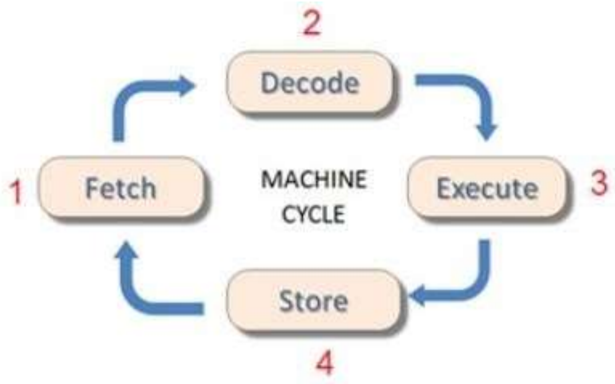
**Provide the common types of operands with an example.**

Operands are values used in arithmetic or logical operations.

**Example:** In the operation x = y + z
y and z are the operands. The computer retrieves their values from memory, performs the addition, and stores the result in x.

**Specify the role of the fetching phase in a machine cycle.**

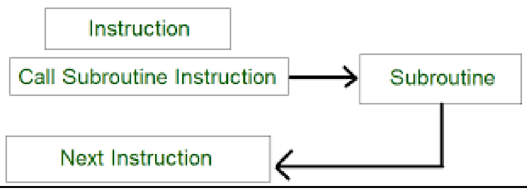


The CPU fetches **instruction and data** from memory.

**Highlight the advantages of hardwired realization in the control unit design of a microprocessor.**

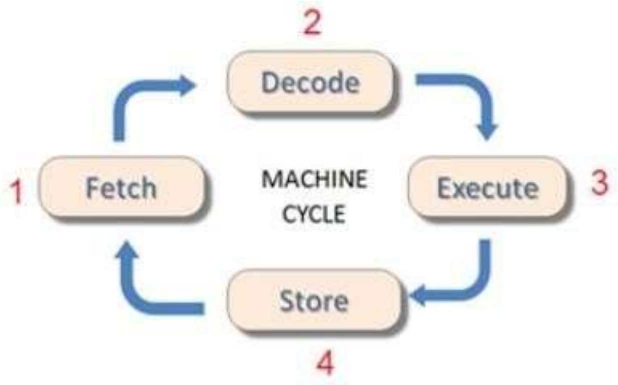
1. Speed
2. Low latency
3. Security
4. Parallelism

**Specify the purpose of subroutine call in computer programming.**



The purpose of a subroutine call in computer programming is to execute a specific sequence of code that performs a particular task or function.

**Explore the role of the decoding phase in a machine cycle.**



The control unit decodes the instruction.

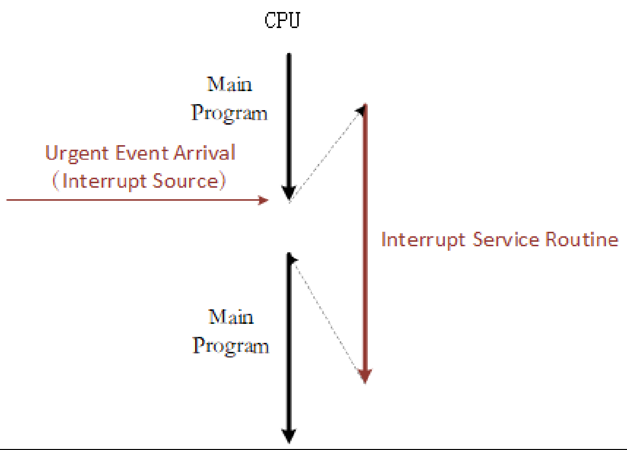
**Illustrate the cause of structural hazards in pipelining.**

1. Resource contention
2. Limited hardware resources
3. Instruction overlap.
4. Memory access conflicts

**Explore the significance of a microprocessor in modern computing devices.**

1. Processing Power
2. Versatility
3. Integration (System-on-Chip)
4. Energy Efficiency

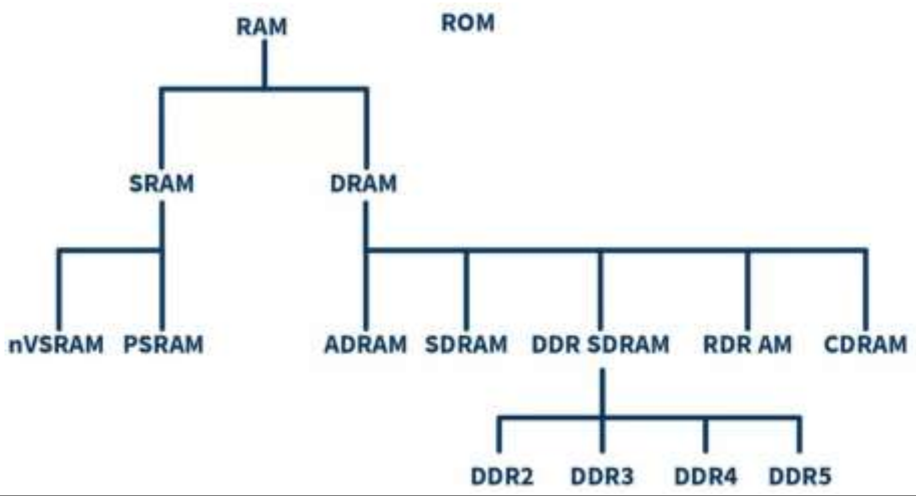
**Draw the illustration of subroutine call and return mechanism in computer programming.**



**List the characteristics of a multicycle implementation in processor design.**

1. Reuse of Hardware Components
2. Variable Execution Time
3. Control Logic Complexity
4. Efficiency and Performance

**Identify and categorize different types of RAM according to their characteristics.**



**Formulate cache performance and its purpose.**

1. Cache Performance is,

$$\text{Hit Ratio} = \frac{\text{Number of cache hits}}{\text{Number of searches}}$$

2. Cache memory speeds up computer programs by storing frequently accessed data in a faster location closer to the CPU.

**Summarize potential reasons for buffering in IOoperations.**

1. Hardware Limitations
2. Speed Mismatch
3. Resource Competition
4. Data Volume
5. Network and File System Factors

**Highlight various policies of cache data replacement.**

1. Least Recently Used (LRU)
2. First-In, First-Out (FIFO)
3. Random Replacement
4. Least Frequently Used (LFU)
5. Most Recently Used (MRU)

6. Optimal Replacement

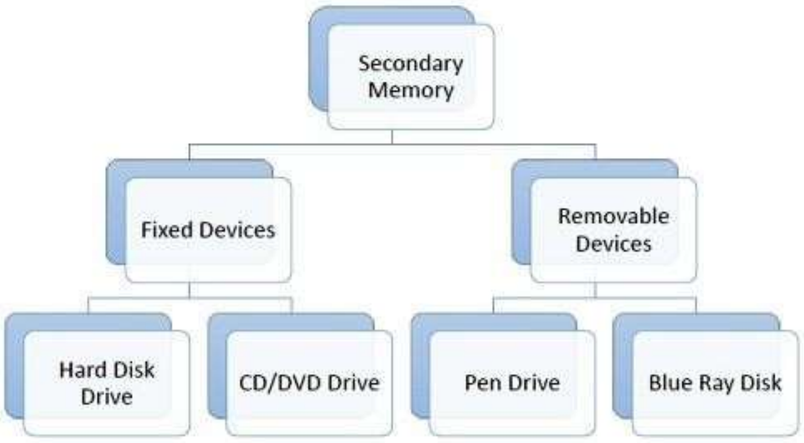
**List the various data transfer methods in IOcommunication.**

1. Programmed I/O (PIO)
2. Interrupt-Driven I/O
3. Direct Memory Access (DMA)
4. Memory Mapped I/O

**Represent the use of virtual memory in computersystem.**

1. Running larger programs
2. Multitasking Support
3. Memory Extension
4. Memory Sharing
5. Memory Protection

**Identify and list some of the secondary storage devices.**



**Specify the role of memory cell in the context of memory organization.**

Memory cells store binary data (0s and 1s) and serve as the fundamental units for data storage and retrieval in memory organization.

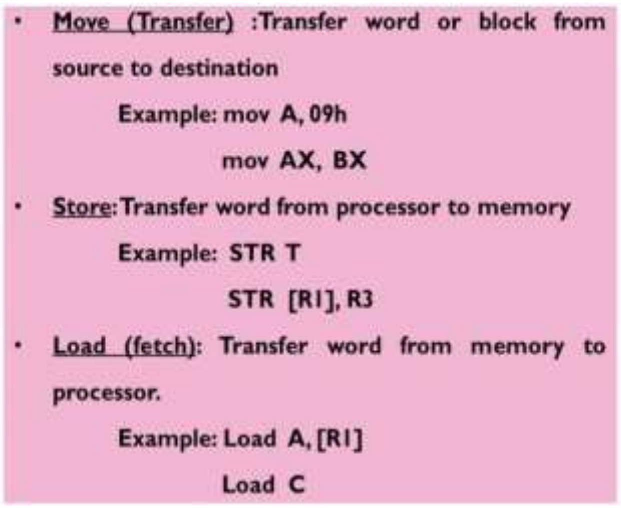
**Summarize various Asynchronous Data Transfer methods.**

1. **Strobe Control:** Syncs data transfer with one signal, indicating when to exchange data.
2. **Handshaking:** Sender says "ready," receiver confirms, ensuring reliable communication.

# DDCA
# CO-3 TERMINAL QUESTIONS & ANSWERS

**Discuss the data transfer and arithmetic logic instruction sets with examples.**

**Data Transfer Instructions:**



- Move (Transfer) :Transfer word or block from source to destination

  Example: mov A, 09h

  mov AX, BX
- Store: Transfer word from processor to memory

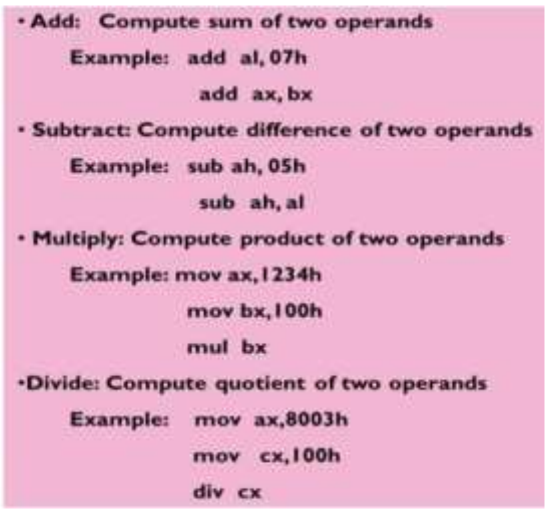
  Example: STR T

  STR [R1], R3
- Load (fetch): Transfer word from memory to processor.

