Computer Organization and Architecture

Study Notes

Chapter 4: Memory and I/O Devices

Topics Covered

4.1 Memory Hierarchy Model and Comparison on Cost, Speed and Size

4.2 Cache Memory, Mapping Techniques, Hit Ratio, Replacement Algorithms

4.3 Virtual Memory Technique, Address Translation Method, TLB

4.4 Different Methods of I/O Access Mechanism

4.5 Programmed I/O, Interrupt Mechanism, DMA Data Transfer, I/O Processor

4.6 Types of Interrupt, Priority Interrupt, Simultaneous Interrupt

4.7 DMA Transfer Modes — Burst Mode, Cycle Stealing Mode

Prepared for Engineering / Polytechnic Students

4.1 Memory Hierarchy Model and Comparison on Cost, Speed and Size

A computer system uses several kinds of storage devices that differ widely in speed, cost per bit, and storage capacity. No single memory technology can be fast, cheap, and large at the same time, so designers arrange memory into a layered structure called the memory hierarchy. At the top of the hierarchy are very small, very fast, and very expensive memories close to the CPU; at the bottom are very large, slow, and cheap memories used for long-term storage.

Levels of the Memory Hierarchy (top to bottom): CPU Registers → Cache Memory (L1, L2, L3) → Main Memory (RAM) → Secondary Storage (Magnetic Disk / SSD) → Tertiary / Off-line Storage (Magnetic Tape, Optical Disk).

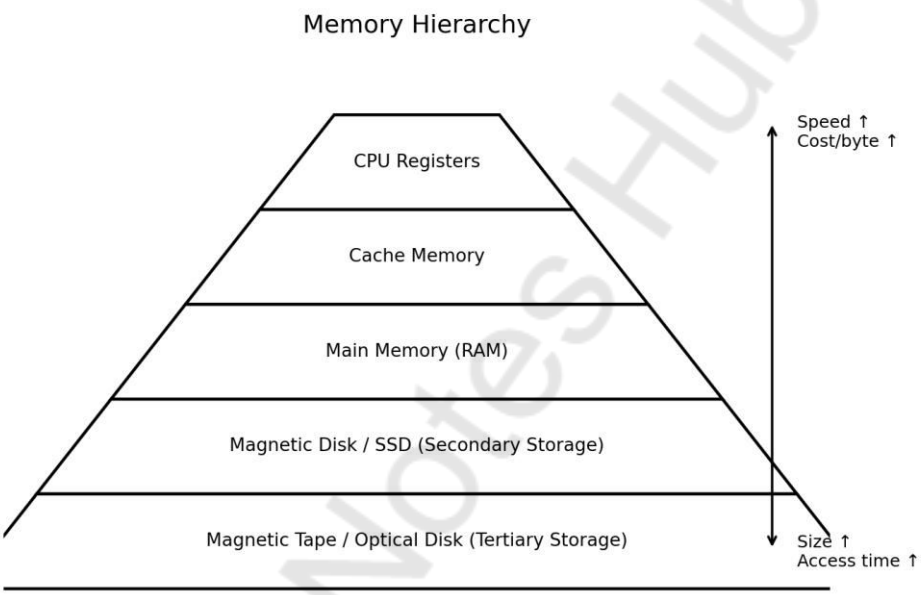

Fig. 4.1 — The Memory Hierarchy, showing the trade-off between speed, cost and size

Comparison of Memory Levels

Memory Level	Approx. Access Time	Cost per Bit	Typical Size
CPU Registers	< 1 ns	Highest	A few bytes (32–64 registers)
Cache Memory	1 – 10 ns	Very High	KB to a few MB
Main Memory (RAM)	50 – 100 ns	Moderate	GB range
Secondary Storage (HDD/SSD)	Milliseconds	Low	GB to TB
Tertiary Storage (Tape/Optical)	Seconds	Lowest	TB to PB

Key Observations

- **Speed vs Cost:** As we move up the hierarchy (towards the CPU), access speed increases but cost per bit rises sharply.
- **Size vs Access Time:** As we move down the hierarchy (away from the CPU), storage capacity increases but access time becomes slower.
- **Principle of Locality:** The memory hierarchy works efficiently because of temporal locality (recently used data is likely to be used again soon) and spatial locality (data near recently used data is likely to be used soon). Cache and virtual memory designs exploit this principle.
- **Inclusion Property:** Data present in a smaller, faster memory (e.g., cache) is normally also present in the larger, slower memory below it (e.g., main memory).

