Program:	**Applied Electronics and Instrumentation Engineering**
Course Title:	**Microprocessors and Microcontroller**
Course Code:	PC-EI-402

The reader is expected to know the following concepts:

- ✓ Number systems (binary, octal and hexadecimal)
- ✓ Boolean algebra, logic gates, flip-flops and registers.
- ✓ Concepts in combinational and sequential logic.

At the end of the course, a student will be able to:

Course Outcome:
PC-EI-402.1: **Define** the architecture of Microprocessors and Microcontroller (8051).
PC-EI-402.2: **Explain** the importance and function of different modules of Microprocessor and Microcontroller.
PC-EI-402.3: **Apply** the fundamentals of assembly level programming of Microprocessor and Microcontroller that perform specific tasks.
PC-EI-402.4: Students will be able to **analyze** and **evaluate** different interfacing techniques used to connect microprocessors and microcontrollers with peripheral devices, identifying the most suitable methods for specific applications.

Course Overview

This course introduces the fundamental concepts of microprocessors and microcontrollers, focusing on the 8085 microprocessor and 8051 microcontroller. Students will learn about their architecture, memory interfacing, stack and interrupts, as well as peripheral interfacing. The course aims to build a solid foundation in assembly language programming and interfacing techniques.

Unit Plan and Unit Outcomes Details
Unit 1: 8085 Architecture
Unit Rationale: This unit introduces students to the fundamental concepts of microprocessor-based systems with a specific focus on the Intel 8085 microprocessor. Understanding the internal architecture, pin configuration, instruction set, and programming of the 8085 is crucial for designing and analyzing embedded systems. Students will also gain hands-on experience in assembly language programming and learn how different addressing modes influence instruction execution.

By mastering these topics, students will develop a strong foundation in microprocessor technology, which is essential for advanced topics such as microcontrollers, interfacing, and embedded system design.

Unit Outcomes:

- Understand the architecture and operation of the 8085 microprocessor.
- Identify the instruction set and programming model of 8085.
- Develop basic assembly language programs.

Session Plan:

Introduction to Microprocessor-based systems – Definition, applications, and evolution of microprocessors.

Overview of 8085, Internal Architecture –Explain the internal structure, registers of Components, buses, and operation of 8085..

Pin Diagram Description – Understanding functional blocks and signal classification.

Instruction Set – Overview of different types of instructions and their operations.

Instruction Set (Continued) – Addressing modes and instruction classification.

Assembly Language Programming Basics – Structure of assembly programs and programming techniques.

Assembly Language Programming (Continued) – Writing simple programs in 8085.

Addressing Modes – Explanation with examples and their applications.

Tutorial – Discussion of programming exercises and problem-solving.

Tutorial – Revision and practice session.

Unit Outcomes: Introduction to 8085 Architecture

Topic	**Learning Outcomes**	**Bloom's Taxonomy Level**
Introduction to Microprocessor-Based Systems	Define a microprocessor and explain its significance in computing.	Remembering (L1)
	Describe the basic structure and working of a microprocessor-based system.	Understanding (L2)
Overview of 8085, Internal Architecture	Recall the features and specifications of the 8085 microprocessor.	Remembering (L1)

	Explain the internal architecture of 8085 and describe its functional units.	Understanding (L2)
Pin Diagram Description	Identify and list the pins and signals of 8085 microprocessor.	Remembering (L1)
	Explain the purpose of different control, status, and power signals.	Understanding (L2)
	Demonstrate how various pins are used in system design.	Applying (L3)
Instruction Set of 8085	List and classify different types of instructions in 8085.	Remembering (L1)
	Explain the function of data transfer, arithmetic, logical, branching, and control instructions.	Understanding (L2)
	Write simple assembly programs using different instruction types.	Applying (L3)
Assembly Language Programming	Explain the syntax and structure of an assembly language program.	Understanding (L2)
	Develop basic programs using assembly language to perform arithmetic and logical operations.	Applying (L3)
	Debug and optimize an assembly language program.	Analyzing (L4)
Addressing Modes	Define different types of addressing modes used in 8085.	Remembering (L1)
	Explain how different addressing modes affect instruction execution.	Understanding (L2)
	Use the appropriate addressing modes in assembly language programs.	Applying (L3)
Tutorial (Hands-on Practice & Problem Solving)	Solve problems based on 8085 instructions and addressing modes.	Applying (L3)
	Debug given assembly language programs and suggest improvements.	Analyzing (L4)
	Compare the efficiency of different instructions and optimization techniques in assembly programming.	Evaluating (L5)