  Example: Load A, [R1]

  Load C

**Arithmetic instruction:**



- Add: Compute sum of two operands

  Example: add al, 07h

  add ax, bx
- Subtract: Compute difference of two operands

  Example: sub ah, 05h

  sub ah, al
- Multiply: Compute product of two operands

  Example: mov ax, 1234h

  mov bx, 100h

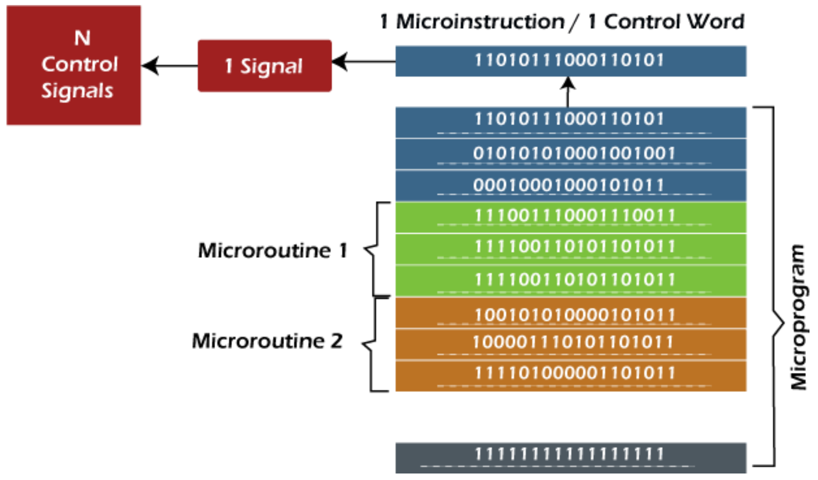
  mul bx
- Divide: Compute quotient of two operands

  Example: mov ax,8003h

  mov cx,100h

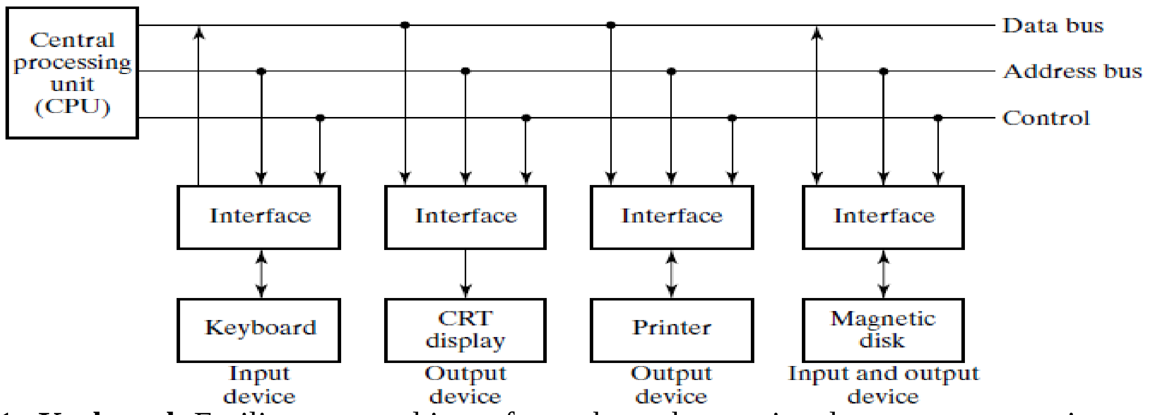
  div cx

**Illustrate the micro-programmed realization in CPU design, detailing its architecture.**



1. Micro-programmed control units operate as basic logic circuits.
2. They execute instructions by generating control signals and sequencing through microinstructions.
3. Microprograms stored in fast memory, termed control store or control memory, dictate the sequence of control signals for each instruction, facilitating efficient execution.

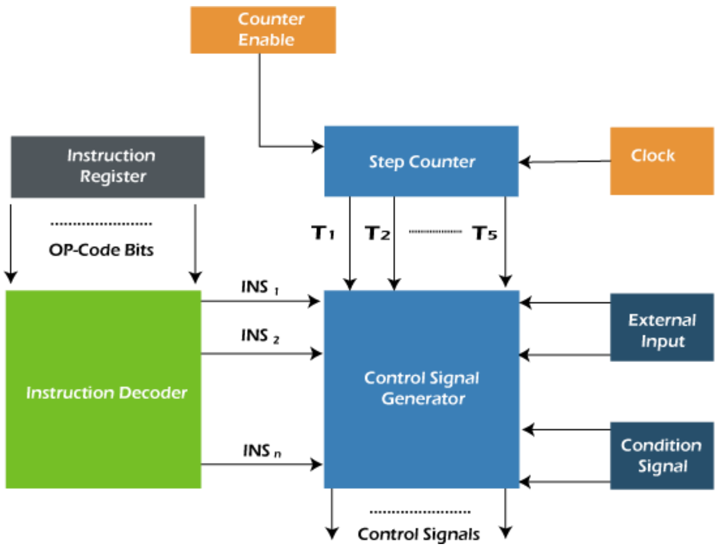
**Analyze the role and significance of IO devices in computer systems with examples.**



1. **Keyboard:** Facilitates textual input for tasks such as typing documents or entering commands.

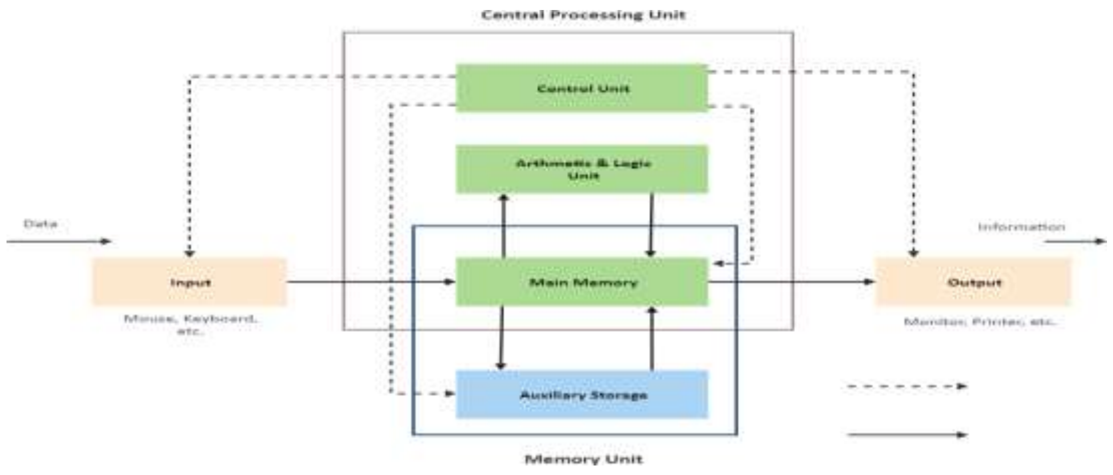
2. **Monitor:** Displays visual output, including text, images, videos, and graphical user interfaces.
3. **Printer:** Produces hard copies of digital documents or images for physical distribution or archival.
4. **Magnetic Disk:** Provides high-capacity, non-volatile storage for operating systems, applications, and user data, allowing quick access to stored information.

**Illustrate the hardwired realization in CPU design,detailing its architecture.**



1. Hardwired control units execute instructions by generating control signals at the right time and sequence.
2. Faster than micro-programmed units, they use PLA circuit and state counter to generate control signals.
3. Hardware-based, they employ circuitry to produce control signals needed by the CPU for operation.

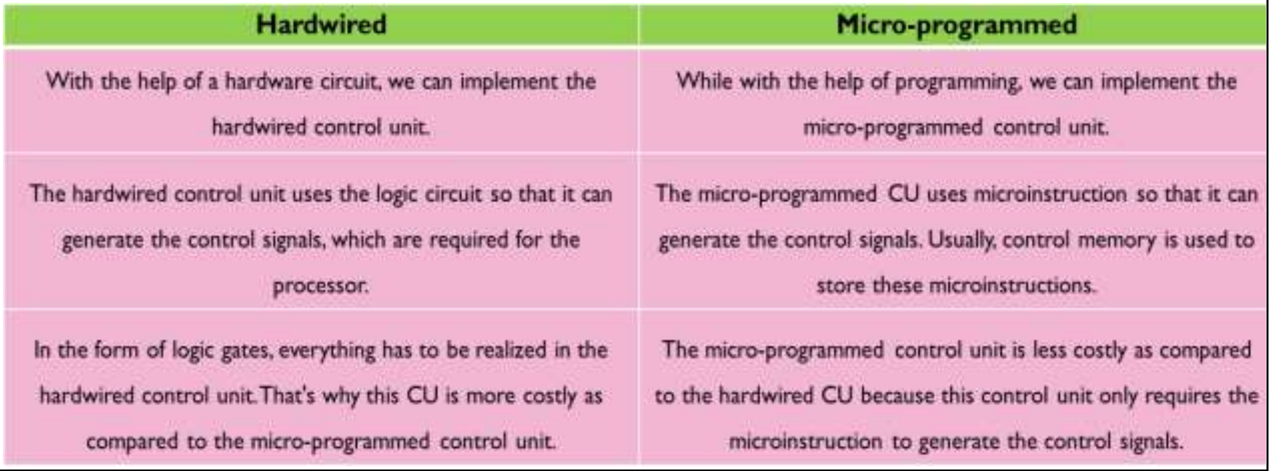
**Illustrate the architecture of a CPU and its constituent blocks, elaborating on their functions.**



1. **Control Unit (CU):** Fetches and decodes instructions, generates control signals for CPU operations.
2. **Arithmetic Logic Unit (ALU):** Performs arithmetic and logical operations on data as directed by the control unit.
3. **Memory Unit (MU):** Stores data and instructions, comprising internal registers for fast, temporary storage, and main memory for larger capacity storage.

