CARDAMOM PLANTERS' ASSOCIATION COLLEGE
(Re-Accredited With 'A' Grade By NAAC)
Pankajam Nagar, Bodinayakanur - 625 582.

DEPARTMENT OF CS & IT

Unit - 5

Counters: Asynchronous and Synchronous Counters - Ripple, Mod, Up-Down Counters– Ring Counters. Memory: Basic Terms and Ideas –Types of ROMs –Types of RAMs.

5.1. Introduction

A counter is a sequential digital circuit that counts the number of clock pulses applied to it.

- Counters are built using flip-flops
- Output changes according to the clock signal
- Count is stored in binary form

Applications of Counters

Counters are mainly used in:

- Digital clocks
- Timers
- Frequency counters
- Digital electronics systems
- Traffic light controllers
- Digital communication systems

5.1.1. Classification of Counters

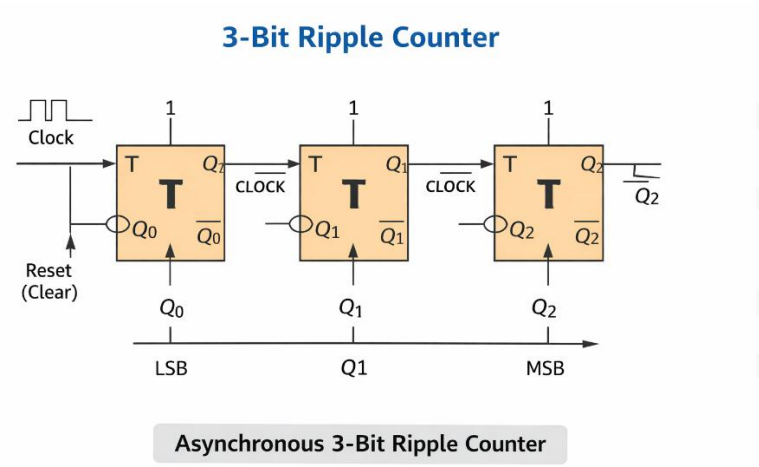
Type	Definition	Working Principle	Applications
Asynchronous Counter (Ripple Counter)	Clock is applied only to the first flip-flop; output ripples through the remaining flip-flops	Flip-flops toggle one after another, causing propagation delay	Frequency division, simple event counting, digital clocks
Synchronous Counter	All flip-flops receive the same clock signal simultaneously	All flip-flops toggle at the same time, making operation faster and reliable	High-speed processors, control systems, digital counters
MOD Counter	Counter that counts a fixed number of states and then resets to zero	Number of states depends on the modulus value	Digital clocks, decade counters, frequency division
Up-Down Counter	Counter capable of counting in both upward and downward directions	Counting direction controlled using UP/DOWN input signal	Elevators, position counters, digital timers
Ring Counter	Shift register counter with feedback connection	A single '1' circulates continuously through the flip-flops	Sequence generation, digital pattern generators

5.1.1.1. Asynchronous Counter (Ripple Counter)

An asynchronous counter is a counter in which the clock signal is applied only to the first flip-flop, and the output of each flip-flop is used as the clock input for the next flip-flop. Because the clock signal ripples through the flip-flops, it is also called a **Ripple Counter**.

Working Principle

- The **first flip-flop** receives the external clock pulse.
- Its **output acts as the clock input** for the next flip-flop.
- Each flip-flop toggles when the previous one changes state.
- This sequential triggering introduces a **propagation delay**.



Asynchronous 3-Bit Ripple Counter

Setup

- **Reset**: All flip-flops (FFs) are cleared to 0 using the direct reset input.
- **T Flip-Flop Inputs**: Each T FF has its input permanently tied to logic 1.
- **Clock Input**: The external clock is applied only to the first flip-flop (Q0).

Working Principle

1. **First FF (Q0)** → toggles on every **falling edge (1→0 transition)** of the system clock.
2. **Second FF (Q1)** → uses Q0's output as its clock. It toggles whenever Q0 goes from 1 → 0.
3. **Third FF (Q2)** → uses Q1's output as its clock. It toggles whenever Q1 goes from 1 → 0.

