Computer Organization and Architecture

Exercise & Exam-Oriented Questions and Answers

Chapter 4: Memory and I/O Devices

Contents

Based on Chapter 4 — Memory and I/O Devices, for Engineering / Polytechnic Students

Section A — Very Short Answer Questions (1 Mark)

Q1. What is memory hierarchy?

Ans. Memory hierarchy is the arrangement of different storage devices — registers, cache, main memory, secondary storage — in layers according to their speed, cost, and size, from the fastest and costliest at the top to the slowest and cheapest at the bottom.

Q2. Define cache memory.

Ans. Cache memory is a small, high-speed memory placed between the CPU and main memory that stores frequently and recently used data/instructions so as to reduce the average memory access time.

Q3. What is hit ratio?

Ans. Hit ratio is the fraction of total memory references that are found in the cache. It is given by Hit Ratio = Hits ÷ (Hits + Misses).

Q4. Define virtual memory.

Ans. Virtual memory is a memory management technique that gives the programmer the illusion of a very large main memory by using secondary storage (disk) as an extension of physical RAM.

Q5. What is TLB?

Ans. TLB (Translation Lookaside Buffer) is a small, fast associative-memory cache that stores recently used page table entries so as to speed up virtual-to-physical address translation.

Q6. What is DMA?

Ans. DMA (Direct Memory Access) is a technique that allows an I/O device to transfer data directly to/from main memory through a DMA controller, without continuous CPU intervention.

Q7. Define page fault.

Ans. A page fault occurs when a program refers to a page that is not currently present in main memory, causing the operating system to load it from secondary storage.

Q8. What is cycle stealing?

Ans. Cycle stealing is a DMA transfer mode in which the DMA controller transfers one word at a time, "stealing" a single memory cycle from the CPU between its normal instruction cycles.

Q9. What is an interrupt?

Ans. An interrupt is a signal sent to the CPU by hardware or software to indicate that an event needs immediate attention; the CPU suspends its current task and executes an Interrupt Service Routine (ISR).

Q10. What is daisy chaining?

Ans. Daisy chaining is a hardware method of establishing interrupt priority in which devices are connected in series; the device electrically closest to the CPU is given the highest priority.

Q11. Define programmed I/O.

Ans. Programmed I/O is an I/O technique in which the CPU itself continuously monitors (polls) the status of the device and personally carries out every data transfer.

Q12. What is an I/O processor?

Ans. An I/O processor (I/O channel) is a specialised processor that independently executes its own I/O programs and manages I/O devices, relieving the main CPU of detailed I/O control.

Q13. Define locality of reference.

Ans. Locality of reference is the tendency of a program to repeatedly access a small, localised set of memory locations over a short interval of time; it may be temporal or spatial.

Q14. What is a page table?

Ans. A page table is a data structure maintained by the operating system that stores the mapping between logical (virtual) page numbers and physical frame numbers.

Q15. Define burst mode DMA.

Ans. Burst mode is a DMA transfer mode in which the DMA controller retains control of the system bus continuously until an entire block of data has been transferred.

Q16. What is set-associative mapping?

Ans. Set-associative mapping is a cache mapping technique in which the cache is divided into sets of fixed size; a memory block maps to one specific set but may occupy any line within that set.

Q17. What is a non-maskable interrupt (NMI)?

Ans. A non-maskable interrupt is a high-priority interrupt that cannot be disabled by software; it is reserved for critical events such as power failure.

Q18. What is thrashing in the context of cache memory?

Ans. Thrashing occurs in direct-mapped caches when two or more frequently used blocks map to the same cache line, causing repeated replacement and a high miss rate.

Section B — Short Answer Questions (5 Marks)

Q1. Differentiate between Direct Mapping and Fully Associative Mapping.

Basis	Direct Mapping	Fully Associative Mapping
Block placement	A block can go into only one fixed cache line	A block can go into any available cache line
Hardware needed	Simple — one comparator	Complex — one comparator per line
Miss rate	Higher (collisions/thrashing possible)	Lowest, since placement is fully flexible
Replacement algorithm	Not required	Required (LRU, FIFO, LFU, etc.)
Cost	Low	High

Q2. Explain average memory access time with its formula, and illustrate with a short example.