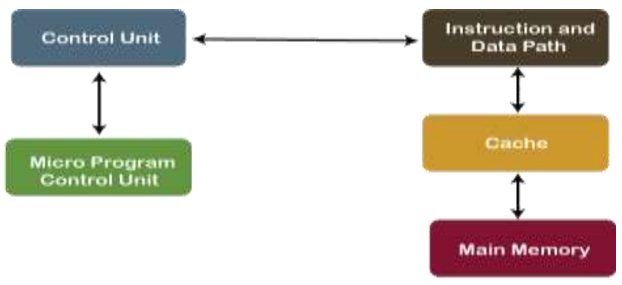
# DDCA
# CO-3 TERMINAL QUESTIONS & ANSWERS

**Differentiate hardwired realization and micro- programmed realization in the design of control units within a CPU.**



| Hardwired | Micro-programmed |
|---|---|
| With the help of a hardware circuit, we can implement the hardwired control unit. | While with the help of programming, we can implement the micro-programmed control unit. |
| The hardwired control unit uses the logic circuit so that it can generate the control signals, which are required for the processor. | The micro-programmed CU uses microinstruction so that it can generate the control signals. Usually, control memory is used to store these microinstructions. |
| In the form of logic gates, everything has to be realized in the hardwired control unit. That's why this CU is more costly as compared to the micro-programmed control unit. | The micro-programmed control unit is less costly as compared to the hardwired CU because this control unit only requires the microinstruction to generate the control signals. |

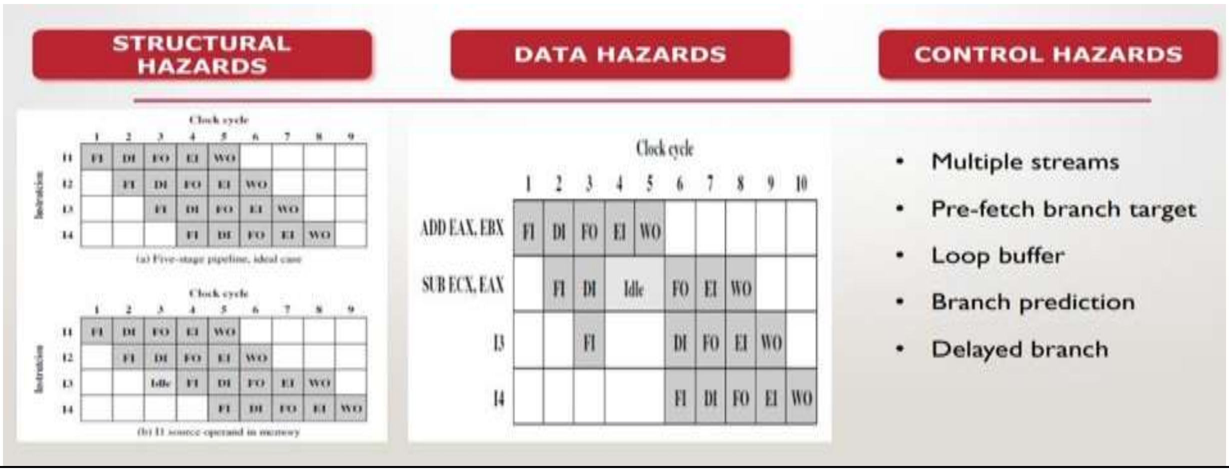
**Develop the model of CISC architecture, detailing its design philosophy, key features.**



- CISC architectures feature a large set of complex instructions.
- Designed to accomplish tasks in fewer lines of code, with instructions capable of performing multiple operations.
- Aims to reduce the software complexity by providing instructions that can execute high-level tasks directly.

CISC Architecture

**Identify the various pipelining hazards in processor design, detailing the types of hazards, their causes.**



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| I1 | FI | DI | FO | EI | WO | | | | |
| I2 | | FI | DI | FO | EI | WO | | | |
| I3 | | | FI | DI | FO | EI | WO | | |
| I4 | | | | FI | DI | FO | EI | WO | |

(a) Five-stage pipeline, ideal case

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| I1 | FI | DI | FO | EI | WO | | | | |
| I2 | | FI | DI | FO | EI | WO | | | |
| I3 | | | Idle | FI | DI | FO | EI | WO | |
| I4 | | | | | FI | DI | FO | EI | WO |

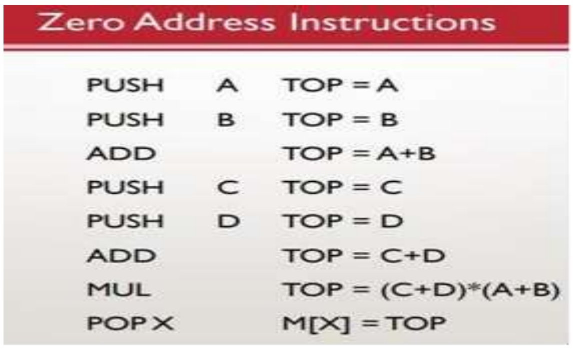
(b) I1 source operand in memory

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| ADD EAX, EBX | FI | DI | FO | EI | WO | | | | | |
| SUB ECX, EAX | | FI | DI | Idle | | FO | EI | WO | | |
| I3 | | | FI | | | DI | FO | EI | WO | |
| I4 | | | | | | FI | DI | FO | EI | WO |

- Multiple streams
- Pre-fetch branch target
- Loop buffer
- Branch prediction
- Delayed branch

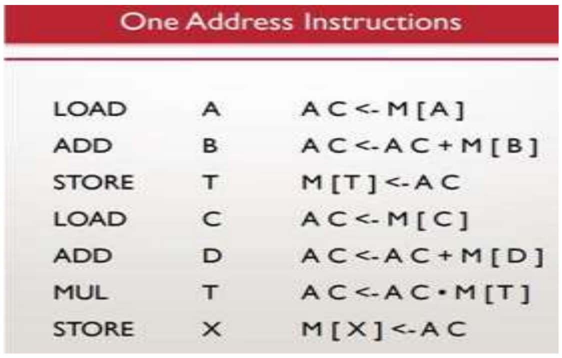
# DDCA
# CO-3 TERMINAL QUESTIONS & ANSWERS

**Model the structures of different instruction formats and provide illustrative examples foreach.**



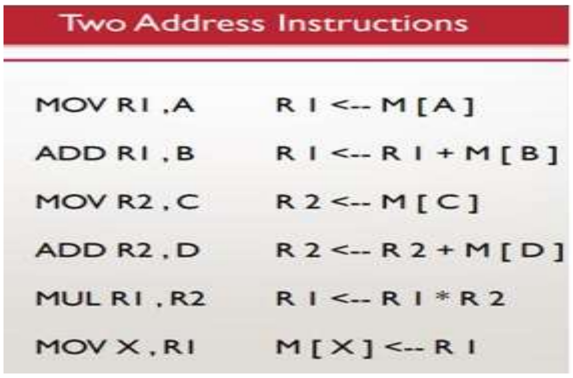
Zero Address Instructions

| | | |
|---|---|---|
| PUSH | A | TOP = A |
| PUSH | B | TOP = B |
| ADD | | TOP = A+B |
| PUSH | C | TOP = C |
| PUSH | D | TOP = D |
| ADD | | TOP = C+D |
| MUL | | TOP = (C+D)*(A+B) |
| POP X | | M[X] = TOP |



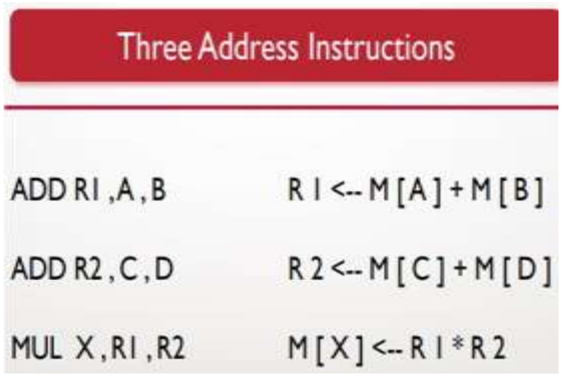
One Address Instructions

| | | |
|---|---|---|
| LOAD | A | AC <- M[A] |
| ADD | B | AC <- AC + M[B] |
| STORE | T | M[T] <- AC |
| LOAD | C | AC <- M[C] |
| ADD | D | AC <- AC + M[D] |
| MUL | T | AC <- AC • M[T] |
| STORE | X | M[X] <- AC |



Two Address Instructions

| | |
|---|---|
| MOV R1 ,A | R1 <-- M[A] |
| ADD R1 ,B | R1 <-- R1 + M[B] |
| MOV R2 ,C | R2 <-- M[C] |
| ADD R2 ,D | R2 <-- R2 + M[D] |
| MUL R1 ,R2 | R1 <-- R1 * R2 |
| MOV X ,R1 | M[X] <-- R1 |



Three Address Instructions

| | |
|---|---|
| ADD R1 ,A ,B | R1 <-- M[A] + M[B] |
| ADD R2 ,C ,D | R2 <-- M[C] + M[D] |
| MUL X ,R1 ,R2 | M[X] <-- R1 * R2 |

**Consider a scenario where you are developing a simple combinational circuit of basic logical operations. Make use of the related instruction sets and accomplish the task.**

Let us consider the expression $F = A.B + C'.D$ to develop using the Logical instruction set.

**Load input values**

LOAD R0, A ; Load input A into register R0

LOAD R1, B ; Load input B into register R1

LOAD R2, C ; Load input C into register R2

LOAD R3, D ; Load input D into register R3

**Perform logical operations**

AND R5, R0, R1 ; Compute A AND B and store the result in R5

NOT R6, R2 ; Compute the complement of C and store it in R6

AND R7, R6, R3 ; Compute (NOT C) AND D and store the result in R7

OR R4, R5, R7 ; Compute (A AND B) OR ((NOT C) AND D) and store the result in R4

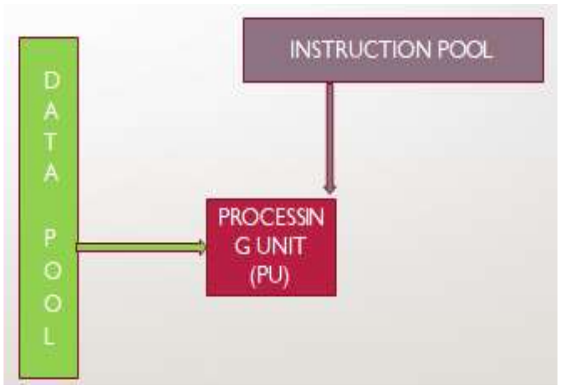
**In order to determine the city with the most consistently warm temperatures from a large weatherdataset, which parallel processing architecture (SISD,SIMD, MISD, MIMD) would be best suited for efficient analysis and why?**

For determine the city with the most consistently warm temperatures from a large weather dataset, most appropriate architecture would be MIMD (Multiple Instruction, Multiple Data) due to following reasons.

