4	2	8
3	7	3	7	3	7	3	7

Cache — ″ — — ″ — — ″ —

3) LFU – Least Frequently Used
Replace the block in the set that has experienced fewest references

e.g. main memory | 0 | 1 | 2 | 3 | 4 | 2 | 3 | 1 | 5 | 6 |

Cache		frequency	C		f	C		C	
0	0	1	0	4	1	0	5	1	0	6	1
1	1	1	1	1	11	1	1	2	1	1	2
2	2	1	2	2	11	2	2	2	2	2	2
3	3	1	3	3	11	3	3	2	3	3	2

I/O Modules :

Module Function

The major functions or requirements for an I/O module fall into the following categories:

- Control and timing
- Processor communication
- Device communication
- Data buffering
- Error detection

Control and timing :- To coordinate the flow of traffic between internal resources and external devices. For example, the control of the transfer of data from an external device to the processor might involve the following sequence of steps:

1. The processor interrogates the I/O module to check the status of the attached device.
2. The I/O module returns the device status.
3. If the device is operational and ready to transmit, the processor requests the transfer of data, by means of a command to the I/O module.
4. The I/O module obtains a unit of data (e.g., 8 or 16 bits) from the external device.
5. The data are transferred from the I/O module to the processor.

Processor communication involves the following operations:

■ **Command decoding**: The I/O module accepts commands from the processor, typically sent as signals on the control bus. For example, an I/O module for a disk drive might accept the following commands: READ SECTOR, WRITE SECTOR, SEEK track number, and SCAN record ID. The latter two commands each include a parameter that is sent on the data bus.

■ **Data**: Data are exchanged between the processor and the I/O module over the data bus.

■ **Status reporting**: Because peripherals are so slow, it is important to know the status of the I/O module. For example, if an I/O module is asked to send data to the processor (read), it may not be ready to do so because it is still working on the previous I/O command. This fact can be reported with a status signal. Common status signals are BUSY and READY. There may also be signals to report various error conditions.

■ **Address recognition**: Just as each word of memory has an address, so does each I/O device. Thus, an I/O module must recognize one unique address for each peripheral it controls.

The I/O module must be able to perform **Device communication**. This communication involves commands

Data buffering :- Whereas the transfer rate into and out of main memory or the processor is quite high, the rate is orders of magnitude lower for many peripheral devices and covers a wide range. Data coming from main memory are sent to an I/O module in a rapid burst. The data are buffered in the I/O module and then sent to the peripheral device at its data rate. In the opposite direction, data are buffered so as not to tie up the memory in a slow transfer operation.

Error detection :- I/O module is often responsible for error detection and for subsequently reporting errors to the processor. One class of errors includes mechanical and electrical malfunctions reported by the device (e.g., paper jam, bad disk track). Another class consists of unintentional changes to the bit pattern as it is transmitted from device to I/O module. Some form of error- detecting code is often used to detect transmission errors.

Block Diagram of an I/O Module :-

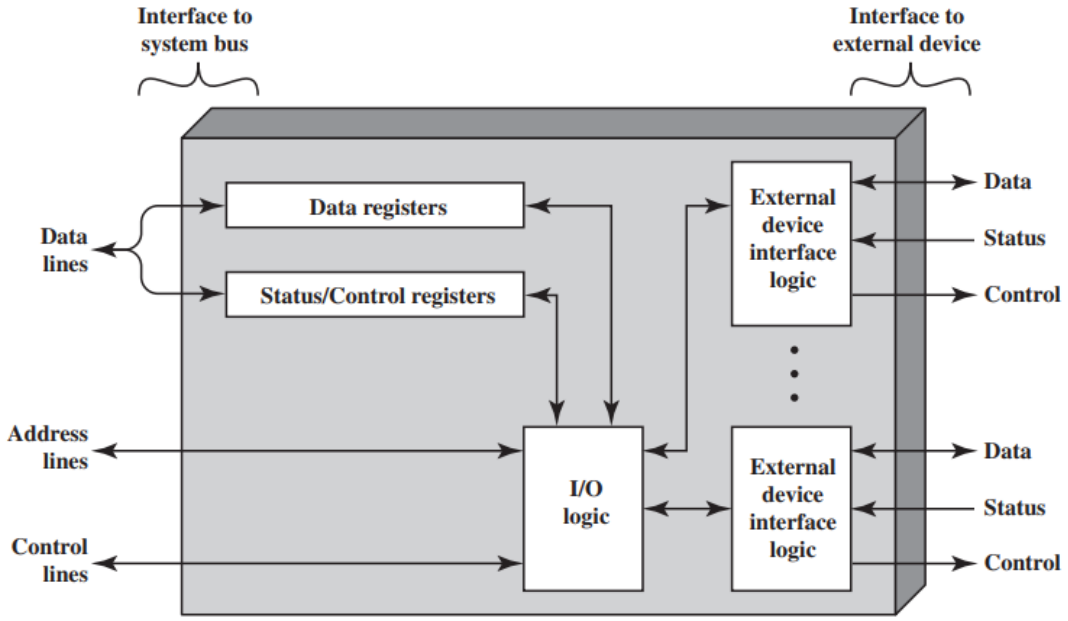


Figure 7.3 Block Diagram of an I/O Module

1. Interface to system bus:

- This component connects the I/O module to the system bus, allowing it to communicate with other components like the CPU and memory.
- It handles the transfer of data, addresses, and control signals between the I/O module and the system bus.