Unit 2: Memory Interfacing

Unit Rationale:

This unit focuses on the fundamental principles of memory and input/output (I/O) interfacing in microprocessors and microcontrollers. Understanding memory structures, their interfacing techniques, and the process of communicating with I/O devices is critical for designing efficient embedded systems. Additionally, the unit covers instruction cycles, machine cycles, and timing diagrams, which are essential for comprehending microprocessor operations at the hardware level. Mastery of these concepts enables students to design, analyze, and troubleshoot microprocessor-based systems effectively.

Unit Outcomes:

- Understand the memory structure and types of memory used in microprocessors.
- Explain how memory is interfaced with the microprocessor.
- Interpret timing diagrams and memory access mechanisms.

Session Plan:

Basic Structure of Memory – Types of memory (RAM, ROM, EPROM) and their functions.

Memory Interfacing – Concept of address decoding and memory mapping.

Memory Interfacing (Continued) – Address and control signal generation.

Interfacing I/O Devices – Difference between memory-mapped and I/O-mapped I/O.

Interfacing I/O Devices (Continued) – Practical interfacing examples.

Instruction Cycle, Machine Cycle, Timing Diagrams – Explanation and interpretation.

Instruction Cycle, Machine Cycle, Timing Diagrams (Continued) – Examples with case studies.

Tutorial – Hands-on problem-solving session.

Tutorial – Doubt clearing and revision.

Unit Outcomes (Based on Bloom’s Taxonomy)

Topic	**Learning Outcome**	**Bloom’s Taxonomy Level**
Basic Structure of Memory	Explain the fundamental structure and organization of memory in microprocessor-based systems.	Understand (Level 2)
Interfacing of Memory	Demonstrate the interfacing of various memory components with a microprocessor.	Apply (Level 3)
	Compare different memory interfacing techniques and identify their applications.	Analyze (Level 4)
	Assess the performance and compatibility of different memory interfacing methods.	Evaluate (Level 5)
Interfacing I/O Devices	Implement interfacing of I/O devices with microprocessors and microcontrollers.	Apply (Level 3)

	Differentiate between memory-mapped I/O and peripheral-mapped I/O techniques.	Analyze (Level 4)
Instruction Cycle, Machine Cycle, Timing Diagrams	Describe the instruction cycle, machine cycle, and associated timing diagrams.	Understand (Level 2)
	Interpret timing diagrams to determine microprocessor operations and execution sequences.	Apply (Level 3)

Unit 3: Stack , Interrupts & 8255 PPI

Unit Rationale:

This unit introduces the concepts of stack operations, subroutines, and their significance in microprocessor and microcontroller programming. It covers the role of interrupts, the interrupt vector table, and the implementation of Interrupt Service Routines (ISRs). Additionally, the unit explores the 8255 Programmable Peripheral Interface (PPI) for I/O device interfacing, enhancing students' ability to design real-time embedded systems.

Unit Outcomes:

Understand the concept of stack and its handling in microprocessors.

Explain different types of interrupts and their handling mechanisms.

Develop assembly programs utilizing interrupts.

Session Plan:

Stack and Stack Handling – Concept of stack, PUSH/POP operations.

Stack and Stack Handling (Continued) – Call and subroutine execution.

Counter and Time Delay Generation – Use of counters for timing applications.

Counter and Time Delay Generation (Continued) – Practical implementation.