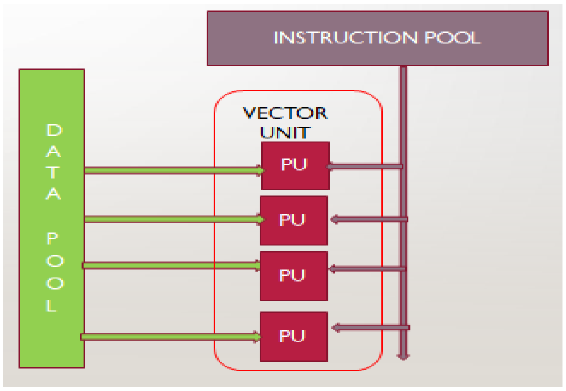
- ✓ Independent Processing

- ✓ Flexibility and Scalability
- ✓ Complexity of Analysis
- ✓ Data Distribution and Communication

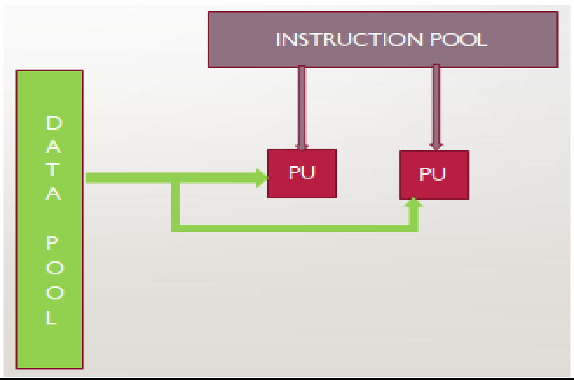
**SISD**



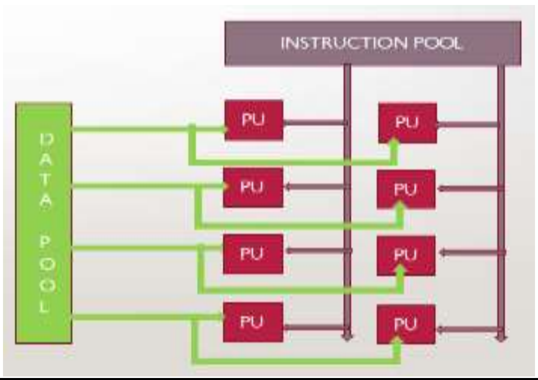
**SIMD**



**MISD**

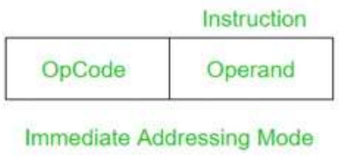


**MIMD**



**Examine the concepts of immediate, direct, and indirect addressing modes in computer architecture.**

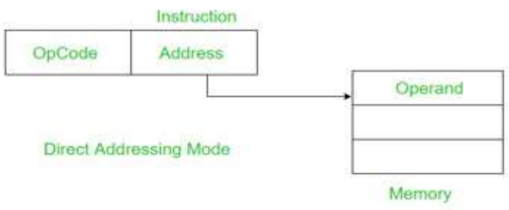
1. **Immediate Addressing mode:**
   Eg: MOV AX, 2000H
   ADD AL, 45H



2. **Direct Addressing mode:**

   Eg: MOV AX, [1592H]

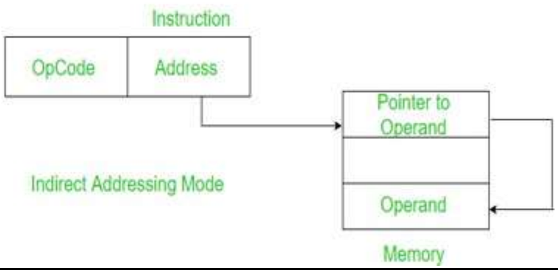
   MOV BL, [03H]



3. **Indirect Addressing mode:**

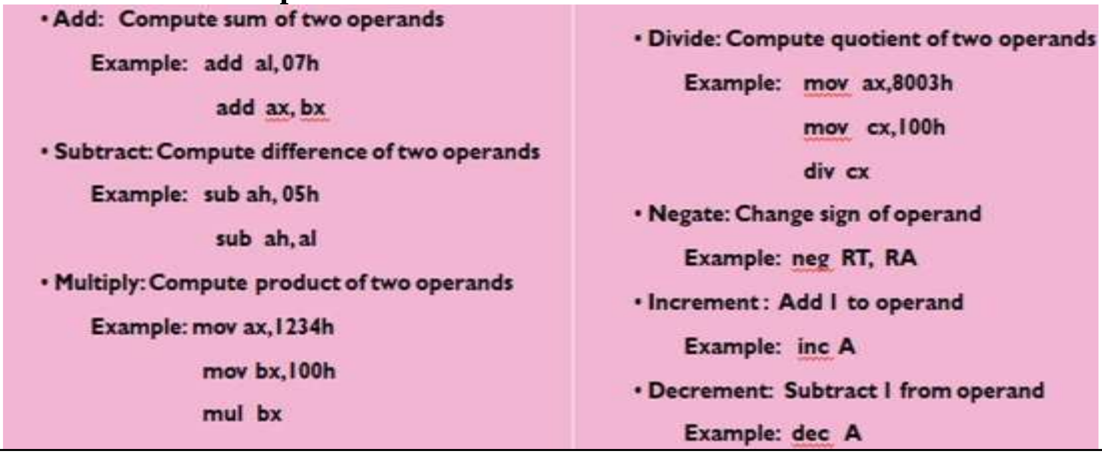
   Eg: MOV AX, @2005H

   LOAD R1, (1345H)



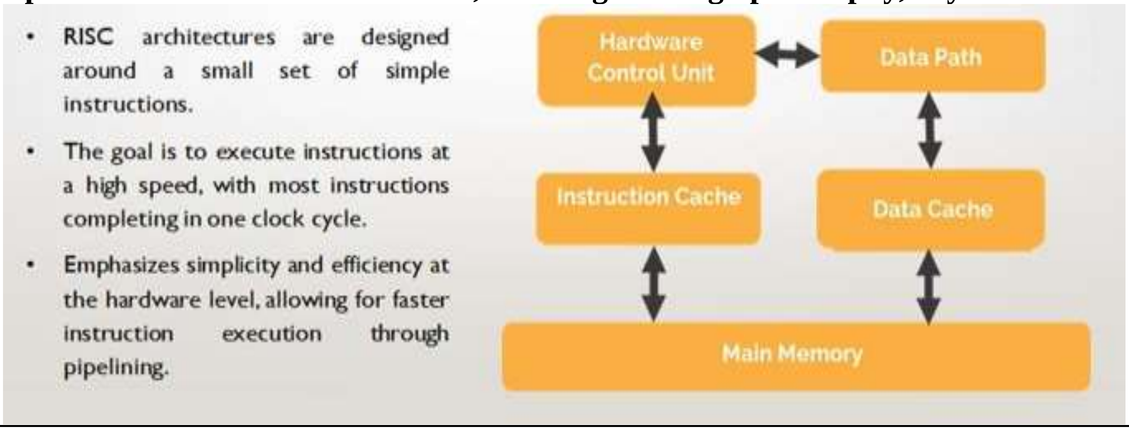
# DDCA
# CO-3 TERMINAL QUESTIONS & ANSWERS

**Consider a scenario where you are developing a simple calculator application for a mobile device. You want to implement basic arithmetic operations. Make use of the related instruction sets and accomplish the task.**



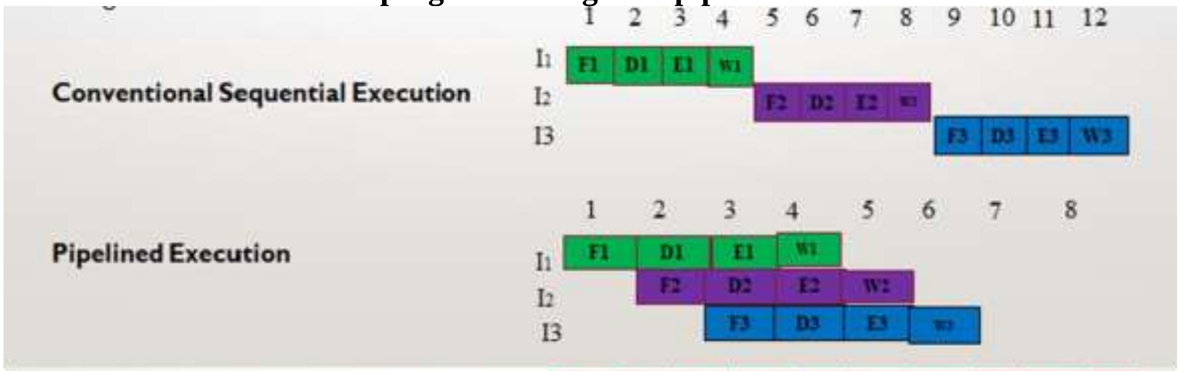
- **Add: Compute sum of two operands**
  Example: add al,07h
  add ax, bx
- **Subtract: Compute difference of two operands**
  Example: sub ah, 05h
  sub ah, al
- **Multiply: Compute product of two operands**
  Example: mov ax,1234h
  mov bx,100h
  mul bx
- **Divide: Compute quotient of two operands**
  Example: mov ax,8003h
  mov cx,100h
  div cx
- **Negate: Change sign of operand**
  Example: neg RT, RA
- **Increment: Add 1 to operand**
  Example: inc A
- **Decrement: Subtract 1 from operand**
  Example: dec A

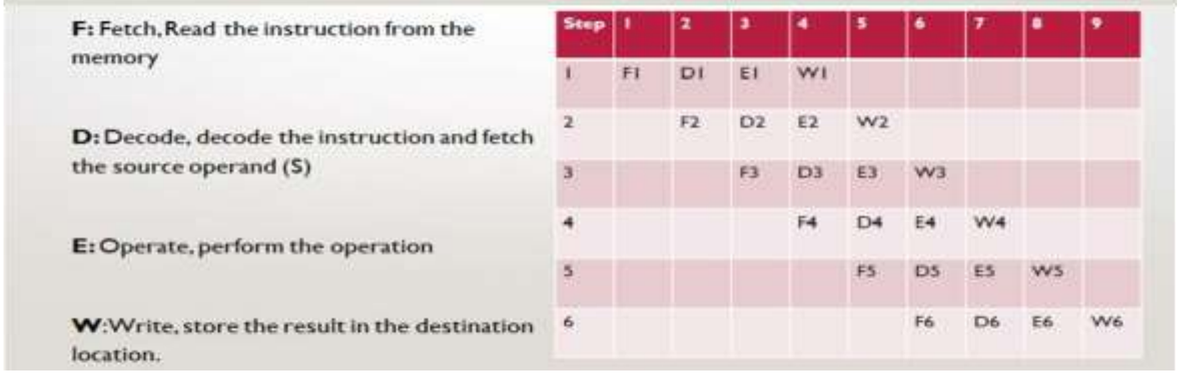
**Develop the model of RISC architecture, detailing its design philosophy, key features.**



- RISC architectures are designed around a small set of simple instructions.
- The goal is to execute instructions at a high speed, with most instructions completing in one clock cycle.
- Emphasizes simplicity and efficiency at the hardware level, allowing for faster instruction execution through pipelining.

