CMPE 101- COMPUTER ENGINEERING AS A DISCIPLINE

Module 6-7-

Software Engineering in Computer Engineering

Lesson Title: Introduction to Software Development and System Design in Computer Engineering

Duration: 1 hour

Learning Objectives:

By the end of this lesson, students will be able to:

1. Understand the fundamentals of software development in Computer Engineering.
2. Recognize the role of programming in computer systems design.
3. Explain the functions of operating systems.
4. Differentiate between software and hardware in system design.

I. Introduction to Software Development in Computer Engineering (15 minutes)

1. **Definition of Software Development**:
    - Software development involves creating, designing, testing, and maintaining software programs or applications.
    - In **Computer Engineering**, software development is crucial for building systems, applications, and embedded systems.
2. **Key Phases of Software Development**:
    - **Requirement Analysis**: Understanding what the software must do.
    - **Design**: Structuring how the software will work.
    - **Implementation (Coding)**: Writing the actual code.
    - **Testing**: Ensuring the software functions as expected.
    - **Deployment**: Making the software available for use.
    - **Maintenance**: Updating and fixing the software over time.
3. **Tools Used in Software Development**:
    - **IDEs (Integrated Development Environments)**: Tools like Visual Studio, Eclipse.
    - **Version Control Systems**: Git for tracking changes in code.
    - **Programming Languages**: C, C++, Python, Java.

II. Role of Programming in Computer Systems Design (10 minutes)

1. **Programming as the Backbone of Computer Systems**:
    - Programming is essential for **defining the behavior** of a system and automating tasks.
    - **Embedded Systems**: Microcontrollers rely on software programming to control operations in devices (e.g., sensors, actuators).
2. **Levels of Programming**:
    - **Low-Level Programming**: Close to hardware (e.g., Assembly, C).
    - **High-Level Programming**: More abstract, dealing with logic and operations (e.g., Python, Java).
3. **Application of Programming in System Design**:
    - **Operating Systems Development**: Operating systems like Linux and Windows are built using low-level programming.
    - **System Performance**: Efficient code can lead to faster and more reliable systems.

III. Overview of Operating Systems and Their Functions (15 minutes)

1. **Definition of Operating Systems (OS)**:
    - An **Operating System** is system software that manages hardware and software resources and provides common services for computer programs.
2. **Key Functions of Operating Systems**:
    - **Process Management**: Manages the execution of processes.
    - **Memory Management**: Allocates and deallocates memory to programs.
    - **File System Management**: Organizes and stores files on storage devices.
    - **Device Management**: Controls peripherals like printers, keyboards, and monitors.
    - **Security and Access Control**: Ensures data privacy and restricts unauthorized access.
3. **Types of Operating Systems**:
    - **Real-Time OS (RTOS)**: Used in embedded systems where timely execution is critical (e.g., automotive systems).
    - **General-Purpose OS**: For personal computers (e.g., Windows, macOS, Linux).

IV. Software vs. Hardware in System Design (15 minutes)

1. **Definition of Hardware**:

- **Hardware** refers to the physical components of a computer system, such as the processor (CPU), memory (RAM), storage (HDD/SSD), and peripherals.

2. **Definition of Software**:
   - **Software** is a set of instructions or code that tells the hardware how to perform tasks. It includes operating systems, applications, and embedded programs.
3. **Comparison:**
   - **Hardware** is tangible, while **software** is intangible.
   - **Hardware** performs mechanical and electronic tasks, while **software** executes logical operations.
   - **Software** can be updated easily without changing the physical system, whereas **hardware** may need to be replaced for upgrades.
4. **Integration in System Design**:
   - Computer systems require both **hardware and software** to function effectively.
   - **Hardware without software** is non-functional, while **software without hardware** has no medium to execute its instructions.
   - **Embedded Systems**: Often, software is embedded directly into the hardware (firmware), making them highly interdependent.

V. Summary and Q&A (5 minutes)

- **Recap Key Points**:
  - Introduction to software development and its importance in Computer Engineering.
  - Role of programming in designing and controlling computer systems.
  - Operating systems and their critical functions.
  - The distinction between hardware and software in system design.
- **Questions**: Encourage students to ask questions and provide clarifications.

References (2019-2024):

1. **Stallings, W. (2020).** *Operating Systems: Internals and Design Principles* (9th ed.). Pearson.
2. **Pressman, R. S., & Maxim, B. R. (2020).** *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill Education.
3. **Tanenbaum, A. S., & Bos, H. (2021).** *Modern Operating Systems* (4th ed.). Pearson.
4. **Sommerville, I. (2020).** *Software Engineering* (11th ed.). Pearson.
5. **Hennessy, J. L., & Patterson, D. A. (2019).** *Computer Architecture: A Quantitative Approach* (6th ed.). Morgan Kaufmann.

This lesson plan provides students with a strong foundational understanding of software development, the role of programming, the importance of operating systems, and the distinction between software and hardware, all within a 1-hour session.