UNIT – V: Input-Output & Memory Organization

Input-Output Organization: Input-Output Interface, Asynchronous data transfer, Modes of Transfer, Priority Interrupt Direct memory Access.

Memory Organization: Memory Hierarchy, Main Memory, Auxiliary memory, Associate Memory, Cache Memory.

1. Input-Output Organization & Interface

- Input devices provide data to the computer; output devices receive processed results.
- The I/O interface is the hardware/software boundary between the CPU, memory and peripheral devices.
- An interface typically contains data, status and control registers.
- Data register: temporarily holds data being transferred.
- Status register: indicates conditions such as ready, busy, error or completion.
- Control register: receives commands such as start, reset, read or write.
- The interface also performs address decoding, timing and control-signal generation.

2. Asynchronous Data Transfer

- Asynchronous transfer is used when two communicating units do not share a common clock.
- CPU and peripheral devices often operate at different speeds.
- Timing is controlled using control signals rather than a common clock.
- Strobe method: one unit generates a pulse indicating when data is valid.
- Handshaking method: sender and receiver exchange request/acknowledge signals.
- Handshaking is more reliable because the receiver confirms that data has been accepted.
- Applications include communication with slow or independently clocked peripherals.

3. Modes of Data Transfer

- Programmed I/O: CPU directly controls the transfer and repeatedly checks device status.
- Interrupt-driven I/O: the device interrupts the CPU when service is required.
- Direct Memory Access (DMA): a DMA controller transfers data directly between I/O and memory.
- Programmed I/O is simple but can waste CPU cycles.
- Interrupt-driven I/O improves CPU utilization for irregular or moderate-speed transfers.
- DMA is preferred for large blocks and high-speed devices.
- The mode selected depends on transfer size, device speed and processor workload.

4. Programmed I/O

- The CPU executes instructions that control every I/O transfer.
- Typical sequence: issue command → read status → wait if busy → transfer data → repeat.
- Polling means repeatedly testing a status flag until the device becomes ready.
- Advantages: simple design, low hardware complexity and easy implementation.
- Disadvantages: CPU remains occupied while waiting for slow peripherals.
- Suitable for simple systems, low-speed devices and operations where hardware simplicity is important.
- Example: CPU repeatedly checks whether a keyboard controller has received a character.

5. Interrupt-Driven I/O

- Instead of continuously polling, the CPU allows the device to request attention through an interrupt.
- An interrupt request causes the CPU to save the current execution state and branch to an Interrupt Service Routine (ISR).
- The ISR identifies the source, transfers data or handles the event, clears the request and returns control.
- Advantages: CPU can perform useful work while the device is idle.
- Interrupts may be maskable or non-maskable depending on system requirements.
- Nested interrupts may allow a higher-priority interrupt to interrupt a lower-priority service routine.
- Used extensively for keyboards, timers, communication interfaces and other event-driven devices.

6. Priority Interrupt

- Several peripherals may request service at the same time, so the system needs a priority mechanism.
- Priority determines which interrupt is serviced first.
- Software polling checks devices in a fixed order; simple but slower.
- Daisy chaining connects devices in a priority chain; the device closest to the priority controller may receive higher priority.
- Parallel priority hardware uses a priority encoder to select the highest-priority request quickly.
- Fixed priority is simple, while rotating or dynamic priority can improve fairness.
- Priority systems help ensure that time-critical devices receive prompt service.

7. Direct Memory Access (DMA)

- DMA allows an I/O device to transfer data directly to or from main memory without CPU handling every word.
- The CPU initializes the DMA controller with memory address, transfer count and transfer direction.
- The DMA controller requests control of the system bus and performs the transfer.
- After completion, DMA can generate an interrupt to notify the CPU.
- DMA reduces CPU overhead and is efficient for block transfers.
- Common applications: storage devices, network interfaces, audio/video streams and high-speed data acquisition.
- DMA requires coordination of bus ownership between CPU and DMA controller.

8. DMA Transfer Methods

- Burst/block mode: DMA retains the bus and transfers an entire block before releasing it.
- Cycle stealing: DMA transfers one word/byte at a time by temporarily taking individual bus cycles.
- Transparent DMA: transfers occur when the CPU is not using the bus.
- Burst mode provides high transfer speed but can delay CPU memory access.
- Cycle stealing balances CPU and DMA activity but introduces small interruptions.
- DMA registers generally include current address, transfer count and control/status information.
- At completion, the DMA controller releases the bus and may interrupt the CPU.