**Design the operational flow for a pipelined processor with four stages, outlining each stage's function and how instructions progress through the pipeline.**





**F:** Fetch, Read the instruction from the memory

**D:** Decode, decode the instruction and fetch the source operand (S)

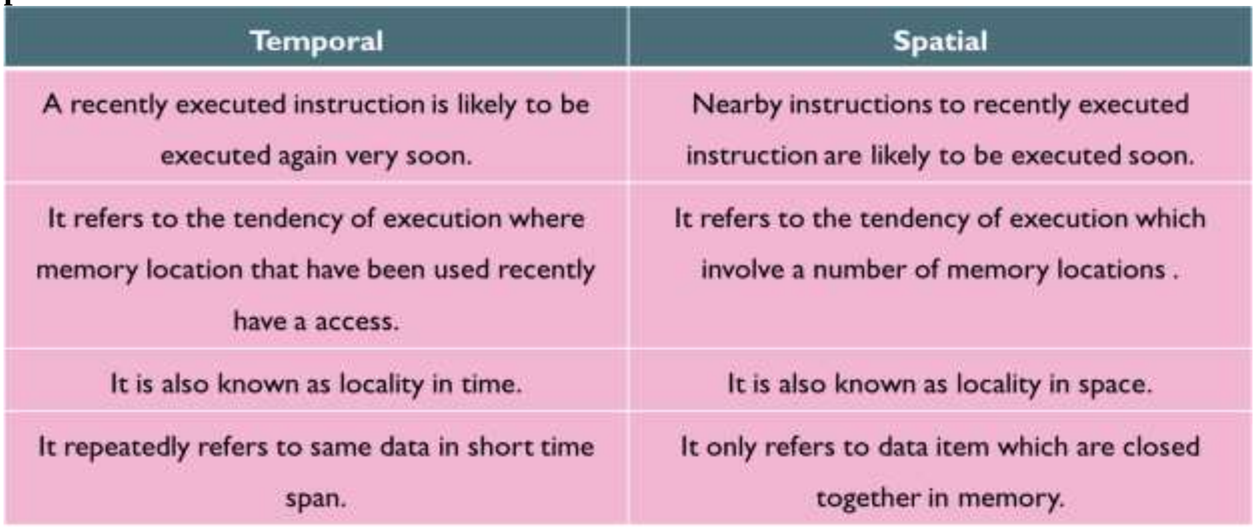
**E:** Operate, perform the operation

**W:** Write, store the result in the destination location.

| Step | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | FI | DI | EI | WI | | | | | |
| 2 | | F2 | D2 | E2 | W2 | | | | |
| 3 | | | F3 | D3 | E3 | W3 | | | |
| 4 | | | | F4 | D4 | E4 | W4 | | |
| 5 | | | | | F5 | D5 | E5 | W5 | |
| 6 | | | | | | F6 | D6 | E6 | W6 |

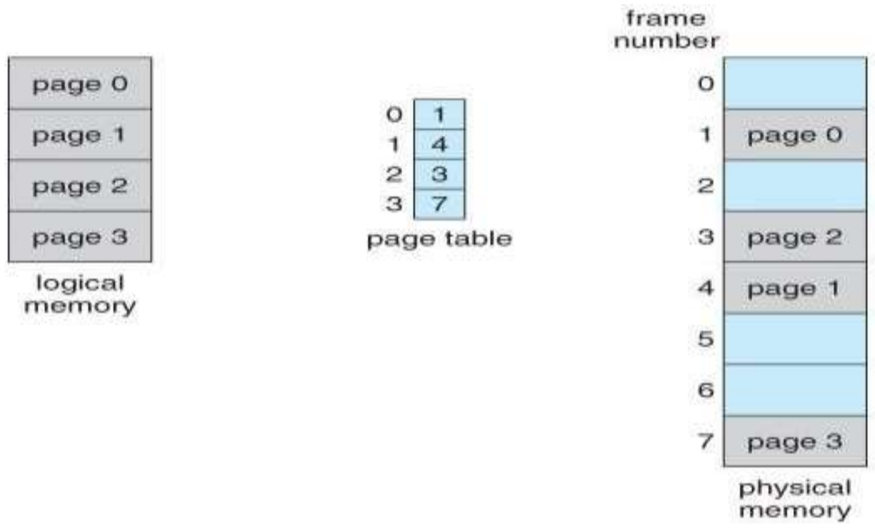
# DDCA
# CO-4 TERMINAL QUESTIONS & ANSWERS

**Differentiate between the concepts of temporal and spatial locality in memory access patterns.**



| Temporal | Spatial |
|---|---|
| A recently executed instruction is likely to be executed again very soon. | Nearby instructions to recently executed instruction are likely to be executed soon. |
| It refers to the tendency of execution where memory location that have been used recently have a access. | It refers to the tendency of execution which involve a number of memory locations . |
| It is also known as locality in time. | It is also known as locality in space. |
| It repeatedly refers to same data in short time span. | It only refers to data item which are closed together in memory. |

**Describe the virtual memory system, focusing on the role of page tables.**



1. Virtual memory provides alternate memory addresses for programs, which are converted to real addresses during execution.
2. Page Table, a data structure in the operating system, maps virtual to physical addresses.
3. It provides frame numbers, indicating where each page is stored in main memory, facilitating memory management.

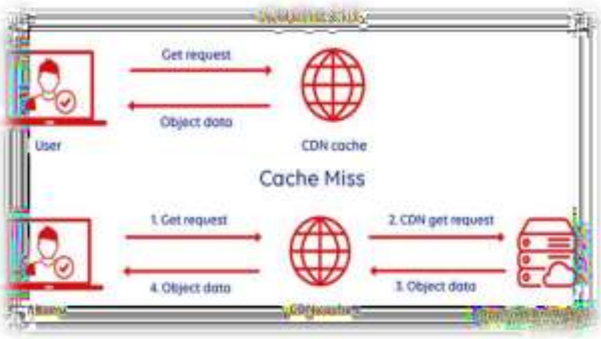
**Describe the importance of temporal and spatial locality with respect to memory.**

1. **Temporal locality:**
   - Recent access predicts future access, a principle vital for caching mechanisms.
   - Caching stores recently accessed data in faster memory layers like CPU caches.
   - Enhances performance by reducing access times for frequently accessed data.

2. **Spatial locality:**
   - Access of one storage location indicates likely access to nearby addresses.
   - Programs typically access data sequentially, benefiting from spatial locality.
   - Fetching adjacent data into faster storage layers optimizes data retrieval by exploiting this principle.

# DDCA
# CO-4 TERMINAL QUESTIONS & ANSWERS

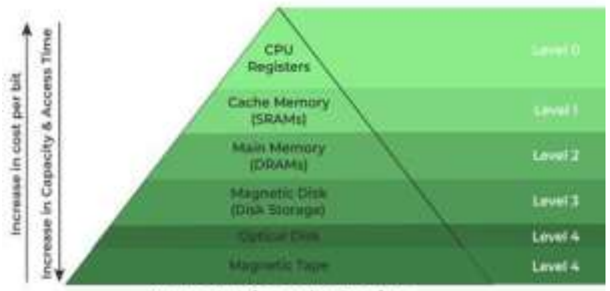
**Investigate the significance and impact of "Hit" and "Miss" events in cache memory, detailing how theseoccurrences influence system performance.**



In cache memory, "Hit" and "Miss" events refer to the outcome of the CPU's attempt to access data:

- In cache memory, a "hit" occurs when the CPU finds the needed data in the cache, enabling quick retrieval without accessing main memory.
- A "miss" happens when the data is not found in the cache, requiring the CPU to fetch it from slower main memory, resulting in slower execution.

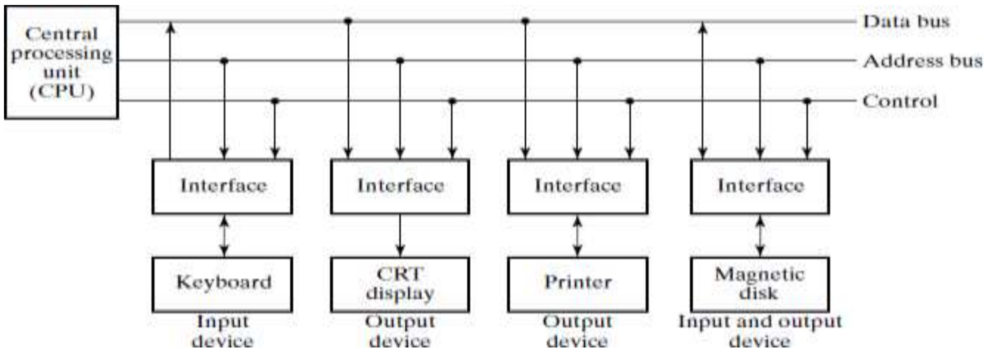
**Investigate the hierarchical organization of memory in computing systems with examples.**



Memory Hierarchy Design

- **Memory hierarchy in computing:** Memory in computing systems is organized hierarchically into multiple levels, with each level offering different characteristics in terms of speed, capacity, and cost.
- **Cache Memory:** Fast, small memory integrated into CPU, stores frequently accessed data for rapid retrieval.
- **Main Memory (RAM):** Larger but slower than cache, holds executing programs and data temporarily.
- **Secondary Storage:** Hard drives, SSDs, etc., offer large storage but slower access than RAM, used for long-term data storage.

**Elaborate role and architecture of IO buses and interface modules with diagram.**

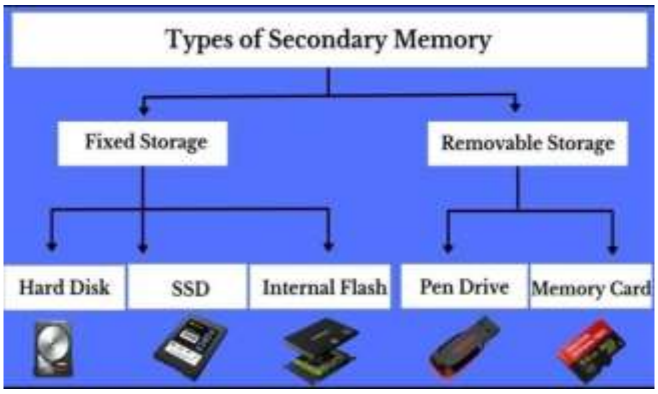


The bus typically consists of three main types of lines:

1. **Data bus:** These carry the actual data being transferred between the CPU and devices.
2. **Address bus:** These specify the location (address) of the data on the device or in memory.
3. **Control bus:** These carry control signals for synchronization and coordination (e.g., read/write, start/stop).

