**Lecture 1 & 2 (Basics & Performance)**

* Difference between **Computer Architecture** (visible to programmer) vs. **Computer Organization** (implementation details).
* Structural view of a computer: CPU, Memory, I/O, System Interconnection.
* History: ENIAC, Von Neumann, IBM/360, PDP series, Intel 4004 → Pentium.
* Generations of computers: Vacuum tube → Transistor → IC → VLSI → ULSI.
* **Performance concepts**: Instruction cycle, Fetch/Execute, CPI, MIPS, Amdahl’s Law.
* Examples and practice questions on performance measurement.

**Lecture 3 (System Interconnections & Buses)**

* Interconnection of CPU, Memory, I/O modules.
* Concept of **Bus** (Data, Address, Control).
* Problems with single bus → multiple bus hierarchies.
* **Bus Arbitration** (centralized vs distributed).
* **DMA (Direct Memory Access)** & Bus Mastering.
* **Synchronous vs Asynchronous buses**, timing diagrams.
* PCI Bus, arbitration, master/target communication.

**Lecture 4 (Arithmetic & ALU)**

* **Arithmetic Logic Unit (ALU)** operations.
* Binary number representation (sign/magnitude, 2’s complement).
* Rules of addition, subtraction (overflow detection).
* Hardware block diagrams for addition/subtraction.
* **Multiplication**: Unsigned integers, signed integers, Booth’s algorithm.
* **Division**: unsigned, signed, handling remainder signs.
* Division algorithm hardware and examples.

**Lecture 5 (Floating Point Arithmetic & IEEE 754)**

* Real numbers, fixed-point vs floating-point representation.
* **IEEE 754 Floating Point Format** (single precision).
* Sign, exponent, fraction fields, biased exponent.
* Examples: conversion between decimal and floating point.
* Special cases: Zero, Infinity, NaN.
* Floating-point addition, subtraction, normalization, rounding.
* Floating-point multiplication with examples.

**🔹 Lecture 1 & 2: Basics & Performance**

1. Differentiate between **computer architecture** and **computer organization** with examples.
2. What are the four main functions of a computer system?
3. Explain the **structural view** of a computer with a neat diagram.
4. What are the main structural components of the CPU?
5. Explain the role of the **Control Unit**, **ALU**, and **Registers**.
6. Write a note on the **Von Neumann architecture**. What is the stored program concept?
7. Brief history: Describe the contributions of **ENIAC**, **IBM/360**, and **Intel 4004**.
8. List the **five computer generations** with their technologies.
9. What is the difference between **CISC** and **RISC** architectures?
10. Define **Instruction Cycle**. Differentiate between Fetch and Execute cycles.
11. Derive the **CPU performance equation**.
12. Explain **CPI (Cycles Per Instruction)** and **MIPS** with examples.
13. What is **Amdahl’s Law**? State its formula and significance.
14. Example Q: If Computer A has 2 GHz clock and runs a program in 10s, while Computer B has 4 GHz clock and runs in 6s, find the speedup.
15. Why is **clock rate** alone not enough to measure CPU performance?
16. What are **benchmark programs**? Give examples.

**🔹 Lecture 3: System Interconnections & Buses**

1. Explain the need for **system interconnection** in a computer.
2. Differentiate between **I/O module**, **CPU**, and **Memory** connections.
3. Define a **Bus**. What are the three types of buses?
   * Data bus
   * Address bus
   * Control bus
4. Why does bus width matter for performance?
5. Explain the problems of a **single bus system**.
6. Describe **Multiple bus hierarchies** with diagrams.
7. Explain **Bus arbitration**. Differentiate between centralized and distributed arbitration.
8. What is **Bus Mastering**?
9. Explain **Direct Memory Access (DMA)**. Why is it useful?
10. Differentiate between **synchronous** and **asynchronous buses**.
11. Draw and explain the **timing diagram** for synchronous bus transfers.
12. Explain how data transfer occurs in an **asynchronous bus**.
13. Advantages & disadvantages of asynchronous buses.
14. Write a note on the **PCI bus** and its arbitration.
15. What is the role of **IRDY#** and **TRDY#** signals in PCI?

**🔹 Lecture 4: Arithmetic & ALU**

1. Define the **Arithmetic Logic Unit (ALU)**.
2. Explain **signed number representation** with examples.
   * Sign-magnitude
   * 1’s complement
   * 2’s complement
3. Show with an example how **overflow** occurs in addition.
4. State the **overflow rule** in binary addition.
5. Explain the **hardware block diagram** for binary addition/subtraction.
6. Explain the rule for subtraction using **2’s complement**.
7. Perform the multiplication of two **unsigned binary numbers** step by step.
8. What is **Booth’s Algorithm**? Perform multiplication using Booth’s algorithm with an example.
9. Explain hardware implementation of **binary multiplication**.
10. Differentiate between multiplication of signed and unsigned integers.
11. Explain the **binary division algorithm** with an example.
12. How is signed division handled in computers? What is the rule for signs of quotient and remainder?
13. Explain with examples why remainder must have the same sign as dividend.

**🔹 Lecture 5: Floating Point Arithmetic & IEEE 754**

1. What is the difference between **fixed-point** and **floating-point** representation?
2. Explain the structure of **IEEE 754 single-precision floating point format**.
3. Define the fields: **Sign bit, Exponent, Fraction**.
4. Why is exponent stored in **biased form**? What is the bias for single precision?
5. Represent the decimal number **–0.8125** in IEEE 754 format.
6. What is the largest normalized float in single precision? Show calculation.
7. What is the smallest normalized float in single precision? Show calculation.
8. What are **special values** in IEEE 754?
   * Zero
   * Infinity (+∞, –∞)
   * NaN (Not a Number)
9. Convert the given binary floating-point number into decimal:

1 01111100 01000000000000000000000

1. Convert decimal 10.375 into IEEE 754 binary.
2. Explain the process of **floating point addition** with an example.
3. Why must floating point results be **normalized**?
4. What is **rounding**? Explain **round-to-nearest-even** rule.
5. Explain floating point subtraction with normalization and rounding.
6. Explain floating point multiplication with an example.
7. What is **overflow** and **underflow** in floating point representation?
8. Why do we need **guard, round, and sticky bits**?