9. Memory Hierarchy

- Memory hierarchy organizes storage into levels according to speed, capacity and cost per bit.
- Typical order: CPU registers → cache → main memory → secondary/auxiliary storage.
- Upper levels are faster, smaller and more expensive per bit.
- Lower levels are slower, larger and cheaper per bit.
- The hierarchy works because programs exhibit locality of reference.
- Temporal locality: recently used data/instructions are likely to be used again.
- Spatial locality: nearby memory locations are likely to be accessed soon.
- Goal: provide a large apparent memory with performance close to the fastest levels.

10. Main Memory

- Main memory is the primary memory directly accessible by the CPU.
- RAM is volatile read/write memory used for currently executing programs and active data.
- DRAM stores bits using capacitors and is commonly used as system main memory.
- SRAM is faster and does not require refresh, but is more expensive and is mainly used for cache.
- ROM and related non-volatile technologies retain information without power and are commonly used for firmware.
- Memory is divided into addressable locations, each identified by a unique address.
- Important characteristics include capacity, access time, cycle time, bandwidth and cost.

11. Auxiliary Memory

- Auxiliary memory is non-volatile secondary storage used for long-term data and program storage.
- Examples include Hard Disk Drives (HDDs), Solid-State Drives (SSDs), optical media and magnetic tape.
- HDDs store data magnetically on rotating disks; SSDs use flash memory and have no moving parts.
- Auxiliary storage has much greater capacity than main memory but generally higher access latency.
- It retains data when system power is removed.
- Used for operating systems, applications, user files, backups and archival data.
- Storage performance depends on latency, transfer rate, interface and workload.

12. Associative Memory

- Associative Memory is also called Content Addressable Memory (CAM).
- Unlike conventional memory, it searches for stored content rather than using only a memory address.
- An input search key is compared with multiple stored entries in parallel.
- A matching entry can be identified very quickly.
- Typical operations involve argument/search registers, key or mask registers and match logic.
- Advantages: extremely fast parallel search.
- Disadvantages: more complex and expensive hardware than ordinary RAM.
- Applications include Translation Lookaside Buffers (TLBs), cache tag lookup and network routing tables.

13. Cache Memory

- Cache is a small, high-speed memory placed close to the CPU.
- It stores copies of frequently accessed instructions and data from main memory.
- Cache hit: requested information is found in cache, producing fast access.
- Cache miss: information is absent and must be fetched from a lower memory level.
- L1 cache is usually smallest and fastest; L2 is larger; L3 is larger and often shared among CPU cores.
- Cache performance is influenced by hit ratio, miss penalty, block size and replacement policy.
- Common replacement policies include Least Recently Used (LRU), FIFO and random replacement.
- Write policies include write-through and write-back.

14. Cache Mapping Techniques

- Direct mapping: each main-memory block maps to exactly one cache line; simple and fast but can cause conflict misses.
- Associative mapping: any memory block can occupy any cache line; flexible but requires more comparison hardware.
- Set-associative mapping: cache is divided into sets and a block maps to one set but can occupy one of several lines.
- Cache address fields commonly include tag, index/set and block offset.
- Block offset selects a byte/word within a cache block.
- Index or set bits select the cache location; tag bits identify the requested memory block.
- Mapping technique affects cost, speed, complexity and miss rate.

15. Summary, Comparison & Key Takeaways

- I/O interfaces provide registers and control logic for communication between CPU and peripherals.
- Asynchronous transfer uses strobe or handshaking when timing is not shared.
- Programmed I/O is simple but consumes CPU time; interrupts improve CPU utilization.
- Priority interrupt hardware/software decides which simultaneous request is handled first.
- DMA provides efficient block transfers with minimal CPU involvement.
- Memory hierarchy balances speed, capacity and cost through multiple storage levels.
- Main memory supports active execution; auxiliary memory provides long-term storage.
- Associative memory performs content-based searches, while cache reduces average CPU memory-access time.
- Key exam point: understand the working, advantages, disadvantages and applications of each I/O and memory technique.