Thus, the clock "ripples" through Q0 → Q1 → Q2 sequentially.

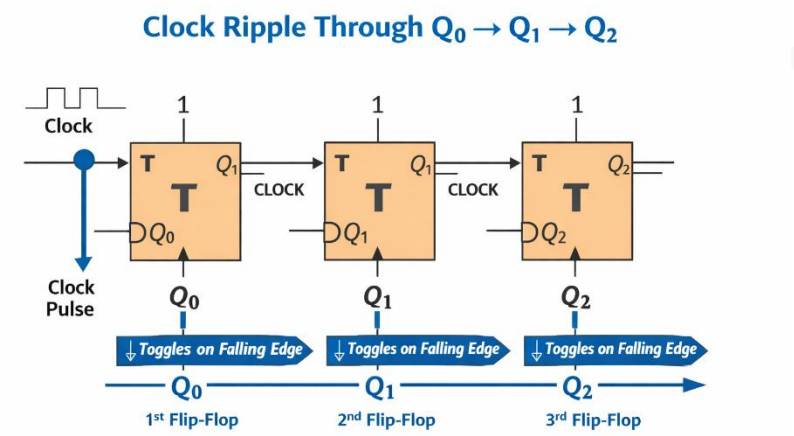
Explanation of the Diagram

- **Clock Input** → applied only to the first flip-flop (Q0).
- **Q0 Output** → acts as the clock for the second flip-flop (Q1).

- **Q1 Output** → acts as the clock for the third flip-flop (Q2).
- **Reset Line** → clears all flip-flops to 0 before counting begins.
- **Permanent T=1** Inputs → ensure each flip-flop toggles whenever its clock input receives a falling edge.

Truth Table: 3-bit Ripple Counter (MOD-8)

Clock Pulse	Q2 (MSB)	Q1	Q0 (LSB)	Decimal
0	0	0	0	0
1	0	0	1	1
2	0	1	0	2
3	0	1	1	3
4	1	0	0	4
5	1	0	1	5
6	1	1	0	6
7	1	1	1	7
8	0	0	0	Reset



Key Takeaways

- **Q0 (LSB)** toggles with every clock pulse.
- **Q1** toggles when Q0 transitions from 1→0.
- **Q2 (MSB)** toggles when Q1 transitions from 1→0.
- Together, they count from 000 → 111 (0 to 7) before resetting.
- The ripple effect introduces a propagation delay, visible as staggered transitions in timing diagrams.

Characteristics

- Simple design → fewer logic gates.
- Propagation delay → outputs don't change simultaneously.

- Low-speed operation → unsuitable for high-frequency systems.

Applications

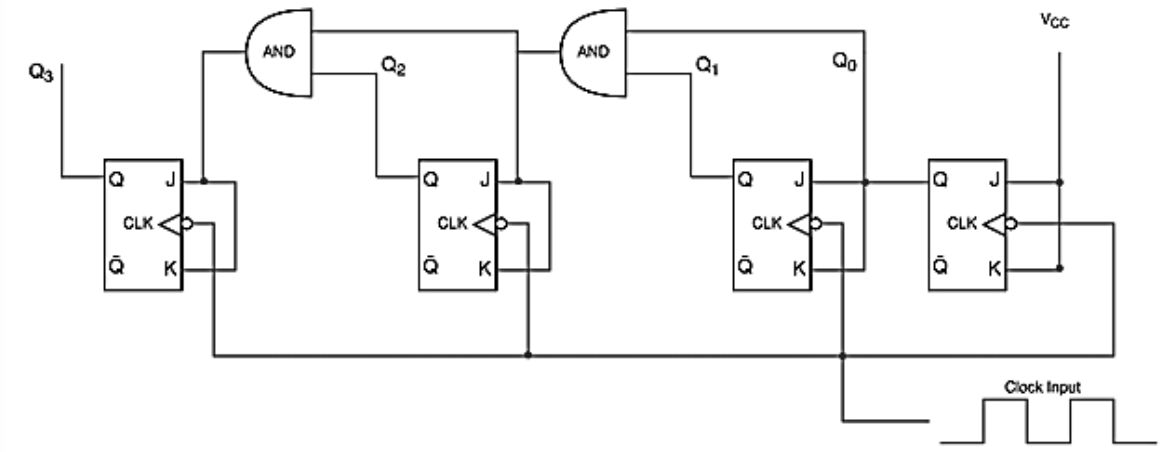
- Digital clocks → dividing crystal oscillator frequency.
- Frequency dividers → communication systems.
- Event counters → automation (e.g., conveyor belt item counting)