# DDCA
# CO-4 TERMINAL QUESTIONS & ANSWERS

**Evaluate the types and functionality of secondary memory devices exploring their role in data storage.**



**Hard Disk Drives (HDDs):**

- Store data magnetically on rotating platters.
- Commonly used for long-term storage in computers and servers.

**Solid State Drives (SSDs):**

- Store data using flash memory chips.
- Provide faster access times and higher data transfer rates compared to HDDs.

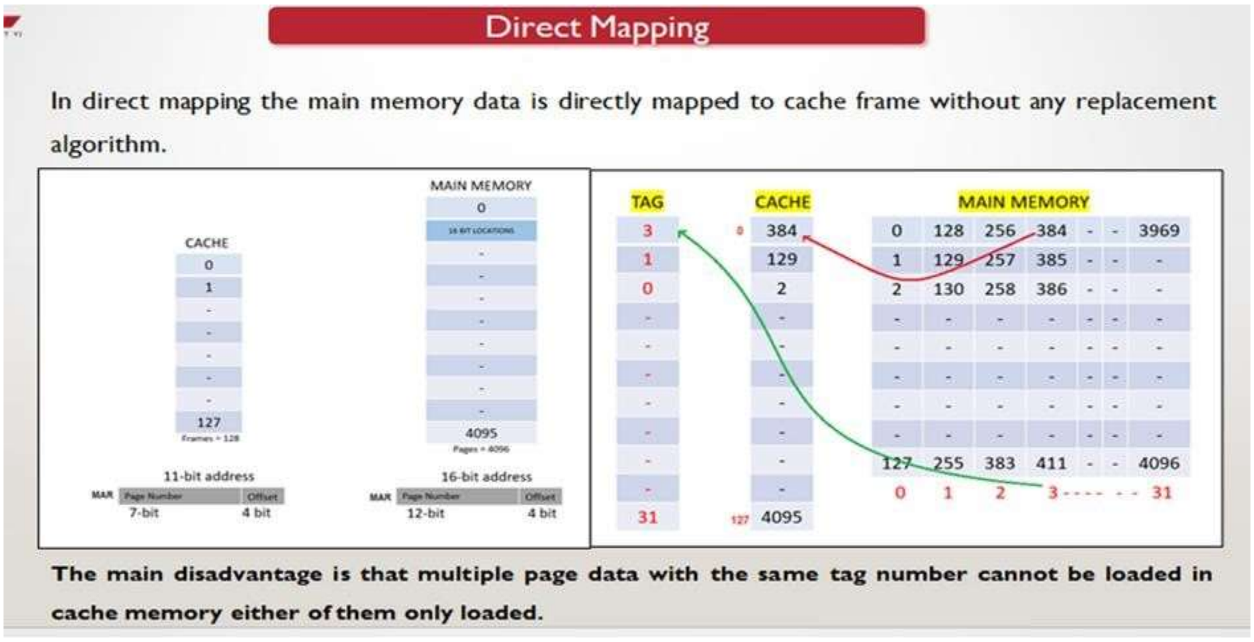
**USB Flash Drives:**

- Portable storage devices that use flash memory.
- Used for transferring files between computers and carrying data on the go.

**External Hard Drives:**

- Similar to internal HDDs but housed in external enclosures.
- Provide additional storage capacity for backups and large data sets.

**Designing a gaming console with a focus on cache memory efficiency, outline and exemplify three mapping procedures, each tailored to specific scenarios to optimize performance in gaming console architecture.**

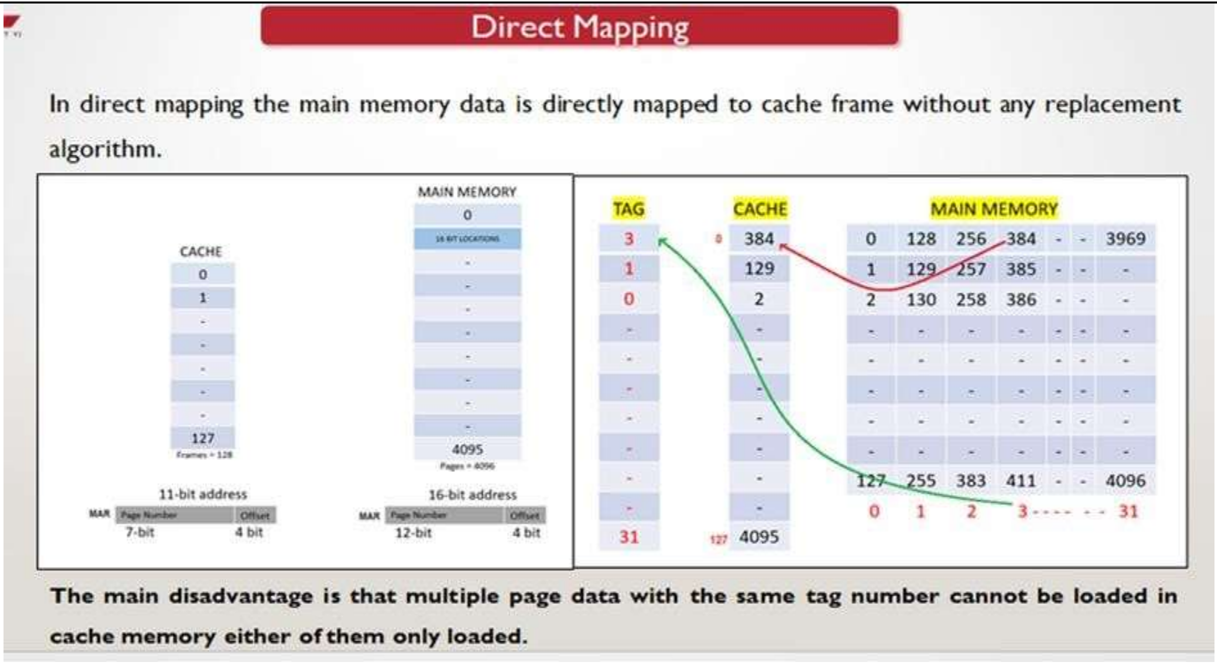


## Direct Mapping

In direct mapping the main memory data is directly mapped to cache frame without any replacement algorithm.

**The main disadvantage is that multiple page data with the same tag number cannot be loaded in cache memory either of them only loaded.**

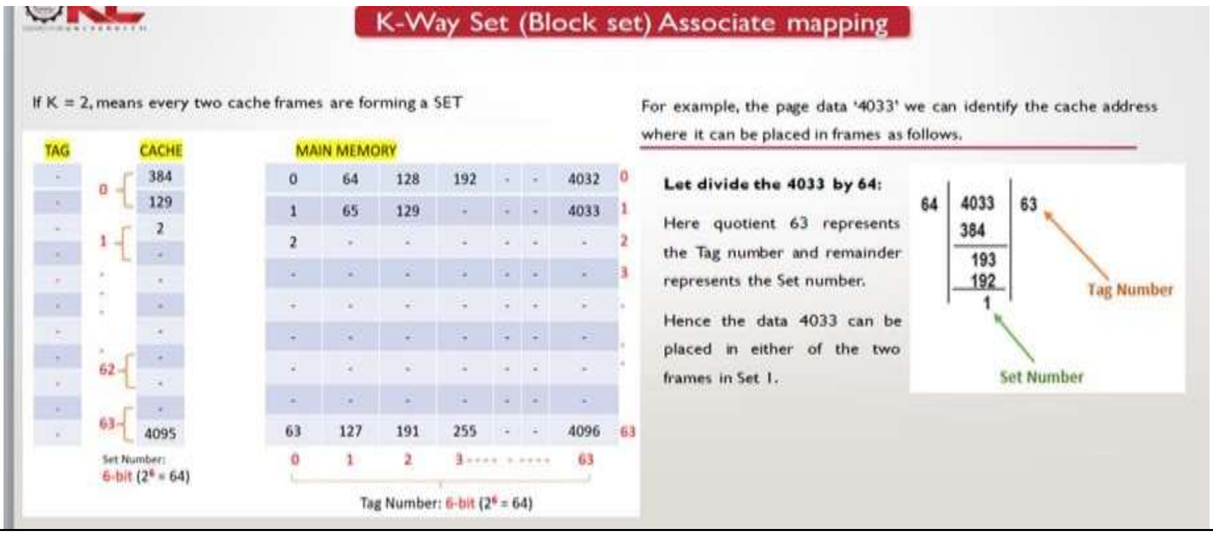
# DDCA
## CO-4 TERMINAL QUESTIONS & ANSWERS



### Direct Mapping

In direct mapping the main memory data is directly mapped to cache frame without any replacement algorithm.

**The main disadvantage is that multiple page data with the same tag number cannot be loaded in cache memory either of them only loaded.**



### K-Way Set (Block set) Associate mapping

If K = 2, means every two cache frames are forming a SET

For example, the page data '4033' we can identify the cache address where it can be placed in frames as follows.

**Let divide the 4033 by 64:**

Here quotient 63 represents the Tag number and remainder represents the Set number.

Hence the data 4033 can be placed in either of the two frames in Set 1.

