PES UNIVERSITY, Bangalore

Established under Karnataka Act No. 16 of 2013)

Department of Computer Science & Engineering

UE24CS251A: DIGITAL DESIGN AND COMPUTER ORGANIZATION

Unit 3:

1. **Explain the basic functions of a computer system.**
2. **Explain the role of the Memory Unit in a computer system. Differentiate between Primary and Secondary Memory.**
3. **Explain the role of the Control Unit in the execution of a program.**
4. **A research lab needs to perform complex weather simulations that involve trillions of floating-point calculations.** Which type of computer should be used? Justify your answer with respect to **processing speed** and **memory hierarchy**.
5. **Explain the basic operational steps of instruction execution in a computer system.**
6. **What are the functions of registers inside a CPU? Describe different types of registers.**
7. **Differentiate between single-bus and two-bus architectures.**
8. **What is a bus in computer organization? Why are buffer registers needed in bus structures?**
9. A hospital uses a **real-time patient monitoring system**. The system continuously measures patient vitals (heart rate, blood pressure, oxygen levels) and stores them in memory. However, if a patient's oxygen level falls below a threshold, the system must **immediately alert doctors** through an alarm, even if the CPU is currently processing other tasks.

   Question:

   Explain how the **interrupt mechanism** helps in this situation. Which registers are affected during this process?
10. **Scenario:**

    An online banking server handles thousands of transactions per second. Each request involves **reading account details from disk, performing arithmetic (balance updates), and writing results back**. The system uses a **single-bus structure**, but performance issues arise as the number of requests increases. **Question:** Why is the **single-bus structure** becoming a bottleneck in this scenario? Suggest how a **two-bus architecture** could improve performance.

11. Justify why computers do not use sign-magnitude or one's complement for arithmetic even though they are conceptually simpler.
12. Prove that the two's complement of a number is equivalent to subtracting it from 2n. Illustrate with examples.
13. A computer uses 8-bit words. Show all the steps to add the decimal numbers 93 and -46 using two's complement representation. Indicate whether overflow occurs.
14. Discuss how overflow can be detected by comparing the signs of operands and the result in two's complement addition.
15. Compare binary number representation with character codes. Why is binary representation natural for computers, whereas ASCII is needed for human interaction?
16. Explain Big-Endian and Little-Endian assignments with neat diagrams.
17. Explain how numbers, characters, and instructions are stored in memory.
18. A computer system uses a **32-bit word length** with a **byte-addressable memory**. The system supports both **Big-Endian** and **Little-Endian** formats.

The system has **20 address lines**. Calculate the total size of the memory in bytes and words.

If an integer `0x12345678` is stored at starting address `1000`, show how the bytes will be stored in **Big-Endian** and **Little-Endian** format.

Demonstrate how **word alignment** affects storage by showing the addresses of the first four words.

If a character string `"GATE"` is stored beginning at address `2000`, show how it is represented in memory.

Briefly explain why byte-addressable memory and endianness conventions are important for **system compatibility**.

19. A program needs to compute
    S=P+Q
    where **P, Q, and S** are memory locations. Write the sequence of assembly-level instructions using **(i) three-address**, **(ii) two-address**, and **(iii) one-address** instruction formats.

20. Explain the role of **condition codes** by writing the steps to execute the instruction:
    `ADD LOC, R0`
    Assume the instruction is stored at memory location **INSTR** and PC initially points to **INSTR**.
21. A stack-based processor supports **zero-address instructions**. Write the sequence of stack operations to compute:
    $Z=(A+B)\times(C-D)$

22. Explain Indexed Addressing Mode. Write an assembly program to add marks of 5 students stored in array LIST, and store result in TOTAL.
23. (a) Explain with an example how Auto-Increment and Auto-Decrement modes simplify stack/array operations.
    (b) Write a program using Auto-Decrement mode to push values A, B, C into a stack.

24. Write ALL code to swap two numbers
25. Explain the steps involved in the **assembly and execution of an assembly language program**. In your answer, clearly describe the roles of the assembler, loader, and directives such as `ORIGIN`, `RESERVE`, `RETURN`, and `END`. Give a short example program and trace how it is assembled and loaded into memory.
26. A university has an **electronic notice board** controlled by a microprocessor. Messages are typed using a keyboard and displayed character by character on the board. The interface uses the following registers and flags:

    DATAIN: 8-bit buffer register of the keyboard

    DATAOUT: 8-bit buffer register of the display

    SIN: Input status flag (set to 1 when a key is pressed, reset after read)

SOUT: Output status flag (set to 1 when the display is ready for a new character, reset after write)

Write an **assembly-language program** (using mnemonics like MOVE, ADD, BGTZ, BRANCH) to read a line of characters from the keyboard and display them on the board using **program-controlled I/O**.

Explain how synchronization between processor and devices is ensured using SIN and SOUT flags.

Discuss the disadvantage of this method compared to **interrupt-driven I/O**.

27. system designer is evaluating two I/O schemes: **polling** and **interrupt-driven I/O**.
    (a) Compare their CPU utilization when servicing a keyboard that generates 10 characters per second, assuming the CPU executes 10^6 instructions/sec.
    (b) Explain why interrupt-driven I/O becomes essential in modern multiprogrammed systems.
28. Suppose two devices generate interrupts **simultaneously** in a vectored interrupt system.
    (a) Explain how the system ensures only one device places its interrupt vector on the bus.
    (b) Why is the **interrupt acknowledge (INTA)** signal critical in such a system?
    (c) Propose what could go wrong if both devices placed their vector simultaneously.
29. Explain the working of a DMA controller. How does it improve system performance compared to programmed I/O and interrupt-driven I/O?
30. What is bus arbitration? Explain centralized and distributed arbitration methods with examples.
31. A CPU initiates a read operation on a **PCI bus** to fetch four 32-bit words from an I/O device.
    During the transfer, the **target device** is temporarily not ready to supply data for the 3rd word. **Explain what happens on the bus during this condition and how PCI ensures proper data transfer.**
32. Assume a **processor requests data blocks** from a non-contiguous disk region using a **SCSI controller**.
    Describe how the initiator and target coordinate this operation to maintain bus efficiency.
33. A USB keyboard and webcam are connected through a hub to a host computer.
    Explain how USB handles **simultaneous low-speed (keyboard)** and **isochronous (webcam)** data transfers without collisions.
34. Explain the basic difference between Static RAM (SRAM) and Dynamic RAM (DRAM). Describe their internal structures, operation, and typical applications.
35. Explain the concept of memory organization using multiple memory chips. How can a 2M × 32-bit memory be built using 512K × 8 SRAM chips?