Interrupts: Introduction – Types, priority, and vector table.

Design of Programs using Interrupts – Writing interrupt-driven assembly programs.

8255 PPI (Programmable Peripheral Interface) – Concept and applications.

8255 PPI (Continued) – Interfacing and programming examples.

Unit Outcomes (Based on Bloom's Taxonomy)

Topic	Learning Outcome (Student will be able to...)	Bloom's Level
Stack and Stack Handling, Call and Subroutine	Explain stack operations and the significance of subroutines in microprocessor programming.	Understand
	Implement stack-based operations using assembly language.	Apply

	Compare the efficiency of stack-based and non-stack-based function calls.	Analyze
Counter and Time Delay Generation	Describe the working of counters and time delay techniques in microprocessors.	Understand
	Develop delay generation programs using loops and counters.	Apply
	Optimize delay routines for efficient timing in microcontroller applications.	Evaluate
Interrupts: Introduction, Interrupt Vector Table, Interrupt Service Routine	Explain the need for interrupts and their handling in a microprocessor system.	Understand
	Develop programs incorporating interrupts and ISRs in microcontrollers.	Apply
	Differentiate between hardware and software interrupts with practical examples.	Analyze
Design of Programs using Interrupts	Implement real-time control systems using interrupts.	Apply
	Design and test interrupt-driven programs for specific applications.	Create
8255 Programmable Peripheral Interface (PPI)	Explain the architecture and working of the 8255 PPI.	Understand
	Configure 8255 in different modes for interfacing peripherals.	Apply
	Design and develop microcontroller-based applications using the 8255 PPI.	Create

This structured approach ensures a comprehensive understanding of stack operations, interrupts, and peripheral interfacing, aligned with Bloom's Taxonomy to promote higher-order thinking.

Unit 4: Microcontroller

Unit Rationale:

This unit introduces the architecture and instruction set of the Intel 8051 microcontroller. Understanding its internal structure, registers, memory organization, and instruction set is crucial for programming and interfacing applications. This knowledge enables students to design embedded systems and develop efficient microcontroller-based solutions.

Unit Outcomes:

- Understand the architecture and components of the 8051 microcontroller.
- Explain the function of registers, memory, and ports.
- Write assembly language programs for microcontroller applications.

Session Plan:

1. **Introduction to Microcontrollers** – Microcontroller vs. microprocessor.
2. **8051 Architecture Overview** – Block diagram and functional units.
3. **8051 Registers, Oscillators, Ports, and Memory** – Internal structure and working.

4. **Timers/Counters and Special Function Registers** – Role and functionality.
5. **Addressing Modes in Microcontroller** – Explanation with examples.
6. **Instructions related to Data Transfer and Manipulation** – Execution with assembly programs.
7. **Arithmetic, Logical, and Branch Operations** – Programming concepts.
8. **Programming Exercises** – Hands-on coding session.

Unit Outcomes (Mapped with Bloom's Taxonomy)

Bloom's Taxonomy Level	**Learning Outcomes**
Remembering (Knowledge)	- Define the architecture and components of the Intel 8051 microcontroller. - List different types of registers, memory, and special function registers in the 8051. - Recall various instruction formats and addressing modes.
Understanding (Comprehension)	- Explain the working of different components such as ports, oscillators, timers, and counters in the 8051 microcontroller. - Describe the role of different registers in executing programs. - Summarize the functionality of various instruction categories.
Applying (Application)	- Demonstrate simple programs using 8051 assembly language for data transfer, arithmetic, and logic operations. - Implement programs using different addressing modes. - Use timers and counters for delay generation and event counting.
Analyzing (Analysis)	- Compare different addressing modes and their suitability for specific applications. - Analyze the impact of different instructions on registers and memory. - Differentiate between various instruction types based on their operation.
Evaluating (Evaluation)	- Justify the selection of a specific instruction set for efficient programming. - Assess the efficiency of different programming approaches in microcontroller-based applications.
Creating (Synthesis)	- Design and develop embedded applications using the 8051 microcontroller. - Write optimized programs using different instructions and addressing modes.