Answer:

The average memory access time (Ta) of a system that uses a cache is given by:

$$Ta = H \times Tc + (1 - H) \times Tm$$

where H is the hit ratio, Tc is the cache access time, and Tm is the main memory access time. A higher hit ratio drives the average access time closer to the fast cache access time.

Example: If H = 0.95, Tc = 5 ns and Tm = 80 ns, then Ta = 0.95 × 5 + 0.05 × 80 = 4.75 + 4 = 8.75 ns — far closer to the cache speed than to the main memory speed.

Q3. Differentiate between Cache Memory and Virtual Memory.

Basis	Cache Memory	Virtual Memory
Purpose	Speeds up CPU-to-memory access	Extends the effective size of main memory
Positioned between	CPU and main memory	Main memory and secondary storage
Managed by	Hardware	Operating system, with hardware (MMU/TLB) support
Unit of mapping	Block / line	Page / frame
On a miss	Block fetched from main memory (fast)	Page fault — page fetched from disk (slow)

Q4. Explain the steps involved in handling a page fault.

Answer:

1. The CPU generates a logical address that refers to a page whose valid/invalid bit in the page table is marked invalid.
2. This causes a trap (page-fault interrupt) to the operating system.

3. The OS checks whether the reference was a valid memory access; if invalid, the process is aborted.
4. If valid, the OS looks for a free frame in main memory. If none is free, a page-replacement algorithm (e.g., LRU/FIFO) selects a victim page, writing it back to disk if it has been modified.
5. The OS schedules a disk I/O operation to read the required page into the chosen free frame.
6. The page table is updated with the new frame number, and the valid bit is set.
7. The instruction that caused the page fault is restarted from the beginning.

Q5. Differentiate between a TLB hit and a TLB miss.

Answer:

A TLB hit occurs when the page number required for translation is already present in the TLB; the corresponding frame number is obtained immediately, giving a fast translation without a page-table access. A TLB miss occurs when the page number is not found in the TLB; the CPU must then access the page table in main memory to obtain the frame number, which is slower, and the retrieved entry is loaded into the TLB for future use. Because of locality of reference, TLB hit ratios are usually very high in practice, so most accesses are fast.

Q6. Compare Programmed I/O and Interrupt-Driven I/O.

Basis	Programmed I/O	Interrupt-Driven I/O
CPU involvement	CPU continuously polls status (busy-wait)	CPU is free until the device interrupts it
CPU efficiency	Low — CPU time wasted waiting	Higher — CPU does useful work meanwhile
Hardware complexity	Simple	Needs interrupt hardware and an ISR
Suitable for	Very simple or very slow devices	Devices with moderate/variable response time

Q7. Explain the working of a DMA controller with its main registers.

Answer:

A DMA controller typically contains three main registers: an Address Register (holds the starting main memory address for the transfer), a Word Count Register (holds the number of words still to be transferred), and a Control/Status Register (specifies the transfer mode — read/write, burst/cycle-stealing — and the status of the transfer).

To begin a transfer, the CPU loads these registers with the starting address, word count, and transfer direction, and then instructs the DMA controller to start. The DMA controller then requests control of the system bus (bus request/bus grant), and once granted, it directly transfers data between the I/O device and main memory, incrementing the address register and decrementing the word count register after every word, until the count reaches zero. It then releases the bus and interrupts the CPU to signal that the transfer is complete.

Q8. Differentiate between Maskable and Non-Maskable Interrupts.

Basis	Maskable Interrupt	Non-Maskable Interrupt (NMI)
Can be disabled?	Yes, by the programmer using an enable/disable instruction	No, it can never be disabled
Priority	Normally lower priority	Highest priority
Typical use	Routine I/O device requests	Critical events, e.g., power failure

Q9. Explain the LRU (Least Recently Used) replacement algorithm with a suitable example.

Answer:

LRU replaces the block that has not been referenced for the longest period of time, on the assumption that a block unused for a long time is unlikely to be needed again soon (temporal locality).

Example: Consider a fully associative cache with only 3 lines, initially empty, and the following sequence of block references: 1, 2, 3, 4, 1, 2, 5.

- References 1, 2, 3 → all are misses; cache fills up with blocks [1, 2, 3].
- Reference 4 → miss; the least recently used block, 1, is replaced → cache becomes [2, 3, 4].
- Reference 1 → miss again (1 was evicted); the least recently used block, 2, is replaced → cache becomes [3, 4, 1].
- Reference 2 → miss; least recently used block, 3, is replaced → cache becomes [4, 1, 2].
- Reference 5 → miss; least recently used block, 4, is replaced → cache becomes [1, 2, 5].