4.2 Cache Memory, Mapping Technique, Hit Ratio, Replacement Algorithm

Cache Memory

Cache memory is a small, high-speed memory placed between the CPU and main memory. Its purpose is to hold copies of the most frequently and recently used instructions and data so that the CPU does not have to wait for the much slower main memory on every access. Cache memory works on the principle of locality of reference — programs tend to access a relatively small portion of their address space repeatedly over a short period of time.

Working Principle: Whenever the CPU requests data, the cache is checked first. If the required word is found in the cache, it is called a cache hit and the data is supplied immediately. If it is not found, it is called a cache miss, and the block containing the required word is brought from main memory into the cache before being supplied to the CPU.

Cache Mapping Techniques

Mapping refers to the method used to decide where a block of main memory is placed inside the cache. There are three main mapping techniques:

1. Direct Mapping

Each block of main memory maps to exactly one specific cache line, calculated using the formula: Cache line number = (Main memory block number) modulo (Number of cache lines). It is simple and cheap to implement using a single comparator, but suffers from a high number of collisions if two frequently used blocks map to the same cache line (this problem is called thrashing).

2. Fully Associative Mapping

A main memory block can be placed in any available cache line. This offers maximum flexibility and the lowest miss rate, but requires comparing the tag against every line in the cache simultaneously, which needs expensive parallel comparator hardware, making it costly for large caches.

3. Set-Associative Mapping

This is a compromise between direct and fully associative mapping. The cache is divided into a number of sets, and each set contains a small number of lines (e.g., 2-way, 4-way set associative). A memory block

maps to a specific set (using modulo, like direct mapping) but can occupy any line within that set (like associative mapping). This technique is widely used in real processors because it balances hardware cost against performance.

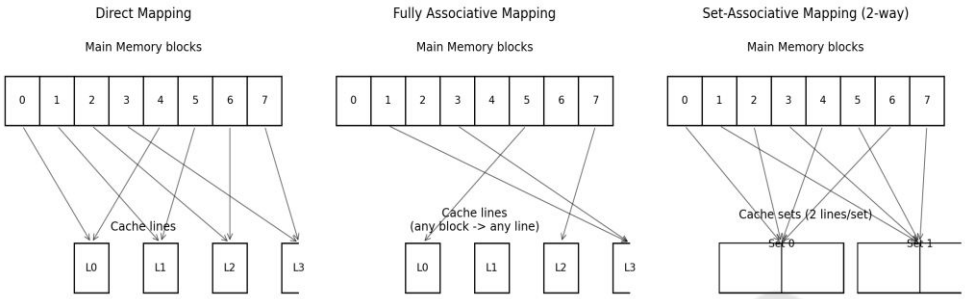

Fig. 4.2 — Direct, Fully Associative, and Set-Associative Cache Mapping

Hit Ratio

Hit ratio (H) is defined as the fraction of memory references that are found in the cache:

Hit Ratio (H) = Number of Hits / Total Number of Memory References

The average memory access time (Ta) of a system with cache can be expressed as: $Ta = H \times Tc + (1 - H) \times Tm$, where Tc is the cache access time and Tm is the main memory access time. A higher hit ratio results in an average access time closer to the fast cache access time, which is the main goal of cache design.

Cache Replacement Algorithms

When a cache miss occurs and the cache (or the relevant set) is full, an existing block must be replaced (evicted) to make room for the new block. Direct-mapped caches do not need a replacement algorithm since each block has only one possible location; replacement algorithms are needed for associative and set-associative caches.