5.1.1.2. Synchronous counters

A synchronous counter is a counter in which all flip-flops receive the clock signal at the same time. Unlike asynchronous (ripple) counters, there's no ripple delay because every flip-flop toggles in sync with the same clock pulse.

Working Principle

- **Common clock line:** All flip-flops are triggered by the same clock edge.
- **Control logic:** Additional Logic gates determine when each flip-flop should toggle.
- **Simultaneous state change:** Outputs update together, avoiding propagation delay.



Example: 3-bit Synchronous Counter (MOD-8)

Clock Pulse	Q2	Q1	Q0	Decimal
0	0	0	0	0
1	0	0	1	1
2	0	1	0	2
3	0	1	1	3
4	1	0	0	4
5	1	0	1	5
6	1	1	0	6
7	1	1	1	7
8	0	0	0	Reset

Characteristics

- **Fast operation** → no ripple delay.
- **Reliable** → suitable for high-speed digital systems.
- **Complex design** → requires extra logic gates compared to asynchronous counters.

Applications

- **Digital processors** → instruction cycles, program counters.
- **Control systems** → traffic light sequencing, automation.
- **High-speed counters** → frequency measurement, timers.

5.1.1.3. MOD counters

A **MOD counter** (modulus counter) is a sequential circuit that **counts a fixed number of states** and then **resets to zero**. The term "MOD" refers to the **modulus**, i.e., the total number of unique states the counter cycles through before restarting.

$$\text{MOD} = 2^n$$

where n= number of flip-flops.

Working Principle

- Each flip-flop divides the input clock frequency by 2.
- The counter advances one count per clock pulse.
- When the count reaches the modulus value, the counter resets to 0.
- For non-binary MOD values (like MOD-10), **reset logic** is added to truncate the count sequence.

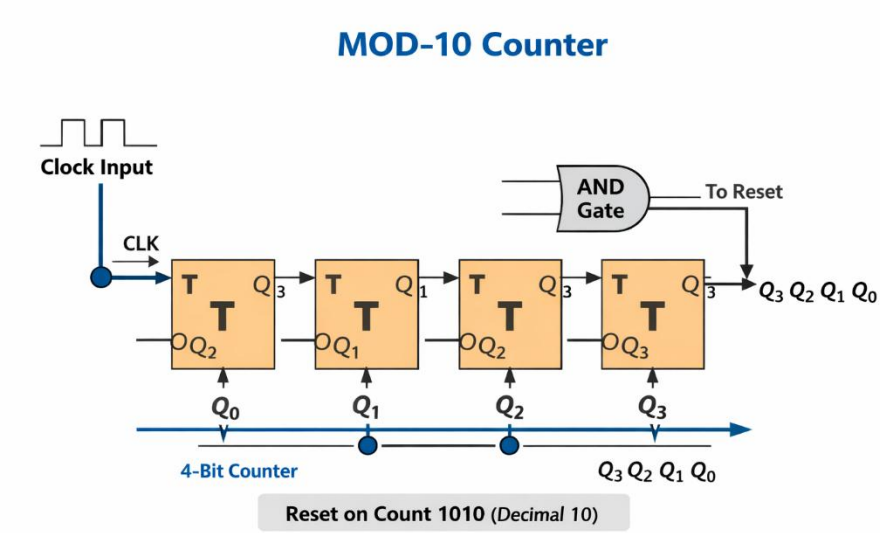
Examples

Counter Type	MOD Value	States	Applications
MOD-2	2	0–1	Frequency division by 2
MOD-4	4	0–3	Divide clock by 4
MOD-8	8	0–7	Binary counting
MOD-10	10	0–9	Digital clocks, decade counters

Example: MOD-10 Counter

Counts: 0 → 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → Reset to 0

To achieve MOD-10, a **MOD-16 counter** can be truncated using reset logic that clears the counter when it reaches 10 (binary 1010).