This shows how LRU keeps the most recently accessed blocks in the cache and evicts the ones used longest ago.

Q10. Differentiate between Burst Mode and Cycle Stealing Mode of DMA transfer.

Basis	Burst Mode	Cycle Stealing Mode
Bus control	Held continuously for the entire block	Released after every single word
CPU involvement	CPU idle for the whole transfer	CPU pauses only briefly, periodically
Speed	Fastest	Slower (more bus-arbitration overhead)

Q11. What is the need for an I/O processor over a simple DMA controller?

Answer:

A DMA controller can only transfer a raw, fixed block of data between memory and a device; it has no ability to interpret data, handle multiple devices intelligently, or make decisions. As the number and speed of I/O devices in a system grows, this becomes a limitation.

An I/O processor removes this limitation because it is a dedicated processor capable of fetching and executing its own I/O programs (channel programs) stored in memory. It can manage several devices at once, perform data formatting and error-checking, and interrupt the main CPU only when an entire complex I/O task is complete — rather than for every block. This offloads I/O management almost

completely from the main CPU, which is essential in large, multi-device systems such as mainframes and servers.

Q12. Explain the Parallel Priority Interrupt method.

Answer:

In the parallel priority interrupt method, dedicated hardware is used instead of a serial daisy-chain to resolve which of several simultaneously requesting devices should be serviced first. It typically uses:

- **Interrupt Register:** one bit per device, set when that device raises an interrupt request.
- **Mask Register:** one bit per device, used to individually enable or disable interrupts from that device.
- **Priority Encoder:** combinational logic that examines all pending (unmasked) requests simultaneously and generates the address/vector of the highest-priority device requesting service.

Because all requests are examined in parallel rather than being passed serially through a chain of devices, this method is significantly faster than daisy chaining, though it requires more dedicated hardware.

Section C — Numerical / Problem-Based Questions (Solved)

Q1. A computer system has a hit ratio of 0.92, a cache access time of 8 ns, and a main memory access time of 90 ns. Calculate the average memory access time.

Answer:

Given: H = 0.92, Tc = 8 ns, Tm = 90 ns

Ta = H × Tc + (1 − H) × Tm

Ta = (0.92 × 8) + (0.08 × 90)

Ta = 7.36 + 7.2 = 14.56 ns

Answer: The average memory access time is 14.56 ns.

Q2. A direct-mapped cache has 256 lines. Main memory has 8192 blocks. Find the cache line to which main memory block number 5000 will be mapped.

Answer:

Given: Number of cache lines = 256, Block number = 5000

Cache line number = Block number mod Number of cache lines

5000 ÷ 256 = 19 remainder 136 (since 19 × 256 = 4864, and 5000 − 4864 = 136)

Answer: Block 5000 maps to cache line number 136.

Q3. A system has a 24-bit logical address space, and the page size is 4 KB. Find the number of bits used for the page offset, the number of bits for the page number, and the total number of pages.

Answer:

Given: Logical address = 24 bits, Page size = 4 KB = 2^{12} bytes

Page offset bits = $\log_2$(page size) = $\log_2(2^{12}) = 12$ bits

Page number bits = Total address bits − Offset bits = 24 − 12 = 12 bits

Number of pages = 2^(page number bits) = 2^{12} = 4096 pages

Answer: Offset = 12 bits, Page number = 12 bits, Total pages = 4096.

Q4. In a system, the average memory access time is 20 ns, the cache access time is 5 ns, and the main memory access time is 105 ns. Find the hit ratio.

Answer:

Given: Ta = 20 ns, Tc = 5 ns, Tm = 105 ns

Ta = H × Tc + (1 − H) × Tm

20 = 5H + 105(1 − H) = 5H + 105 − 105H

20 − 105 = −100H $\implies$ −85 = −100H $\implies$ H = 0.85

Answer: The hit ratio is 0.85 (i.e., 85%).

Q5. A system has a TLB access time of 10 ns and a TLB hit ratio of 90%. Main memory access time is 100 ns. On a TLB miss, the page table (in main memory) must be accessed before the actual data (also in main memory) is accessed. Find the effective memory access time.