- **FIFO (First-In-First-Out):** Replaces the block that has been in the cache the longest, regardless of how recently it was used.
- **LRU (Least Recently Used):** Replaces the block that has not been referenced for the longest time. It closely follows the principle of temporal locality and generally performs well, but is more complex/costly to implement in hardware.
- **LFU (Least Frequently Used):** Replaces the block that has been referenced the fewest number of times, on the assumption that infrequently used blocks are less likely to be used again.
- **Random Replacement:** Selects a block at random for replacement. It is very simple to implement in hardware and, surprisingly, performs reasonably well in practice.

4.3 Concept of Virtual Memory Technique, Address Translation Method, TLB

Concept of Virtual Memory

Virtual memory is a memory management technique that creates an illusion for the user/programmer of having a very large main memory, even though the actual physical main memory (RAM) may be much smaller. It does this by using secondary storage (hard disk/SSD) as an extension of main memory. Programs use logical addresses (also called virtual addresses) generated by the CPU, which are translated into physical addresses before the actual memory access takes place.

- **Advantages:** Allows programs larger than physical memory to run, enables efficient multiprogramming, provides memory protection between processes, and simplifies programming since the programmer need not worry about physical memory limits.
- **Implementation techniques:** Virtual memory is commonly implemented using Paging (dividing memory into fixed-size pages/frames) or Segmentation (dividing memory into variable-size logical segments), or a combination of both (paged segmentation).

Address Translation Method (Paging)

In a paging system, the logical address generated by the CPU is divided into two parts: a page number and a page offset (displacement within the page). The page number is used as an index into a page table, which stores the corresponding physical frame number for each page. The physical address is then formed by combining this frame number with the same page offset (the offset does not change during translation).

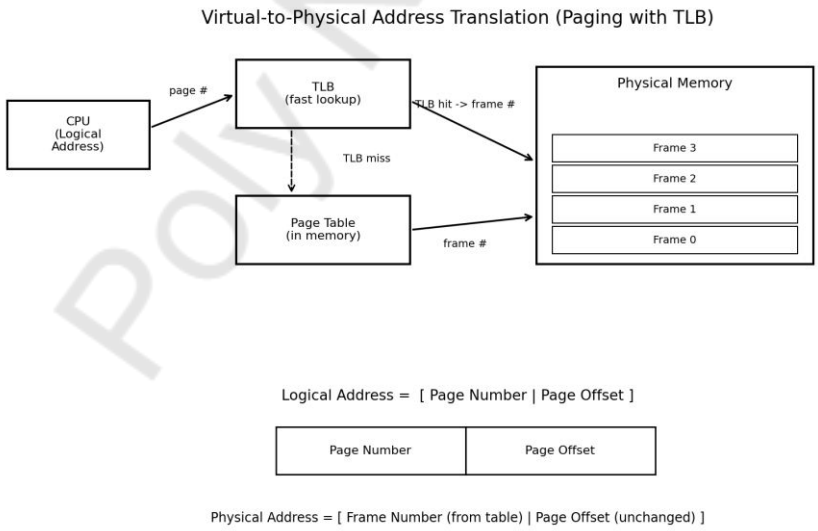

Fig. 4.3 — Virtual-to-Physical Address Translation using Paging and TLB

Page Fault: If the required page is not currently present in main memory (only on disk), a page fault occurs. The operating system then loads the required page from secondary storage into a free frame in main memory, updates the page table, and restarts the interrupted instruction. If main memory is full, a page replacement algorithm (e.g., LRU, FIFO, Optimal) is used to select a page to remove.

Translation Lookaside Buffer (TLB)

Since the page table itself resides in main memory, every ordinary memory access using paging would require two memory references — one to read the page table entry, and another to access the actual data. This doubles the effective access time. To solve this problem, a small, very fast, associative-memory cache called the Translation Lookaside Buffer (TLB) is used to store the most recently used page table entries (page number → frame number mappings).

- **TLB Hit:** If the required page number is found in the TLB, the corresponding frame number is obtained immediately without consulting the page table in memory, giving a very fast translation.
- **TLB Miss:** If the page number is not found in the TLB, the page table in main memory must be consulted; the retrieved entry is then also loaded into the TLB for potential future use.