Step 1. Initialize Flip-Flops (Setup)

Start with a 4-bit binary counter using T flip-flops.

- Use **4 T flip-flops** (Q0–Q3)
- Tie all T inputs to logic '1'
- Apply clock input to the first flip-flop (Q0)
- Clear all flip-flops to '0000' using reset

Step 2. Count Binary States

Allow the counter to cycle through binary values.

- Q0 toggles with each clock pulse
- Q1 toggles when Q0 transitions 1→0
- Q2 toggles when Q1 transitions 1→0
- Q3 toggles when Q2 transitions 1→0

Step 3. Add Reset Logic

Introduce logic to truncate the sequence at decimal 10.

- Detect binary **1010 (decimal 10)** using outputs Q1 and Q3
- Connect Q1 and Q3 to an **AND gate**
- Feed AND gate output to reset line
- Reset counter back to '0000' when 1010 is reached

Step 4. Verify MOD-10 Operation

Confirm the counter cycles through 0–9 only.

- Sequence: 0000 → 0001 → … → 1001
- At 1010, reset triggers → back to 0000
- Total states = 10 (MOD-10)
- Output frequency = Input clock ÷ 10

Key Points

- **Flip-flops required**: 4 (since $2^4 = 16$, truncated to 10).
- **Reset condition**: Binary 1010 (decimal 10).
- **Applications**:
    - **Digital clocks** → decade counting for seconds/minutes.
    - **Frequency division** → divide by 10.
    - **Timers** → embedded systems.

5.1.1.4. Up-Down Counter

An **Up-Down Counter** is a counter that can **increment (count up)** or **decrement (count down)** based on the logic level of a control input.

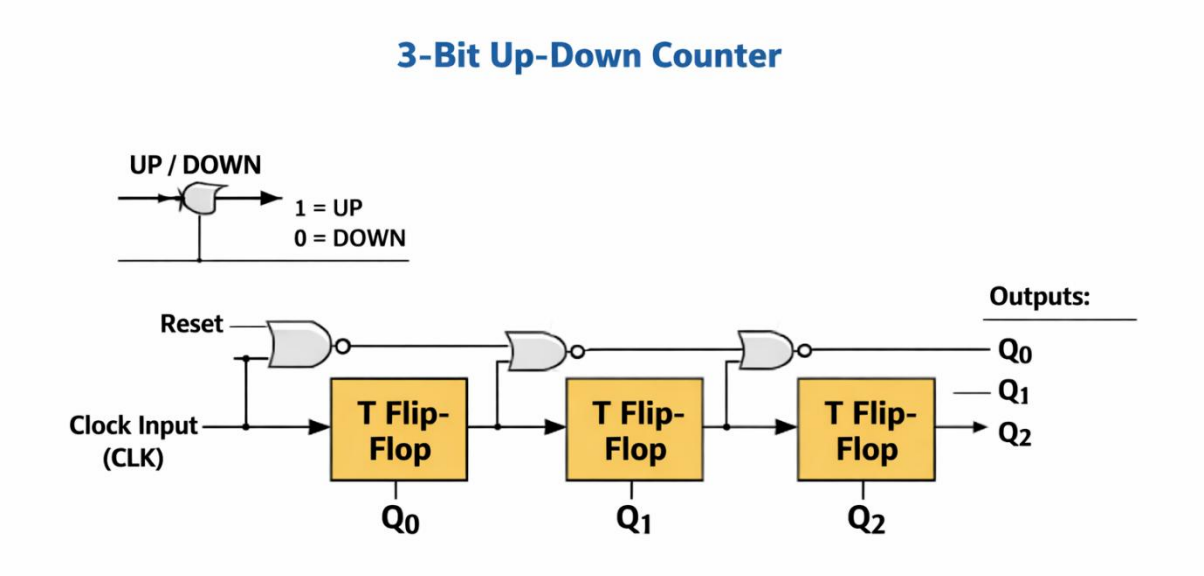
- When the **UP/DOWN control = 1**, the counter counts upward.
- When the **UP/DOWN control = 0**, it counts downward.