Answer:

On a TLB hit: Time = TLB access + Memory access for data = 10 + 100 = 110 ns

On a TLB miss: Time = TLB access + Page table access (memory) + Data access (memory) = 10 + 100 + 100 = 210 ns

Effective Access Time = (Hit ratio × Hit time) + (Miss ratio × Miss time)

Effective Access Time = (0.90 × 110) + (0.10 × 210) = 99 + 21 = 120 ns

Answer: The effective memory access time is 120 ns.

Section D — Long Answer Questions (10 Marks)

Q1. 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.

Answer:

A computer system uses several kinds of storage devices differing widely in speed, cost per bit, and capacity. Since no single memory technology can be fast, cheap, and large at the same time, memory is arranged in a layered structure called the memory hierarchy — very small, fast, expensive memory near the CPU, and very large, slow, cheap memory at the bottom.

Levels (top to bottom): CPU Registers → Cache Memory → Main Memory (RAM) → Secondary Storage (Magnetic Disk/SSD) → Tertiary Storage (Magnetic Tape/Optical Disk).

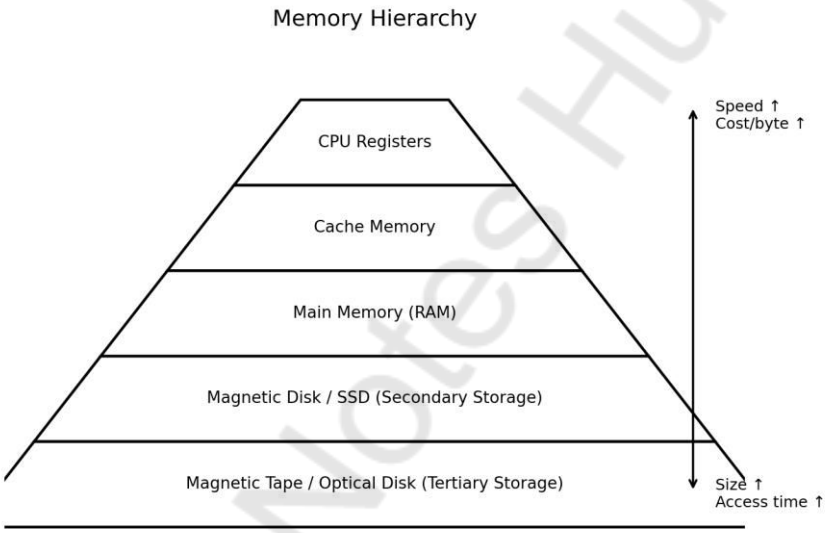


Fig. — The Memory Hierarchy

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

As we move up the hierarchy, speed increases but cost per bit rises sharply; as we move down, capacity increases but access time becomes slower. The hierarchy works efficiently because of the principle of locality of reference (temporal and spatial), which allows small, fast memories to satisfy most references.

Q2. What is cache memory? Explain direct mapping, fully associative mapping, and set-associative mapping techniques with suitable diagrams.

Answer:

Cache memory is a small, high-speed memory placed between the CPU and main memory to hold copies of frequently/recently used instructions and data, reducing the average memory access time by exploiting locality of reference.

Mapping decides where a main-memory block is placed inside the cache. There are three techniques:

- **Direct Mapping:** Each block maps to exactly one cache line, given by (block number) mod (number of lines). Simple and cheap, but prone to collisions/thrashing.
- **Fully Associative Mapping:** A block may be placed in any available line. Gives the lowest miss rate but needs parallel comparators for every line, making it expensive.
- **Set-Associative Mapping:** A compromise — the cache is divided into sets of a few lines each; a block maps to a specific set (as in direct mapping) but can occupy any line within that set (as in associative mapping). This is the technique most commonly used in real processors.

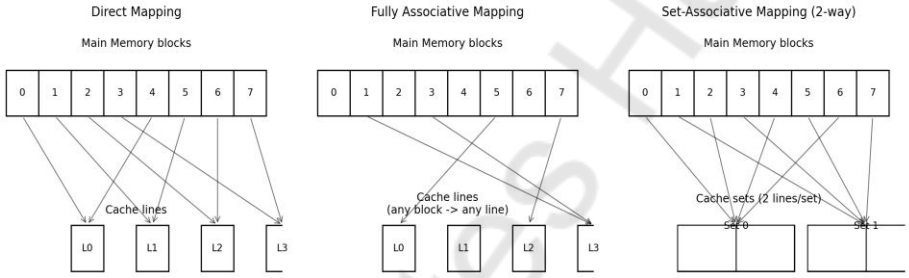


Fig. — Direct, Fully Associative, and Set-Associative Mapping

Q3. Define hit ratio and derive the expression for average memory access time in a two-level (cache–main memory) system.