Because most programs exhibit strong locality of reference, the TLB achieves a very high hit ratio in practice, making virtual memory access nearly as fast as direct physical memory access.

4.4 Different Methods of I/O Access Mechanism

Input/Output (I/O) devices operate at speeds very different from the CPU, and the CPU needs an organized mechanism to transfer data to and from these devices. There are three principal methods by which the CPU can communicate with, and transfer data to/from, I/O devices:

- **1. Programmed I/O (Status-Check I/O):** The CPU itself executes a program that continuously checks (polls) the status of the I/O device and performs the data transfer word by word. The CPU is fully occupied during the transfer.
- **2. Interrupt-Driven I/O:** The CPU issues a command to the device and continues with other useful work. When the device is ready to transfer data, it sends an interrupt signal to the CPU, which then temporarily suspends its current task to service the I/O request.
- **3. Direct Memory Access (DMA):** A dedicated hardware unit called the DMA controller directly transfers blocks of data between the I/O device and main memory without continuous CPU involvement, freeing the CPU almost entirely during the transfer.

In addition to these transfer mechanisms, I/O devices may be addressed using two different addressing schemes:

- **Isolated (I/O-Mapped) I/O:** I/O devices have a separate address space from memory, and distinct instructions (e.g., IN, OUT) are used to access them.
- **Memory-Mapped I/O:** I/O device registers share the same address space as main memory, and ordinary memory-reference instructions (e.g., MOV, LOAD, STORE) can be used to access I/O devices.

These three access methods (Programmed I/O, Interrupt-Driven I/O and DMA) represent an increasing level of hardware sophistication and a decreasing level of CPU involvement, and are discussed in detail in the following section.

4.5 Programmed I/O, Interrupt Mechanism, DMA Data Transfer, I/O Processor

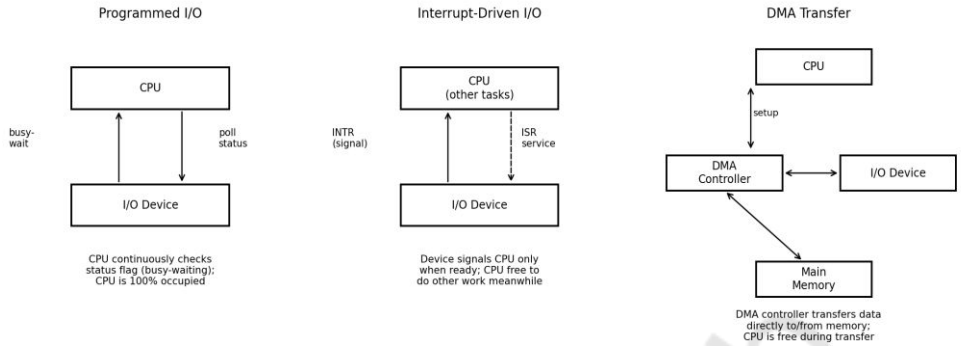

Fig. 4.4 — Programmed I/O, Interrupt-Driven I/O, and DMA compared

Programmed I/O (Status-Check I/O)

In programmed I/O, data transfer is completely controlled by a CPU program. Before transferring each word, the CPU reads the status register of the I/O device in a loop (busy-waiting) until the device signals that it is ready (e.g., a Busy or Ready flag is set). Only then does the CPU perform the actual data transfer instruction.

- **Advantage:** Very simple to implement; needs minimal extra hardware.
- **Disadvantage:** Extremely wasteful of CPU time, since the CPU can do nothing else while it waits for a slow I/O device to become ready.

Interrupt Mechanism

Interrupt-driven I/O overcomes the CPU-wastage problem of programmed I/O. The CPU issues a read/write command to the device and then proceeds with other tasks. The I/O device, once ready, raises an interrupt signal on a dedicated CPU line. On receiving this signal, the CPU completes its current instruction, saves its current state (program counter and registers) onto the stack, and branches to a special routine called the Interrupt Service Routine (ISR) to service the device. After the ISR completes, control returns to the interrupted program.