	- Integrate various components (timers, counters, ports) in an embedded system project.

Unit 5: Microcontroller Peripherals

Unit Rationale:

Microcontrollers play a critical role in embedded systems, where interaction with external peripherals is essential. Understanding timers, counters, serial communication, interrupts, and interfacing with I/O devices is fundamental for designing real-time applications. This unit provides students with theoretical and practical knowledge of how microcontrollers interact with peripheral components, enabling efficient programming and hardware integration.

Unit Outcomes:

- Understand timers/counters and their role in microcontrollers.
- Explain the concepts of serial communication and interrupts.
- Interface microcontrollers with peripheral devices.

Session Plan:

1. **Introduction to Timer/Counter** – Working principles and applications.
2. **Serial Communication and Interrupts** – UART, SPI, and I2C protocols.
3. **Harvard vs. Von Neumann Architecture** – Differences and advantages.
4. **Serial Communication and Interrupt Programming** – Implementation and examples.
5. **Interfacing with Peripheral I/O Devices** – ADC, DAC, and display interfacing.
6. **Interfacing with Peripheral I/O Devices (Continued)** – Case studies and hands-on practice.
7. **Tutorial and Practice Session** – Debugging and project discussion.

Unit Outcomes (Based on Bloom's Taxonomy)

Topic	**Learning Outcome**	Bloom's Taxonomy Level
Introduction to the Timer/Counter	Explain the operation and significance of timers and counters in microcontrollers.	Understand (Level 2)
	Implement timer and counter programming using assembly language.	Apply (Level 3)
Serial Communication and Interrupts: Operations, Special Function Registers, and Programming	Describe the principles of serial communication and the role of interrupts.	Understand (Level 2)
	Analyze the function of special registers used in serial communication and interrupts.	Analyze (Level 4)

	Develop serial communication and interrupt-based programs for microcontrollers.	Apply (Level 3)
Harvard Architecture, Von Neumann Architecture	Differentiate between Harvard and Von Neumann architectures.	Analyze (Level 4)
	Justify the use of specific architecture for different applications.	Evaluate (Level 5)
Interfacing with Peripheral Input / Output Devices: ADC, DAC, Display	Describe the working principles of ADC, DAC, and display interfaces.	Understand (Level 2)
	Design and implement programs for ADC, DAC, and display interfacing.	Apply (Level 3)
	Troubleshoot interfacing issues and optimize performance.	Evaluate (Level 5)

Conclusion

This course provides a comprehensive understanding of microprocessors and microcontrollers, covering theoretical concepts and practical applications. By the end of the course, students will be proficient in programming and interfacing microprocessors/microcontrollers with peripheral devices, preparing them for real-world embedded system applications.

Bloom Taxonomy Table

Cognitive Processes	**Knowledge Categories**			
	Factual	**Conceptual**	**Procedural**	**Metacognitive**
Remember	**CO 1**			
Understand		**CO 2**		
Apply			**CO 3**	
Analyze		**CO 4**		
Evaluate				
Create				

CO-PO Mapping:

	PO 1	**PO 2**	**PO 3**	**PO 4**	**PO 5**	**PO 6**	**PO 7**	**PO 8**	**PO 9**	**PO 10**	**PO1 11**	**PO1 12**	**PS O1**	**PS O2**	**PS O3**
CO 1:	3	3	2	2	1	-	-	-	-	-	-	2	3	3	3
CO 2:	3	2	2	2	1	-	-	-	-	-	-	2	2	2	3
CO 3:	3	3	3	3	1	1	1	1	1	1	1	2	3	3	3
CO 4:	3	2	3	3	1	1	1	1	1	1	1	2	3	2	2

Justification for PO Attainment:

PO1 (Engineering Knowledge)

- The subject involves **fundamental concepts of microprocessors and microcontrollers**, including architecture, assembly programming, and interfacing techniques.
- As students gain a strong foundation in **electronics and instrumentation**, the **attainment level of 3** is appropriate for all COs.