Answer:

Hit ratio (H) is the fraction of total memory references that are satisfied by the cache: H = Hits ÷ (Hits + Misses). The miss ratio is simply (1 − H).

Derivation: Let Tc be the cache access time and Tm be the main memory access time. Out of every memory reference, a fraction H is a hit (satisfied in time Tc) and a fraction (1 − H) is a miss, which must first fetch the block from main memory before it is available (taking time Tm, in the simplest model). The average (expected) access time is the weighted sum of these two cases:

Ta = H × Tc + (1 − H) × Tm

Since Tc ≪ Tm, and H is normally kept very close to 1 by good cache design, Ta stays close to Tc, giving the CPU an effective access time much closer to the cache's speed than to main memory's speed, even though main memory is far larger. For example, with H = 0.9, Tc = 10 ns, and Tm = 100 ns, Ta = 0.9×10 + 0.1×100 = 9 + 10 = 19 ns — much closer to 10 ns than to 100 ns.

Q4. Explain any three cache replacement algorithms (FIFO, LRU, LFU) with their advantages and disadvantages.

Answer:

A replacement algorithm decides which existing block to evict when a cache miss occurs and the cache (or relevant set) is full. Direct-mapped caches do not need one, since each block has only one possible location.

- **FIFO (First-In-First-Out):** Replaces the block that has been in the cache the longest, regardless of how recently it was used. Advantage: simple to implement (just a queue). Disadvantage: may evict a block that is still being used frequently, simply because it was loaded early.
- **LRU (Least Recently Used):** Replaces the block that has not been referenced for the longest time. Advantage: follows the principle of temporal locality closely and generally gives a good hit ratio. Disadvantage: needs extra hardware/counters to track usage order, making it more expensive to implement exactly, especially for large caches.
- **LFU (Least Frequently Used):** Replaces the block that has been referenced the fewest number of times. Advantage: keeps genuinely popular blocks in cache longer. Disadvantage: a block that was frequently used in the past but is no longer needed may remain in the cache for a long time (it "looks" popular from old counts), and counters add hardware overhead.

(A fourth common technique, Random Replacement, simply picks a line at random; it is extremely simple in hardware and performs surprisingly well in practice, though it is less predictable.)

Q5. What is virtual memory? Explain the concept of paging and the method of address translation using a page table, with a diagram.

Answer:

Virtual memory is a memory management technique that gives the programmer the illusion of having a very large main memory, even though actual physical RAM may be much smaller, by using secondary storage (disk) as an extension of main memory. Programs use logical (virtual) addresses generated by the CPU, which are translated into physical addresses before the actual memory access.

Paging divides both logical address space and physical memory into fixed-size blocks — pages (logical) and frames (physical) — of the same size. The logical address generated by the CPU is split into a page number and a page offset. The page number indexes into a page table (maintained by the OS) which gives the corresponding physical frame number; the physical address is then formed by combining this frame number with the unchanged offset.

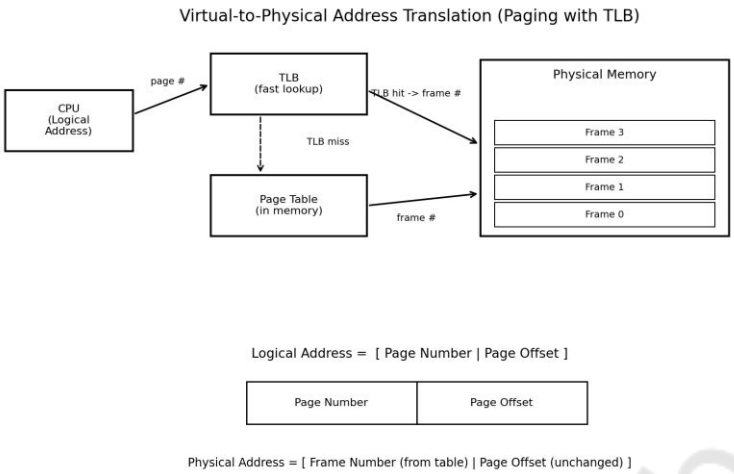


Fig. — Address Translation using Paging (with TLB)

If the required page is not currently in memory, a page fault occurs, and the operating system brings it in from disk, replacing another page if memory is full (using a page-replacement algorithm such as LRU or FIFO), before the instruction is restarted.