Working Principle

- Built using **T or JK flip-flops**.
- Each flip-flop toggles based on the previous stage's output and the direction control.
- The **UP/DOWN control** determines whether the next flip-flop receives the normal or inverted output of the previous stage.

Example: 3-Bit Up-Down Counter

A **3-bit Up-Down Counter** is a sequential circuit that can **count upward (0 → 7)** or **downward (7 → 0)** depending on the logic level of a **control input** called *UP/DOWN*. It uses **three T flip-flops**, each toggling based on the previous stage's output and the direction control signal.



Key Features

- **Inputs**:
    - **Clock (CLK)** → drives Q0.
    - **UP/DOWN control** → decides counting direction (1 = Up, 0 = Down).
    - **Reset** → clears all flip-flops to 000.

- **Flip-flops**:
  - **Q0 (LSB)** toggles every clock pulse.
  - **Q1 (middle stage)** toggles based on Q0 and UP/DOWN control.
  - **Q2 (MSB)** toggles based on Q1 and UP/DOWN control.
- **Outputs**: Q0, Q1, Q2 → together form the 3-bit count.

Clock Pulse	UP Mode (Q2 Q1 Q0)	DOWN Mode (Q2 Q1 Q0)
0	000	111
1	001	110
2	010	101
3	011	100
4	100	011
5	101	010
6	110	001
7	111	000

Working Principle

1. **Initial Reset**
   - All flip-flops are cleared to 0 using the reset input.
   - The counter starts from 000 in up mode or 111 in down mode.
2. **UP/DOWN Control**
   - When **UP/DOWN = 1**, the counter counts **upward**.
   - When **UP/DOWN = 0**, the counter counts **downward**.
   - This control signal determines whether each flip-flop receives the **normal** or **inverted** output of the previous stage (using XOR gates).
3. **Clock Input**
   - The clock is applied to the first flip-flop (Q0).
   - Each subsequent flip-flop toggles when the previous one transitions from 1 → 0 (for up-counting) or 0 → 1 (for down-counting).
4. **Counting Sequence**
   - **Up Mode (UP/DOWN = 1)**: 000 → 001 → 010 → 011 → 100 → 101 → 110 → 111 → 000
   - **Down Mode (UP/DOWN = 0)**: 111 → 110 → 101 → 100 → 011 → 010 → 001 → 000 → 111

Characteristics

- **Bidirectional counting** → counts both up and down.

- **MOD value** → $2^3 = 8$.
- **Control logic** → XOR gates switch clock polarity for direction change.
- **Sequential operation** → each flip-flop toggles based on the previous one's output.

Applications

- **Digital timers** → for countdown and elapsed time.
- **Position counters** → robotics and automation.
- **Frequency measurement** → bidirectional pulse counting.
- **Elevator control systems** → floor tracking.

5.1.1.5. Ring Counters

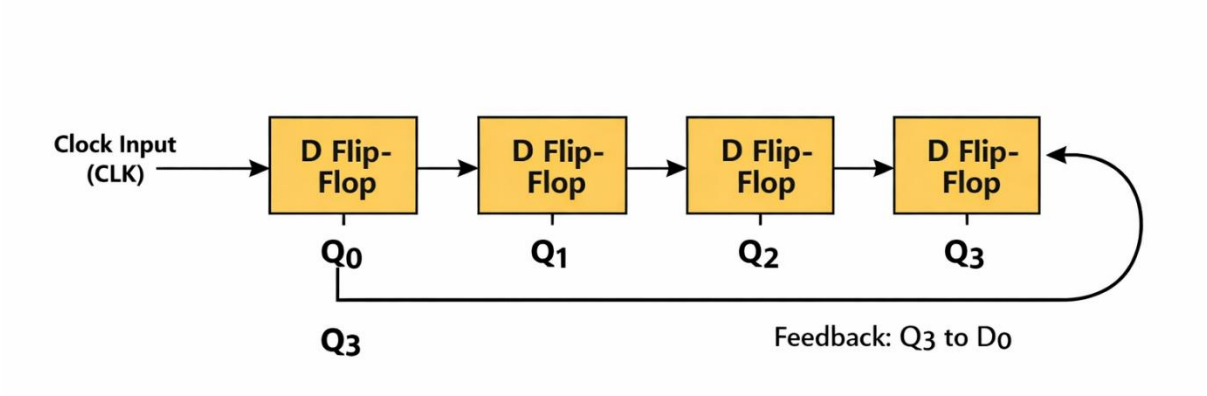
A **Ring Counter** is a type of **shift register counter** where the output of the last flip-flop is fed back to the input of the first flip-flop. It forms a **closed loop**, allowing a single logic '1' (or '0') to circulate through the flip-flops with each clock pulse.