- **Advantage:** CPU time is used efficiently since it is not tied up in a wait loop.
- **Disadvantage:** Still involves the CPU for every word transferred, and has overhead due to context saving/restoring; not efficient for very high-speed, large block transfers.

DMA (Direct Memory Access) Data Transfer

For high-speed devices that need to transfer large blocks of data (e.g., disk drives), even interrupt-driven I/O is too slow because the CPU is still involved in every word transfer. DMA solves this using a special hardware unit called a DMA controller. The CPU only initializes the transfer by supplying the DMA controller with the starting memory address, the device address, and the number of words to be transferred. The DMA controller then takes over control of the system bus and directly transfers the

entire block of data between the I/O device and main memory, without CPU involvement, and interrupts the CPU only once — after the complete transfer is finished.

- **Advantage:** Extremely fast, since the CPU is free to perform other tasks during the entire block transfer; the CPU is interrupted only once per block instead of once per word.
- **Disadvantage:** Requires additional, relatively complex and costly hardware (the DMA controller).

I/O Processor (IOP) / I/O Channel

An I/O Processor (also called an I/O channel) is a specialized processor dedicated entirely to managing I/O operations, representing an even higher level of offloading than DMA. Unlike a simple DMA controller (which can only transfer raw blocks of data), an I/O processor can fetch and execute its own I/O programs (channel programs) stored in main memory, handle multiple I/O devices, perform data formatting/error checking, and communicate results to the main CPU only when the entire I/O task is complete. This further reduces the main CPU's involvement in I/O management and is typically used in large-scale/mainframe computer systems.

4.6 Different Types of Interrupt, Priority Interrupt, Simultaneous Interrupt

Types of Interrupts

Interrupts can be broadly classified as follows:

- **Hardware Interrupts (External Interrupts):** Generated by external hardware devices (e.g., keyboard, disk controller, timer) to request CPU attention. These may be further classified as:
  - Maskable Interrupts: Can be disabled (masked) by the programmer using an interrupt enable/disable instruction; ignored by the CPU when masked.
  - Non-Maskable Interrupts (NMI): Cannot be disabled; used for critical events such as power failure, which must always be serviced immediately.
- **Software Interrupts (Traps):** Generated intentionally by the execution of a special instruction in a program (e.g., INT instruction), commonly used to invoke operating system services (system calls).
- **Internal Interrupts (Exceptions/Traps):** Generated internally by the CPU due to some exceptional condition arising during program execution, such as division by zero, arithmetic overflow, or an illegal opcode.

Priority Interrupt

When a computer system has multiple I/O devices that can each generate an interrupt, a situation may arise where more than one device requests service. A priority interrupt system is used to establish which device's request should be serviced first, typically assigning higher priority to faster or more critical devices. Two common methods are used to establish interrupt priority:

- **1. Daisy-Chaining (Hardware Priority):** All devices requesting an interrupt are connected in a serial chain based on priority. The device closest to the CPU has the highest priority. When the CPU sends an interrupt-acknowledge signal, it passes through each device in the chain; a device that is requesting service blocks the signal from passing further and inserts its own interrupt vector address, so lower-priority devices further down the chain do not get acknowledged.
- **2. Parallel Priority Interrupt:** Uses a separate interrupt register (to record which devices are requesting) along with a mask register (to enable/disable individual device interrupts) and priority-encoder hardware to determine, among all pending requests, the one with the highest priority. This method is faster than daisy chaining as it does not depend on signal propagation through a chain of devices.
- **3. Software Polling:** The CPU, on receiving a general interrupt signal, polls (checks) each device one by one in a fixed priority order to determine which device requested service. It is simple but slow because it is a sequential, software-controlled process.

Simultaneous Interrupts

A simultaneous interrupt situation occurs when two or more I/O devices request CPU attention at exactly the same time. Since the CPU can service only one interrupt at a time, priority resolution hardware/software (as described above — daisy chaining, parallel priority encoders, or polling) is

essential to decide which of the simultaneously requesting devices should be serviced first, while the remaining requests are held pending and serviced afterward in priority order.