2. Data registers:

- These registers store data that is being transferred between the I/O module and the external device.
- They act as temporary holding places for data during input/output operations.

3. Status/Control registers:

- These registers provide information about the status of the I/O module and the external device.
- They also allow the CPU to control the operation of the I/O module and the external device.

4. I/O logic:

- This component performs the actual input/output operations.

- It controls the flow of data between the I/O module and the external device.
- It also handles the timing and synchronization of input/output operations.

5. External device interface logic:

- This component is specific to the type of external device being connected to the I/O module.
- It provides the necessary interface for communication with the external device.
- It translates the data and control signals between the I/O module and the external device.

6. Data lines:

- These lines carry data between the I/O module and the external device.
- They can be bidirectional, allowing data to flow in both directions.

7. Address lines:

- These lines are used to select a specific device or register within the I/O module.
- They are provided by the system bus and are decoded by the I/O module to identify the target component.

8. Control lines:

- These lines are used to control the operation of the I/O module and the external device.
- They can be used to initiate input/output operations, signal the completion of operations, and indicate error conditions.

Techniques for I/O operations :

1) Programmed I/O
2) Memory Mapped I/O
3) Interrupt Driven I/O
4) Direct Memory Access

Programmed I/O :-

- PIO is a method where the processor directly controls I/O operations.
- When the processor encounters an I/O instruction, it sends a command to the appropriate I/O module.
- The I/O module executes the command and updates its status register to indicate the operation's completion.
- However, the I/O module doesn't actively notify the processor.

- The processor must periodically check the status register to determine if the I/O operation is finished.

Memory Mapped I/O:- When the processor, main memory, and I/O share a common bus, two modes of addressing are possible:

1. memory mapped :- With memory- mapped I/O, there is a single address space for memory locations and I/O devices
2. isolated I/O

Interrupt-Driven I/O :- The following sequence of hardware events occurs:

1. The device issues an interrupt signal to the processor.

2. The processor finishes execution of the current instruction before responding

to the interrupt, as indicated in Figure 3.9.

3. The processor tests for an interrupt, determines that there is one, and sends an

acknowledgment signal to the device that issued the interrupt. The acknowledgment allows the device to remove its interrupt signal.