Working Principle

- Built using **D or JK flip-flops** connected in series.
- Initially, only one flip-flop is set to logic '1'; all others are '0'.
- On each clock pulse, the '1' shifts to the next flip-flop.
- After the last flip-flop, the '1' returns to the first — forming a ring.

Example: 4-Bit Ring Counter

A **4-bit Ring Counter** uses **four D flip-flops** connected in a loop. Each flip-flop stores one bit, and the output of the last flip-flop feeds back into the first, forming a ring.



Key Features

- **Inputs**:
  - **Clock (CLK)** → drives all four flip-flops simultaneously.
  - **Reset** → initializes one flip-flop to '1' and the rest to '0'.
- **Flip-flops**:
  - **Q0, Q1, Q2, Q3** arranged in sequence.
  - Each flip-flop passes its output to the next stage.
- **Feedback Loop**:

- Output of **Q3** is fed back to the input of **Q0**, forming the "ring."

Working Principle

1. On each clock pulse, the '1' shifts to the next flip-flop.
2. After four pulses, the '1' returns to Q0.
3. The pattern repeats cyclically: 1000 → 0100 → 0010 → 0001 → 1000 → …

Clock Pulse	Q3	Q2	Q1	Q0
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	1	0	0	0

Characteristics

- **MOD value** = 4 (for four flip-flops).
- **Simple design** → no decoding logic needed.
- **Self-starting** if initialized correctly.
- **Circulating bit pattern** → one active bit moves around the ring.

Applications

- **Sequence generation**
- **LED chasers**
- **Timing circuits**
- **Digital control systems**

5.2. Memory

Memory is the component of a computer system used to **store data and instructions** for processing. It determines how fast and efficiently a system performs tasks.

