1. *Define a Race Condition in the context of parallel programming and briefly state how it is typically solved in software ?*

Race Condition

- A **race condition** occurs when multiple threads/processes **access and modify shared data simultaneously**,
  and the final result depends on **which thread runs first**, leading to unpredictable behavior.

How it is Typically Solved

- Use **mutual exclusion** techniques so only one thread accesses critical shared data at a time.
- Common synchronization tools include **locks**, **mutexes**, **semaphores**, **monitors**, and **atomic operations**.
- Designing **thread-safe data structures** and using **synchronized blocks** can also prevent race conditions.

2. *Differentiate between a Multi-core CPU and a GPU based on their core count and intended use case.*

Difference Between Multi-core CPU and GPU

1. Core Count

- **Multi-core CPU**
  - Has **few powerful cores** (typically 2–16 cores in consumer systems).
  - Each core is optimized for **complex, sequential tasks** and multitasking.
- **GPU**
  - Has **hundreds to thousands of smaller, simpler cores**.
  - Designed for **massively parallel tasks** where the same operation is repeated many times.

---

2. Intended Use Case

- **Multi-core CPU**
    - Best for tasks requiring **logic, decision-making, branching**, and low-latency response.
    - Used for operating systems, application logic, browsers, running programs, etc.
- **GPU**
    - Best for tasks that can be parallelized, such as **graphics rendering**, **AI/ML computations**, **video processing**, and **scientific simulations**.
    - Optimized for doing many similar calculations at the same time.

Example

- When playing a game:
    - **CPU** handles game logic → player movement, physics, AI decisions.
    - **GPU** renders graphics → calculating millions of pixels and shapes in parallel.

3. Explain the "I/O Bottleneck" and provide one common hardware solution to mitigate it.

I/O Bottleneck

- An **I/O bottleneck** occurs when a system's input/output devices (like hard drives or network interfaces) are **too slow** compared to the CPU.
- The CPU ends up **waiting idle** because data cannot be read/written fast enough.
- This slows down overall system performance even if the processor is very powerful.

Common Hardware Solution

- **Use faster storage devices**, such as upgrading from **HDD → SSD** or from **SATA SSD → NVMe SSD**.
- These provide much higher read/write speeds, reducing wait time for the CPU.

Example

- If a computer uses a slow HDD, opening a large application takes time because the disk loads data slowly.
- After upgrading to an **NVMe SSD**, the same application loads **much faster**, because the CPU receives data without long delays—reducing the I/O bottleneck.

4. In one sentence, what is the main advantage of an ASIC over an FPGA for a fixed, high-volume computational task?

Main Advantage of an ASIC Over an FPGA

- An **ASIC** provides **much higher performance and lower power consumption** than an FPGA because it is **custom-built and optimized** specifically for a fixed, high-volume task.

Why ASIC Is Better

- **Fully optimized hardware** → faster execution for a specific task.
- **Lower power usage** → energy efficient for large-scale deployment.
- **Smaller unit cost** when produced in high volumes.
- **Better heat management** due to streamlined circuitry.

Example

- A company producing millions of smartphones uses an **ASIC** for video processing because it performs the task faster and consumes less battery than a reconfigurable **FPGA**.

References

Hennessy, J. L., & Patterson, D. A. (2019). Computer Architecture: A Quantitative Approach (6th ed.). Morgan Kaufmann.
Patterson, D. A., & Hennessy, J. L. (2017). Computer Organization and Design: The Hardware/Software Interface (5th ed.). Morgan Kaufmann.
Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts (10th ed.). Wiley.
Stallings, W. (2018). Operating Systems: Internals and Design Principles (9th ed.). Pearson.
Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E., & Phillips, J. C. (2008). GPU Computing. Proceedings of the IEEE.