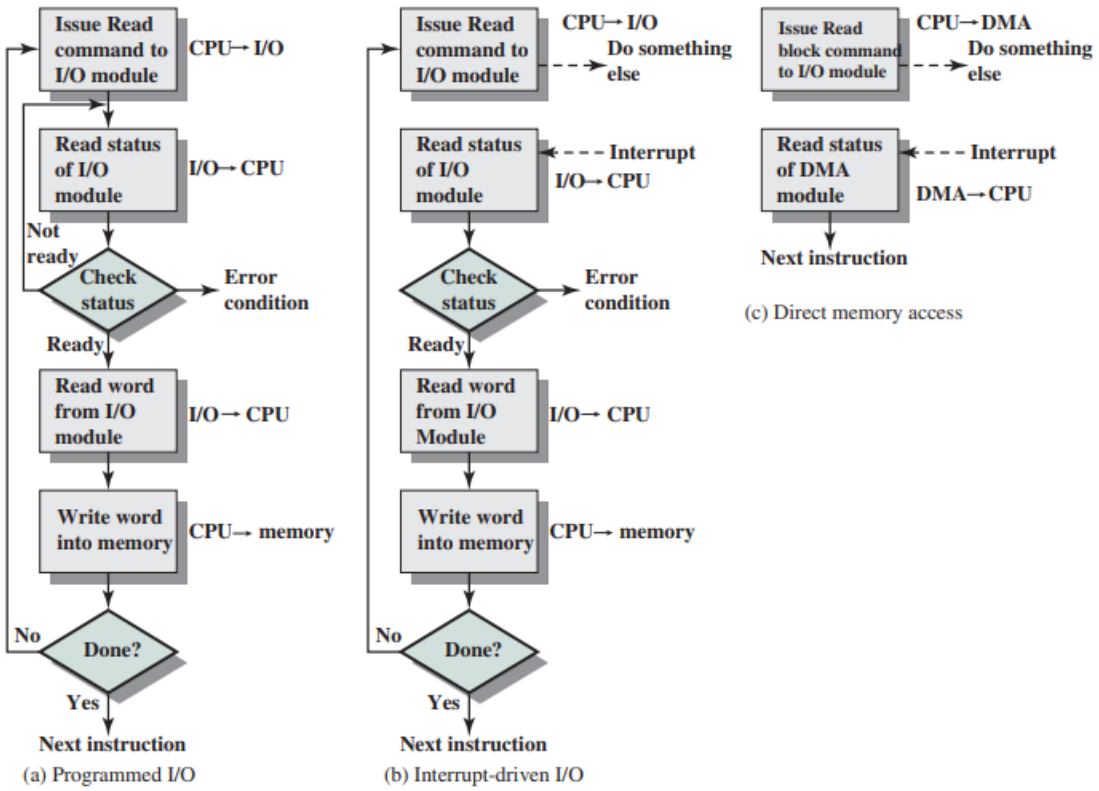


Figure 7.4 Three Techniques for Input of a Block of Data

Direct memory access :-The alternative is known as direct memory access (DMA). In this mode, the I/O module and main memory exchange data directly, without processor involvement.

Drawbacks of Programmed and Interrupt-Driven I/O :

1. The I/O transfer rate is limited by the speed with which the processor can test and service a device.

2. The processor is tied up in managing an I/O transfer; a number of instructions must be executed for each I/O transfer

Direct memory access (DMA)

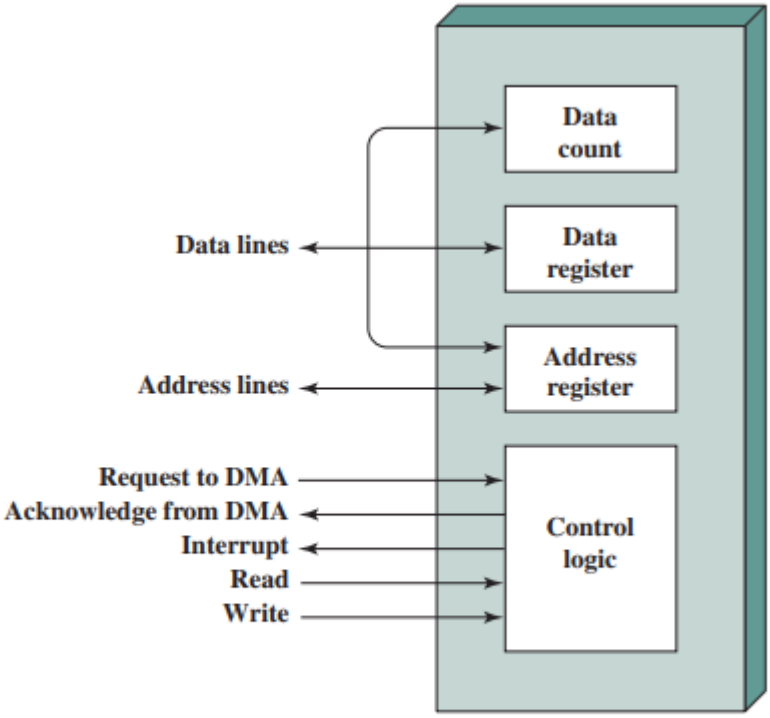


Figure 7.12 Typical DMA Block Diagram

- **Data Count**:
  - Keeps track of the number of data units that need to be transferred between memory and an I/O device. This value is decremented after each data transfer.
- **Data Register**:

- Temporarily holds the data that is either being read from memory or written to memory during a transfer operation. It acts as a buffer between memory and the device.

- **Address Register**:
  - Holds the address of the memory location where data is to be read from or written to. The address is automatically updated as each transfer occurs.

- **Control Logic**:
  - Manages the overall operation of the DMA controller, handling all signals and coordination between the memory and I/O device. The control logic ensures proper timing and data handling.

- **Data Lines**:
  - These are the pathways through which actual data is transferred between the memory and the device.

- **Address Lines**:
  - These lines carry the memory addresses for data transfer, enabling proper location identification in the memory.

- **Request to DMA**:
  - A signal from the peripheral device to request the DMA controller to start the data transfer process.

- **Acknowledge from DMA**:
  - A signal sent from the DMA controller to the peripheral device, confirming that the request for data transfer has been acknowledged and the operation can begin.

- **Interrupt**:
  - A signal sent to the CPU when the DMA has completed the data transfer process. This frees up the CPU from managing the transfer and allows it to focus on other tasks.

- **Read and Write**:
  - Control signals indicating whether data is being read from or written to the memory. "Read" typically refers to reading from memory to the peripheral, while "Write" refers to writing data from the peripheral to memory.

Questions of Unit 3

1) List & access method of memory.

2) Find Transfer Code

3) Draw and Explain every Hierarchy

4) Explain characteristics of memory System.

5) Draw and explain Typical cache organization

6) List and explain I/O module Function

7) Explain block diagram of I/O module.

8) Explain cache read operations

9) Explain Techniques for I/O operation

10) Explain Programmed I/O

11) Difference between memory mapped & isolated I/O

----------END----------