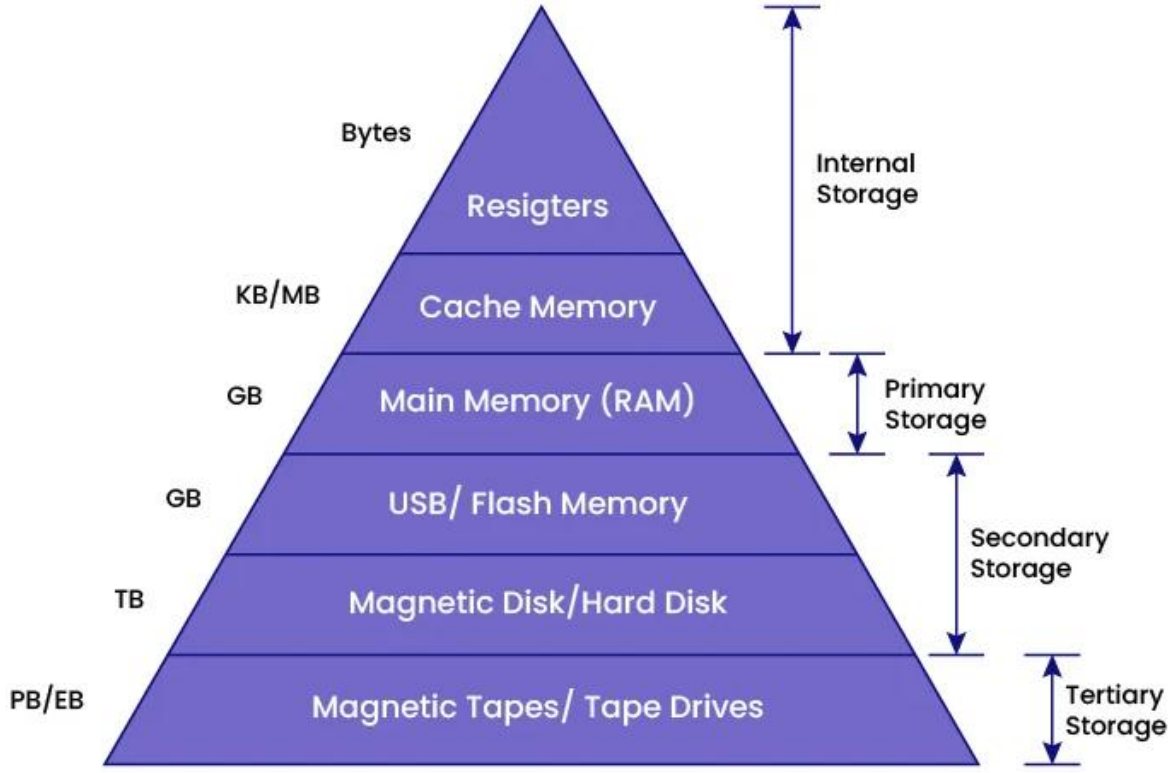
- **Memory Cell:** The smallest storage unit, each with a unique address.
- **Address Lines:** Carry the location of the memory cell.
- **Data Lines:** Carry data to and from the memory.
- **Control Signals:**
  - **R/W (Read/Write):** Determines operation type.
  - **CS (Chip Select):** Enables or disables the memory chip.
- **Bit:** Smallest unit of data (0 or 1).
- **Byte:** Group of 8 bits.
- **Word:** Fixed-size group of bits processed as a unit. (Eg: 8-bit word, 16-bit word,32-bit word)
- **Memory Location:** A memory location is a place where data is stored. Each location has a unique: **Address**

- **Address:** Unique identifier for each memory location.
- **Capacity:** Memory capacity indicates how much data can be stored. (Eg: KB (Kilobyte), MB (Megabyte), GB (Gigabyte))
- **Access Time**: Time taken to read/write data.

5.2.1. Memory Hierarchy

Memory hierarchy arranges devices by **speed, cost, and capacity**:

Level	Example	Speed	Capacity	Cost per Bit
1	CPU Registers	Fastest	Smallest	Highest
2	Cache Memory	Very Fast	Small	High
3	Main Memory (RAM)	Fast	Medium	Moderate
4	Secondary Storage (HDD, SSD)	Slow	Large	Low



Trend: As we move down the hierarchy, **capacity increases**, **cost decreases**, and **access time increases**.

5.2.2.1. Memory Hierarchy Layers

1. **CPU Registers**
    - Registers are small storage locations inside the CPU.
    - Fastest, smallest memory.
    - Stores immediate operands and instructions. (Temporary storage during processing)
2. **Cache Memory**

- High-speed buffer between CPU and RAM.
- Levels: L1, L2, L3.
- Stores frequently accessed data.
- Reduces access time

3. **RAM (Main Memory or Primary Memory)**
   - Main memory stores programs and data currently used by CPU.
   - Volatile, larger than cache.
   - Holds active programs and data.
4. **ROM**
   - Non-volatile, permanent instructions (firmware, BIOS).
   - Ensures system boot and integrity.
5. **Secondary Storage**
   - HDDs, SSDs, optical disks.
   - Largest capacity, slowest access.
   - Used for long-term data storage.
6. **Magnetic Tape / Backup Storage**
   - Used for long-term backup storage.
   - Very large capacity
   - Very slow access
   - Lowest cost

Key Characteristics

Layer	Speed	Capacity	Volatility
CPU Registers	Fastest	Few bytes	Volatile
Cache	Very fast	KB–MB	Volatile
RAM	Fast	GBs	Volatile
ROM	Moderate	MBs–GBs	Non-volatile
Storage	Slowest	TBs	Non-volatile

5.2.3. Types of Memory Devices

5.2.3.1. Volatile Memory

- Requires continuous power to retain data.
- **Example:** RAM.
- **Properties:**
  - High speed.
  - Temporary storage.
  - Used for active data processing.