4.7 DMA Transfer Modes — Burst Mode, Cycle Stealing Mode

Once a DMA controller has been granted control of the system bus by the CPU, it can transfer data between the I/O device and main memory using one of two principal modes: burst mode or cycle stealing mode.

Burst Mode (Block Transfer Mode)

In burst mode, once the DMA controller takes control of the bus, it retains control of the bus continuously until the entire block of data has been transferred. The CPU is completely suspended from using the bus (and hence cannot access memory) for the entire duration of the block transfer. Only after the complete block has been transferred does the DMA controller release the bus back to the CPU and typically generate a single interrupt to signal completion.

- **Advantage:** The fastest possible mode of data transfer, ideal for high-speed devices that require large, continuous blocks of data (e.g., disk transfers).
- **Disadvantage:** The CPU is idle (with respect to bus/memory access) for the entire transfer period, which can degrade overall system performance if the block is very large.

Cycle Stealing Mode

In cycle stealing mode, the DMA controller transfers only one word at a time. After transferring a single word, it releases the bus back to the CPU for one machine cycle, then requests the bus again for the next word, and so on, until the whole block has been transferred. In effect, the DMA controller "steals" one memory cycle from the CPU periodically, forcing the CPU to pause only very briefly and intermittently rather than for the entire block.

- **Advantage:** The CPU is not completely blocked out — it can continue processing between DMA cycles, so the impact on CPU throughput is much smaller than with burst mode.
- **Disadvantage:** Overall data transfer is slower than burst mode, because bus arbitration overhead is incurred for every single word transferred.

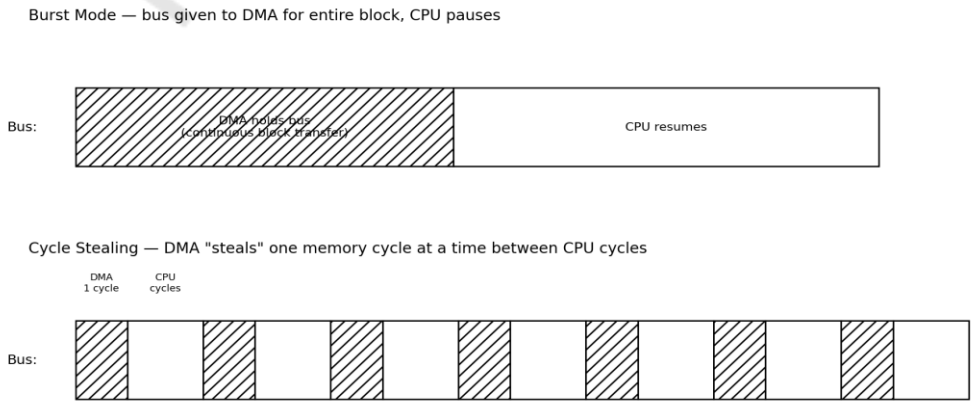

Fig. 4.5 — Burst Mode vs Cycle Stealing Mode of DMA Transfer

Comparison: Burst Mode vs Cycle Stealing Mode

Parameter	Burst Mode	Cycle Stealing Mode
Bus control	Held continuously for the whole block	Released after every single word
CPU involvement	CPU idle for the whole transfer	CPU pauses only briefly, periodically
Transfer speed	Fastest	Slower (more overhead)
Suitable for	High-speed devices, large blocks	Systems where CPU cannot be blocked for long

MCQs — 15 Multiple Choice Questions

1. **Which level of the memory hierarchy has the fastest access time?**
   (a) Cache memory
   (b) CPU registers
   (c) Main memory
   (d) Secondary memory

2. **As we move from the top to the bottom of the memory hierarchy:**
   (a) Speed increases and cost increases
   (b) Speed decreases and size increases
   (c) Speed decreases and size decreases
   (d) Cost increases and size increases

3. **Which cache mapping technique offers the greatest flexibility but is the most expensive to implement in hardware?**
   (a) Direct mapping
   (b) Set-associative mapping
   (c) Fully associative mapping
   (d) Sector mapping