Q6. Explain the role of the Translation Lookaside Buffer (TLB) in virtual memory address translation. Differentiate between a TLB hit and a TLB miss.

Answer:

Since the page table resides in main memory, a naive paging scheme would need two memory references for every access — one to read the page table entry, another to access the actual data — doubling effective access time. The Translation Lookaside Buffer (TLB) is a small, very fast, associative-memory cache that stores the most recently used page-number-to-frame-number mappings, avoiding this overhead for most accesses.

- **TLB Hit:** The required page number is found in the TLB; the frame number is obtained immediately without consulting the page table in memory, giving fast translation.
- **TLB Miss:** The page number is not in the TLB; the page table in main memory must be read to get the frame number (extra memory access), and the entry is then loaded into the TLB for possible future use.

Because programs show strong locality of reference, the TLB achieves a very high hit ratio in practice, making paged virtual memory access nearly as fast as direct physical memory access despite the extra translation step.

Q7. Compare programmed I/O, interrupt-driven I/O, and DMA transfer with respect to CPU involvement, speed, and hardware complexity.

Answer:

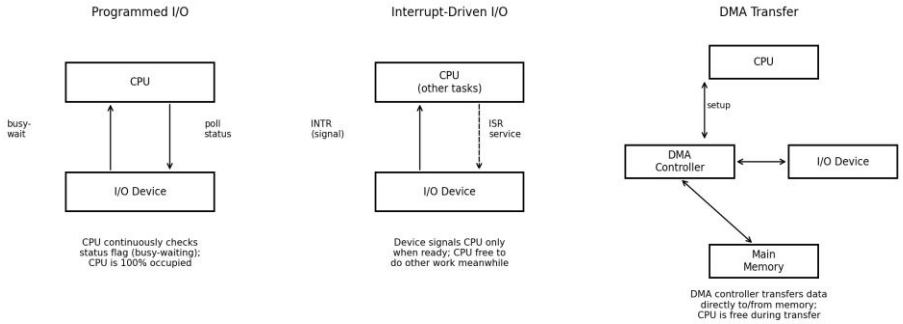


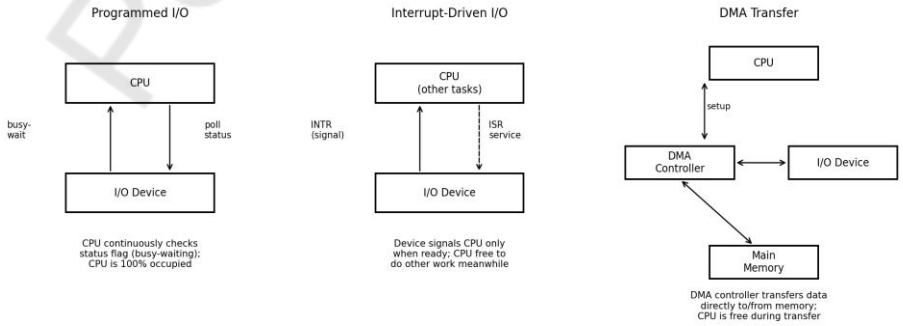
Fig. — Programmed I/O, Interrupt-Driven I/O, and DMA compared

Basis	Programmed I/O	Interrupt-Driven I/O	DMA
CPU involvement	Very high — CPU busy-waits and transfers every word	Moderate — CPU free until interrupted, then transfers word(s)	Very low — CPU only sets up transfer; hardware does the rest
Speed	Slowest (CPU is the bottleneck)	Faster than programmed I/O	Fastest — suited to large, high-speed block transfers
Hardware complexity	Simplest — no extra hardware	Needs interrupt controller and ISR	Needs a dedicated DMA controller
Best suited for	Simple, slow, low-priority devices	Devices with moderate/variable speed	High-speed devices with large data blocks, e.g. disks

Q8. Explain the working of DMA data transfer with a neat block diagram. Describe the role of the DMA controller and the I/O processor.