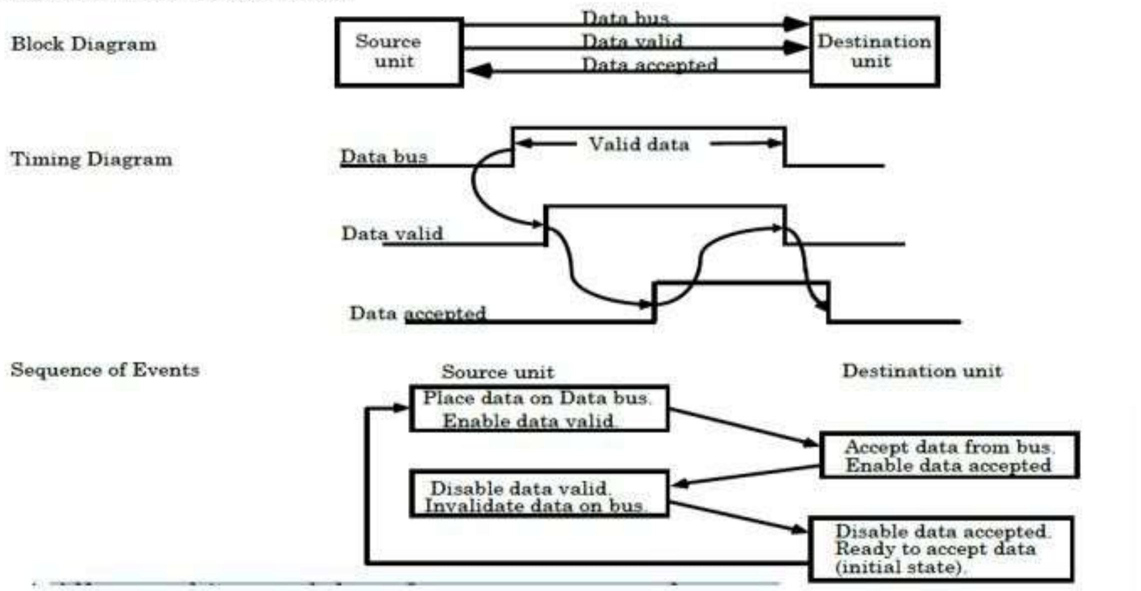
**Analyze handshaking with its types in the context of IO communication.**

**Handshaking Signals:**

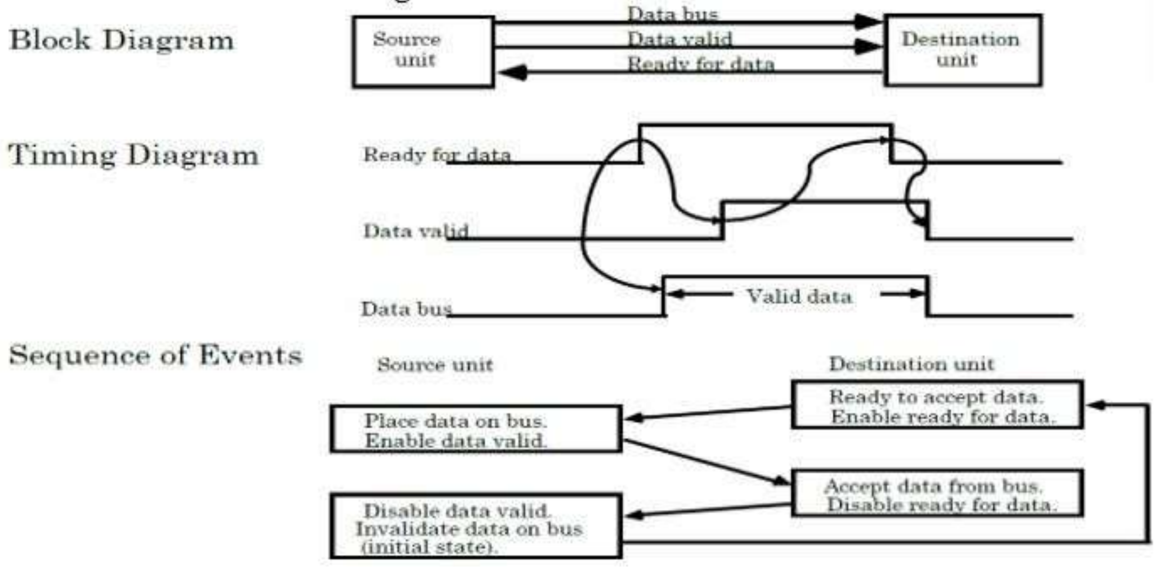
- These are dedicated control lines that carry information about the data transfer status.
- Common handshaking signals include:
  - **Request:** Sent by the receiver to indicate it's ready to receive data.
  - **Acknowledge:** Sent by the receiver after successfully receiving data.
  - **Grant:** Sent by the sender to inform the receiver that data is being sent.
  - **Busy:** Sent by the receiver to indicate it's not ready to receive data.

# DDCA
## CO-4 TERMINAL QUESTIONS & ANSWERS

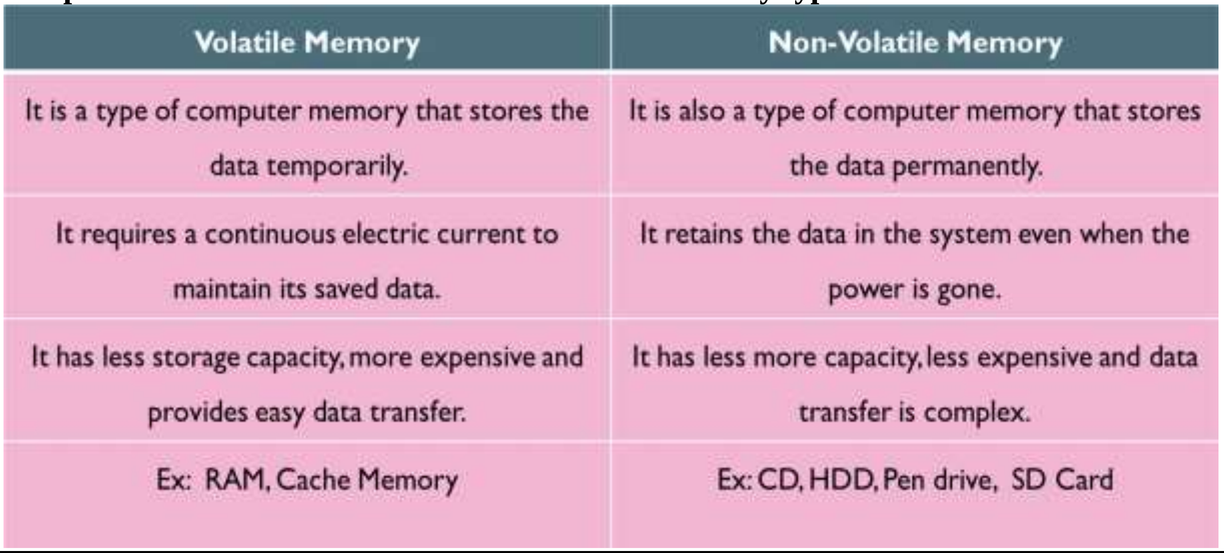
### Source Initiated Handshaking:


Block Diagram
Source unit
Data bus
Data valid
Data accepted
Destination unit
Timing Diagram
Data bus
Valid data
Data valid
Data accepted
Sequence of Events
Source unit
Destination unit
Place data on Data bus. Enable data valid.
Accept data from bus. Enable data accepted
Disable data valid. Invalidate data on bus.
Disable data accepted. Ready to accept data (initial state).

### Destination Initiated handshaking:


Block Diagram
Source unit
Data bus
Data valid
Ready for data
Destination unit
Timing Diagram
Ready for data
Data valid
Data bus
Valid data
Sequence of Events
Source unit
Destination unit
Ready to accept data. Enable ready for data.
Place data on bus. Enable data valid.
Accept data from bus. Disable ready for data.
Disable data valid. Invalidate data on bus (initial state).

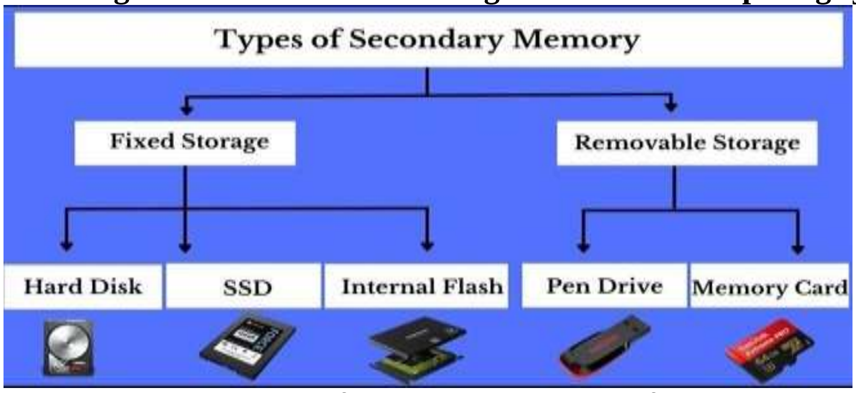
**Compare and contrast Volatile and Non-volatile memory types.**



| Volatile Memory | Non-Volatile Memory |
|---|---|
| It is a type of computer memory that stores the data temporarily. | It is also a type of computer memory that stores the data permanently. |
| It requires a continuous electric current to maintain its saved data. | It retains the data in the system even when the power is gone. |
| It has less storage capacity, more expensive and provides easy data transfer. | It has less more capacity, less expensive and data transfer is complex. |
| Ex: RAM, Cache Memory | Ex: CD, HDD, Pen drive, SD Card |

# DDCA
# CO-4 TERMINAL QUESTIONS & ANSWERS

**Analyze the role and significance of external storage solutions in computing systems.**

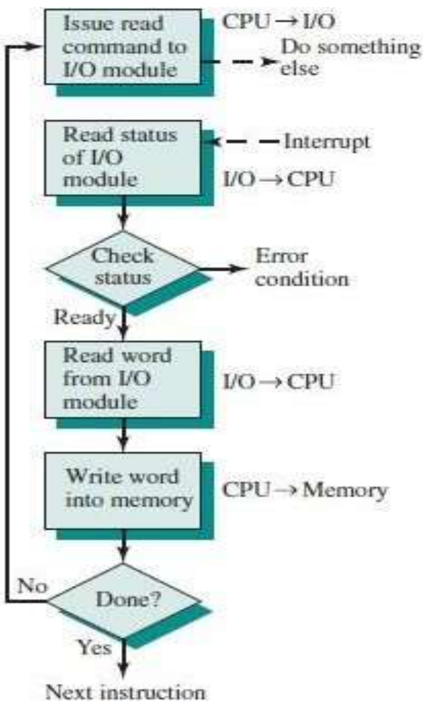


External storage devices come in various forms, each with a specific role:

- **HDDs:** High-capacity workhorses for bulk data (movies, archives) at a lower cost.
- **SSDs:** Blazing-fast for frequently accessed data (applications, games) but pricier per gigabyte.
- **USB Drives:** Compact and portable for data transfer and booting certain systems (limited capacity).
- **Portable SSDs:** Speedy and portable for data transfer on the go (more expensive than USB drives).
- **NAS:** Centralized network storage for sharing files across devices or backing up small businesses.
- **Cloud Storage:** Virtual storage accessed from anywhere for off-site backups and file synchronization (requires internet).