PO2 (Problem Analysis)

- COs related to understanding the **architecture, modules, and interfacing techniques** help students analyze and solve problems in **embedded system design**.
- The ability to analyze **complex engineering problems** aligns well with microprocessor programming and interfacing, justifying a **level 2 or 3** attainment.

PO3 (Design/Development of Solutions)

- **Microprocessor programming** requires students to design and develop optimized **assembly-level solutions** for given problems.
- The interfacing techniques discussed in CO4 also contribute to developing hardware and software solutions, leading to **moderate to high attainment values (2 or 3).**

PO4 (Conducting Investigations)

- The subject includes **assembly language programming and peripheral interfacing**, requiring students to **experiment, analyze, and interpret data**.
- The attainment levels of **2 or 3** are justified as students gain exposure to practical debugging and testing of microprocessor-based systems.

PO5 (Modern Tool Usage)

- The use of **assemblers, debuggers, and interfacing tools** is covered, but the scope of modern tools is limited to **basic programming and interfacing**.
- A **low attainment level of 1** is reasonable since advanced simulation tools or CAD tools are not extensively used.

PO6 (Engineer and Society)

- The subject is **highly technical** with **minimal focus on societal, legal, health, and safety issues**.
- Since microprocessor-based systems **indirectly impact** real-world applications, a **minimal attainment level of 1 for CO3 and CO4** is reasonable.

PO7 (Environment and Sustainability - Low Attainment)

- The subject does not directly address **environmental or sustainability concerns**.
- However, embedded systems play a role in energy-efficient designs, so a **low attainment level of 1** is justified for **CO3 and CO4**.

PO8 (Ethics)

- Ethics is **not a direct focus area** in this subject.
- While professional ethics in **hardware/software design** is crucial, it is **not explicitly covered**, justifying a **low attainment level of 1** for CO3 and CO4.

PO9 (Individual and Team Work)

- The subject primarily involves **individual learning and programming exercises**.
- While team collaboration may occur in **project-based assignments**, it is not a core aspect of the course, leading to a **low attainment level of 1** for CO3 and CO4.

PO10 (Communication)

- The course mainly involves **technical documentation, coding, and debugging**.
- Since formal communication skills (verbal/non-verbal) are not a primary focus, a **low attainment level of 1** is justified for CO3 and CO4.

PO11 (Project Management and Finance)

- The subject does not explicitly cover **project management or financial aspects**.
- Some exposure may exist in implementing **cost-effective interfacing solutions**, leading to a **low attainment level of 1** for CO3 and CO4.

PO12 (Life-long Learning)

- The evolving nature of microprocessor technologies requires **continuous learning**.
- Students are encouraged to **explore advanced architectures and applications**, justifying a **moderate attainment level of 2** for all COs.

Justification for PSO Attainment:

PSO1 (Strong Foundation in STEM)

- The subject builds a **strong foundation in electronics, programming, and interfacing**.
- A **high attainment level of 3** is justified as the course content is core to electronics and instrumentation.

PSO2 (Instrumentation Engineering Features)

- The subject covers **real-time data processing, sensor interfacing, and microcontroller applications**, aligning well with **industrial automation**.
- Attainment levels of **2 or 3** are appropriate, depending on the CO's focus on implementation.

PSO3 (Interdisciplinary Applications)

- Microprocessors and microcontrollers play a role in **biomedical, automation, renewable energy, and Industry 4.0 applications**.
- The skills gained in this subject support **entrepreneurship and research**, justifying a **high attainment level of 3**.

Conclusion

The given attainment values for **COs with POs and PSOs** are well-justified based on the technical nature of the subject. The **higher values (2 or 3) for technical POs (PO1-PO5) and PSOs** reflect the strong relevance of microprocessors in engineering, while the **lower values (0 or 1) for non-technical POs (PO6-PO11)** acknowledge their limited integration in the course content.