5.2.3.1.1. Types of RAM (Random Access Memory)

- **Static RAM (SRAM)**
  - Built using flip-flops.
  - Very fast and reliable.
  - Expensive, consumes more power.
  - Used in **cache memory** and registers.
- **Dynamic RAM (DRAM)**
  - Built using capacitors.
  - Needs periodic refreshing to retain data.
  - Slower than SRAM but cheaper.
  - Used in **main system memory**.
- **Synchronous DRAM (SDRAM)**
  - Operates in sync with the CPU clock.
  - Improves performance by reducing wait states.
  - Common in modern PCs.
- **Double Data Rate SDRAM (DDR SDRAM)**
  - Transfers data twice per clock cycle.
  - Variants: DDR, DDR2, DDR3, DDR4, DDR5.
  - Used in desktops, laptops, and servers.
- **Video RAM (VRAM)**
  - Specialized RAM for graphics processing.
  - Stores image data for fast rendering.
  - Used in **GPUs** for gaming and multimedia.

Key Takeaways

- **SRAM** → fastest, costly, used in cache.
- **DRAM** → slower, cheaper, used in main memory.
- **SDRAM/DDR** → synchronized, high-performance system RAM.
- **VRAM** → dedicated for graphics.
- RAM is **volatile**, meaning it loses data when power is off.

5.2.3.2. Non-Volatile Memory

- Retains data even when power is off.
- **Examples:** ROM, Flash, HDD, SSD.
- **Properties:**
  - Permanent storage.
  - Slower than volatile memory.
  - Used for firmware and long-term data.

5.2.3.2.1. Types of ROM (Read-Only Memory)

- **Mask ROM (MROM)**
    - Programmed permanently during manufacturing.
    - Cannot be altered later.
    - Used for fixed data like microcontroller firmware.
- **Programmable ROM (PROM)**
    - Manufactured blank; user can program it once.
    - One-time programmable.
    - Useful for prototypes or small production runs.
- **Erasable Programmable ROM (EPROM)**
    - Can be erased using **UV light** and reprogrammed.
    - Transparent quartz window on the chip for UV exposure.
    - Reusable but requires special equipment.
- **Electrically Erasable Programmable ROM (EEPROM)**
    - Erased and reprogrammed electrically.
    - More flexible than EPROM.
    - Allows selective erasure of bytes.
- **Flash Memory**
    - Advanced EEPROM variant.
    - Fast, rewritable, widely used in USB drives, SSDs, memory cards.
    - Supports block-level erasure and programming.

5.2.4. Difference Between RAM and ROM

Feature	RAM	ROM
Full Form	Random Access Memory	Read Only Memory
Nature	Volatile Memory	Non-Volatile Memory
Data Storage	Temporary	Permanent
Power Supply	Data is lost when power is OFF	Data is retained even when power is OFF
Operation	Read and Write	Mostly Read Only
Speed	Faster	Slower than RAM
Capacity	Usually larger	Usually smaller
Cost	More expensive	Less expensive
Usage	Stores currently running programs and data	Stores firmware and boot programs
Examples	SRAM, DRAM	PROM, EPROM, EEPROM

5.2.5. Difference Between SRAM and DRAM

Feature	SRAM	DRAM

Full Form	Static Random Access Memory	Dynamic Random Access Memory
Storage Method	Stores data using flip-flops	Stores data using capacitors
Refreshing	No refreshing required	Requires periodic refreshing
Speed	Faster	Slower than SRAM
Cost	Expensive	Less expensive
Density	Lower storage density	Higher storage density
Power Consumption	Higher	Lower
Complexity	More complex circuit	Simple circuit
Capacity	Smaller capacity	Larger capacity
Volatility	Volatile	Volatile
Applications	Cache memory	Main memory (RAM)