4. **In direct mapping, block j of main memory is mapped to cache line number:**
   (a) j mod (number of memory blocks)
   (b) j mod (number of cache lines)
   (c) j / (number of cache lines)
   (d) Assigned randomly

5. **Hit ratio is defined as:**
   (a) Misses ÷ Total references
   (b) Hits ÷ Total references
   (c) Hits ÷ Misses
   (d) Total references ÷ Hits

6. **Which cache replacement algorithm replaces the block that has not been referenced for the longest time?**
   (a) FIFO
   (b) LFU
   (c) LRU
   (d) Random

7. **Virtual memory is most commonly implemented using:**
   (a) Cache memory only
   (b) Paging and/or Segmentation
   (c) CPU registers
   (d) DMA controllers

8. **TLB stands for:**
    (a) Table Lookaside Buffer
    (b) Translation Lookaside Buffer
    (c) Temporary Lookup Block
    (d) Transfer Logic Buffer

9. **A TLB miss means that:**
    (a) The required page is not in main memory at all
    (b) The page table entry is not currently present in the TLB
    (c) A cache miss has occurred
    (d) The DMA transfer has failed

10. **Which I/O technique keeps the CPU continuously busy checking the device status in a loop?**
    (a) Interrupt-driven I/O
    (b) Programmed I/O
    (c) DMA
    (d) I/O processor

11. **In interrupt-driven I/O, while waiting for the device to become ready, the CPU:**
    (a) Continuously polls the device
    (b) Is free to execute other tasks
    (c) Shuts down completely
    (d) Transfers data directly with no ISR

12. **DMA stands for:**
    (a) Direct Memory Access
    (b) Dynamic Memory Allocation
    (c) Data Memory Address
    (d) Direct Module Access

13. **The hardware unit that manages data transfer between the I/O device and memory without continuous CPU intervention is the:**
    (a) I/O processor
    (b) DMA controller
    (c) Interrupt controller
    (d) Cache controller

14. **Which interrupt priority scheme connects devices in a serial hardware chain, where the device nearest the CPU has the highest priority?**
    (a) Software polling
    (b) Daisy chaining
    (c) Parallel priority encoding
    (d) Vectored interrupt table

15. **In burst mode DMA transfer, the DMA controller:**
    (a) Transfers one word per cycle, sharing the bus with the CPU

(b) Holds the bus continuously until the entire block is transferred

(c) Is slower than cycle stealing mode

(d) Does not use the system bus at all

Answer Key

Q1 — (b) Q2 — (b) Q3 — (c) Q4 — (b) Q5 — (b)

Q6 — (c) Q7 — (b) Q8 — (b) Q9 — (b) Q10 — (b)

Q11 — (b) Q12 — (a) Q13 — (b) Q14 — (b) Q15 — (b)

Broad / Descriptive Questions — 10 Questions

1. Explain the memory hierarchy in a computer system with a neat diagram. Compare the different levels of memory on the basis of cost, speed, and size.
2. What is cache memory? Explain direct mapping, fully associative mapping, and set-associative mapping techniques with suitable examples/diagrams.
3. Define hit ratio and derive the expression for average memory access time in a two-level (cache – main memory) system.
4. Explain any three cache replacement algorithms (FIFO, LRU, LFU, Random) with their advantages and disadvantages.
5. What is virtual memory? Explain the concept of paging and the method of address translation using a page table, with a diagram.
6. Explain the role of the Translation Lookaside Buffer (TLB) in virtual memory address translation. Differentiate between a TLB hit and a TLB miss.
7. Compare programmed I/O, interrupt-driven I/O, and DMA transfer with respect to CPU involvement, speed, and hardware complexity.
8. Explain the working of DMA data transfer with a neat block diagram. Describe the role of the DMA controller and the I/O processor.
9. What is a priority interrupt? Explain the daisy-chaining method and the parallel priority interrupt method for establishing interrupt priority.
10. Explain the burst mode and cycle stealing mode of DMA data transfer with suitable diagrams, and differentiate between them.