Answer:

For high-speed devices transferring large blocks of data, even interrupt-driven I/O is inefficient, since the CPU is still involved in every word. DMA solves this using a dedicated DMA controller. The CPU only initialises the transfer — supplying the starting memory address, the device address, the transfer direction, and the word count — after which the DMA controller takes control of the system bus and directly transfers the complete block of data between the device and main memory, without further CPU involvement, interrupting the CPU only once, when the whole block has been transferred.



An I/O Processor (I/O channel) extends this idea further: it is a specialised processor dedicated to I/O management that can fetch and execute its own I/O programs, handle multiple devices, perform data formatting/error checking, and interrupt the main CPU only when an entire I/O task is complete —

offloading even more I/O responsibility from the CPU than a simple DMA controller, and is typically found in large/mainframe systems.

Q9. What is a priority interrupt? Explain the daisy-chaining method and the parallel priority interrupt method for establishing interrupt priority.

Answer:

When a system has multiple I/O devices that can each raise an interrupt, more than one device may request service at the same time. A priority interrupt system establishes which request is serviced first, normally giving higher priority to faster or more critical devices.

- **Daisy Chaining (Hardware/Serial priority):** All interrupting devices are connected in a serial chain; the device closest to the CPU has the highest priority. When the CPU sends an interrupt-acknowledge signal, it passes down the chain; a requesting device blocks the signal from going further and supplies its own interrupt-vector address, so lower-priority devices further down do not get acknowledged in that cycle.
- **Parallel Priority Interrupt:** Uses an interrupt register (records which devices are requesting), a mask register (enables/disables individual devices), and priority-encoder hardware that examines all pending requests simultaneously and outputs the address of the highest-priority one. This is faster than daisy chaining since it does not depend on a signal propagating through a chain of devices, though it needs more dedicated hardware.

A related software method, polling, has the CPU check each device's status in a fixed order to find out which one requested service; it is simple but slower because it is purely sequential.

Q10. Explain the burst mode and cycle stealing mode of DMA data transfer with suitable diagrams, and differentiate between them.

Answer:

Once granted control of the system bus, a DMA controller can transfer data using one of two modes:

- **Burst Mode (Block Transfer Mode):** The DMA controller retains control of the bus continuously until the entire block of data has been transferred; the CPU is completely suspended from using the bus for that whole period. This is the fastest mode, ideal for high-speed devices needing large, continuous transfers, but it blocks the CPU for the full duration.
- **Cycle Stealing Mode:** The DMA controller transfers only one word at a time, releasing the bus back to the CPU after every word and requesting it again for the next. In effect, it "steals" one memory cycle at a time. This keeps the CPU only briefly and intermittently paused rather than blocked for the whole block, but is slower overall due to repeated bus-arbitration overhead.

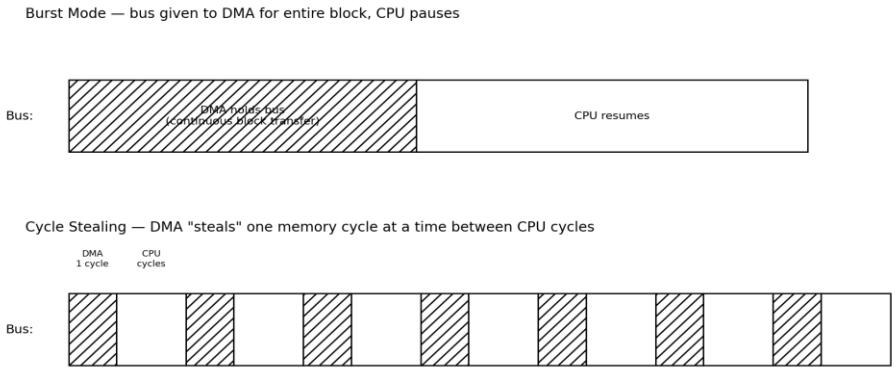


Fig. — Burst Mode vs Cycle Stealing Mode

Parameter	Burst Mode	Cycle Stealing Mode
Bus control	Held for the entire block	Released after every word
CPU impact	Idle for the whole transfer	Paused only briefly, periodically
Speed	Fastest	Slower (more overhead)