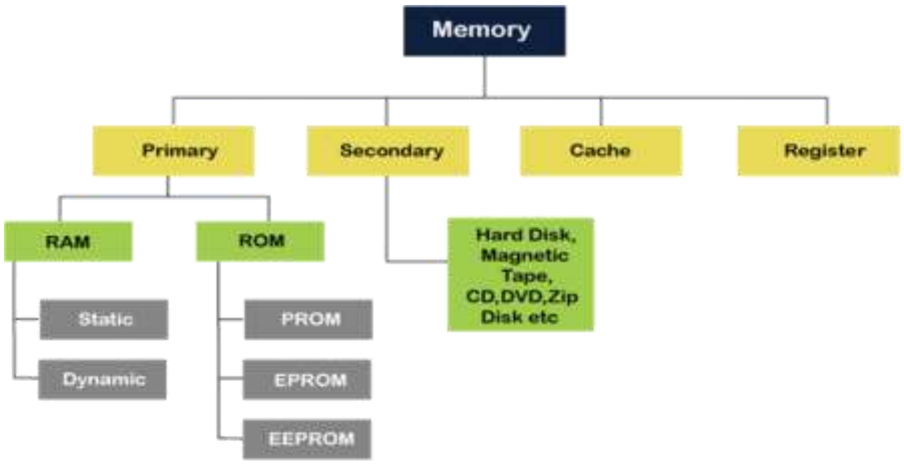
**Utilizing interrupt driven I/O, build a flow diagram for the purpose of sensing data from external devices in a data acquisition system.**

1. System Setup: Configure I/O and enable interrupts for the sensor device.
2. Main Loop: Perform non-critical tasks (wait for interrupts).
3. Interrupt (Sensor Data Ready): Read sensor data, store, set data ready flag.
4. Main Loop (continued): Check data ready flag. If data ready: process data (calculations, display).
5. Loop: Repeat steps 2-4.



# DDCA
# CO-4 TERMINAL QUESTIONS & ANSWERS

**Identify the types and functionality of primary memory devices exploring their role in data storage.**



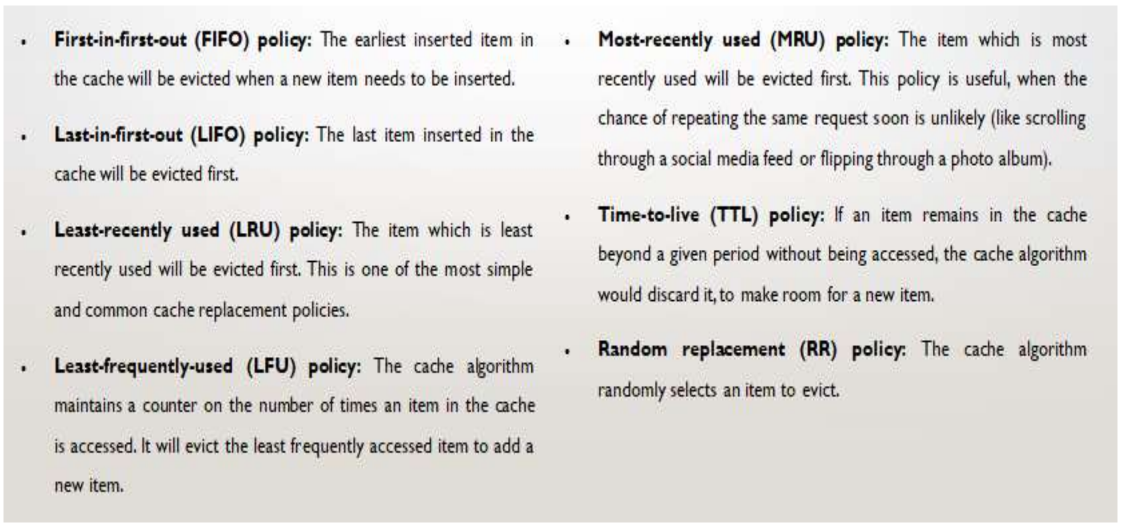
Primary memory, directly accessible by the CPU, offers fast speeds but limited capacity. Here's a breakdown:

1. **RAM (Random Access Memory):** Volatile (loses data on power off). Stores:
   - Running programs
   - Operating system
   - Temporary data

   **Benefits:** Fast access, random access
2. **ROM (Read-Only Memory):** Non-volatile (data persists). It Stores essential startup programs.

   **ROM Classification:**
   - **Masked ROM:** Data permanently programmed during manufacturing.
   - **PROM (Programmable ROM):** One-time programming with a special device.
   - **EPROM (Erasable Programmable ROM):** Erasable with ultraviolet light, then reprogrammable.
   - **EEPROM (Electrically Erasable Programmable ROM):** Electrically erasable and reprogrammable.

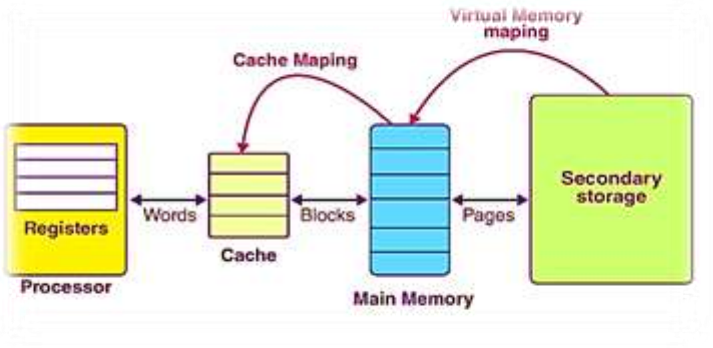
**Identify various cache replacement policies that are useful to manage the cache memory.**



- **First-in-first-out (FIFO) policy:** The earliest inserted item in the cache will be evicted when a new item needs to be inserted.
- **Last-in-first-out (LIFO) policy:** The last item inserted in the cache will be evicted first.
- **Least-recently used (LRU) policy:** The item which is least recently used will be evicted first. This is one of the most simple and common cache replacement policies.
- **Least-frequently-used (LFU) policy:** The cache algorithm maintains a counter on the number of times an item in the cache is accessed. It will evict the least frequently accessed item to add a new item.
- **Most-recently used (MRU) policy:** The item which is most recently used will be evicted first. This policy is useful, when the chance of repeating the same request soon is unlikely (like scrolling through a social media feed or flipping through a photo album).
- **Time-to-live (TTL) policy:** If an item remains in the cache beyond a given period without being accessed, the cache algorithm would discard it, to make room for a new item.
- **Random replacement (RR) policy:** The cache algorithm randomly selects an item to evict.

# DDCA
# CO-4 TERMINAL QUESTIONS & ANSWERS

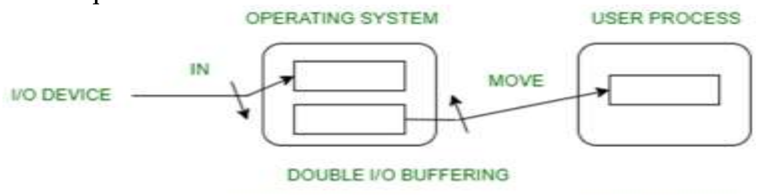
**Interpret the operation of cache memory contribute to improving the performance of a processor, and what advantages does it offer in terms of speed and efficiency?**



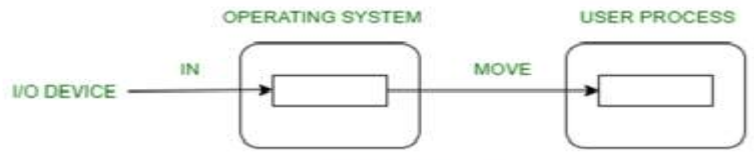
- Cache memory is a speedier, smaller section of memory with an access time that is comparable to registers.
- Cache memory has a shorter access time than primary memory in a memory hierarchy. Since cache memory is typically relatively little, it serves as a buffer.
- Cache provides faster access.
- It acts as buffer between CPU and main memory (RAM).
- Cache primary role is to reduce the average time taken to access data, thereby improving overall system performance.

**Analyze buffering with its types in the context of IO operations.**

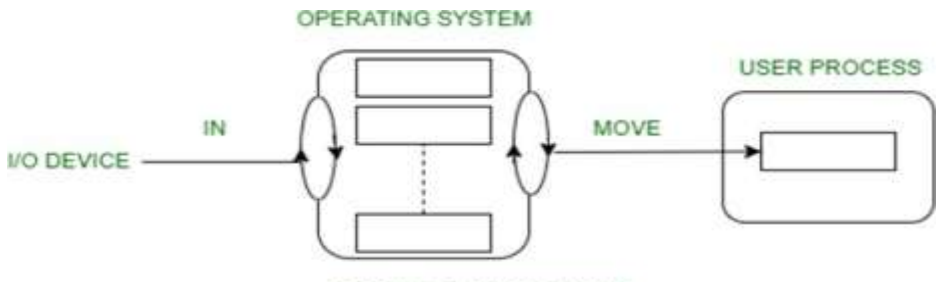
- Buffering creates a synchronization between two devices having different processing speed. For example, if a hard disc (supplier of data) has high speed and a printer (accepter of data) has low speed, then buffering is required.
- Buffering is also required in cases where two devices have different data block sizes.



DOUBLE I/O BUFFERING



SINGLE I/O BUFFERING

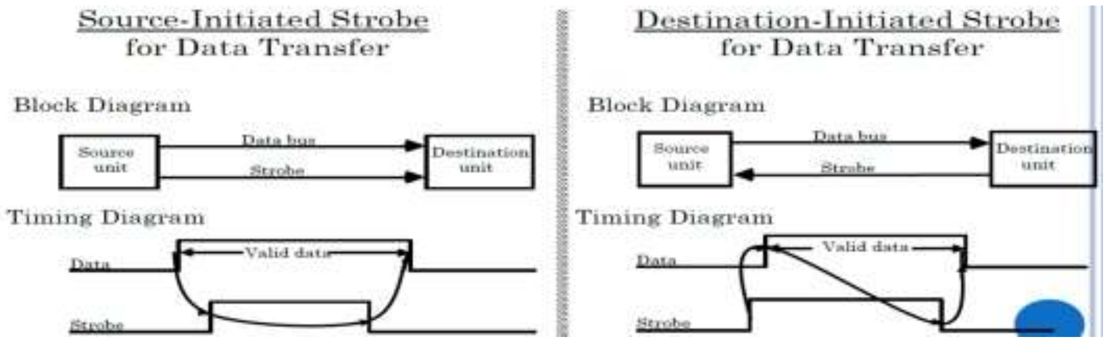


CIRCULAR I/O BUFFERING

- **Single Buffering:** Simple, but CPU stalls if receiver isn't ready.
- **Double Buffering:** Smoother data flow but requires more memory and can overflow.
- **Circular Buffer:** Efficient memory usage, avoids overflow, but adds complexity.

# DDCA
# CO-4 TERMINAL QUESTIONS & ANSWERS

**Illustrate the concepts of source-initiated and destination-initiated strobe pulses in data transmission.**



Source-initiated strobe pulses are signals sent by the transmitting device to indicate the beginning or end of data transmission, helping synchronize communication between devices.

Destination-initiated strobe pulses are signals generated by the receiving device to acknowledge the receipt of data or to request more data, facilitating reliable and efficient data transmission in digital communication systems.