JSPM Univeristy,Pune
Introduction to Embedded Systems Lab Manual (230GETB41)
Viva Question With Answer For Practical Exam
Nov \Dec-2025

Instruction For Student

Read all Practical Carefully.
Below Question is for Reference for Viva Question.
During Practical Perform Follow procedure

1-Aim
2-SOFTWARE REQUIRED:
3-ALGORITHM
4-Example if Required
5-Source CODE With Comment
6-RESULT

Keil Software – Interview Questions & Answers

Basic Questions

Q1. What is Keil software used for?

Keil is an Integrated Development Environment (IDE) used for developing, compiling, debugging, and simulating embedded programs, especially for 8051, ARM, and Cortex-M microcontrollers.

Q2. What are the main components of Keil?

1. µVision IDE – for editing and debugging code.
2. C51 Compiler – compiles C code for 8051.
3. A51 Assembler – converts assembly code to machine code.
4. Linker/Locator – combines object files into a final HEX file.

Q3. What file formats are generated by Keil?

.ASM (source code), .OBJ (object file), .HEX (final output for microcontroller), .LST (listing file), and .MAP (memory map).

Q4. What is a HEX file and why is it important?

The HEX file contains the machine code that is loaded into the microcontroller's memory for execution. It's the final output of the build process.

⚙ **8051-Specific Questions**

Q5. What is the size of the 8051 program memory?

The standard 8051 has 4 KB of ROM (program memory) and 128 bytes of RAM (data memory).

Q6. How many ports are there in 8051?

There are four 8-bit parallel ports: P0, P1, P2, and P3.

Q7. What is the function of the accumulator (A register)?

The accumulator is used in most arithmetic and logic operations and as one operand in multiplication/division (MUL AB, DIV AB).

Q8. What are SFRs in 8051?

SFRs (Special Function Registers) are memory-mapped registers (from address 80H–FFH) used to control timers, serial communication, ports, etc.

Q9. Explain the function of DPTR.

DPTR (Data Pointer) is a 16-bit register used to access external memory or large data blocks. It consists of DPH (high byte) and DPL (low byte).

Q10. What is the difference between MOV and MOVX instructions?

MOV is used to move data within internal RAM/SFRs, while MOVX is used to move data to/from external memory.

Programming & Debugging Questions

Q11. What is the difference between Simulator and Emulator in Keil?

Simulator runs code on the PC (software simulation).
Emulator runs code on the actual hardware (real-time execution).

Q12. What steps are involved in creating a project in Keil?

Q11. What is the difference between Simulator and Emulator in Keil?

1. Open Keil → Create new project
2. Select target microcontroller
3. Add new source file
4. Write code (C or ASM)
5. Build project
6. Debug or simulate
7. Generate HEX file.

Q13. What are Interrupts in 8051?

Interrupts are events that temporarily pause the main program to execute a specific Interrupt Service Routine (ISR). 8051 supports 5 interrupts (2 external, 2 timers, 1 serial).

Q14. How many Timers are there in 8051?

There are two 16-bit timers: Timer 0 and Timer 1, which can operate in various modes.

Q15. What are common errors during compilation in Keil?

- Missing END in assembly code
- Undefined labels
- Syntax errors
- Incorrect register usage
- Stack overflow in simulation.

Practical / Application Questions

Q16. How do you simulate I/O port behavior in Keil?

In the Keil simulator, open Peripherals → I/O Ports, where you can observe or toggle pin values during simulation.

Q17. How can you delay an operation in 8051?

By using software loops (NOPs or decrement counters) or hardware timers configured with a known clock frequency.

Q18. What is the clock frequency of 8051?

Typically 12 MHz, but it can vary depending on the version of the microcontroller.

Q19. How does Keil help in debugging?

Q16. How do you simulate I/O port behavior in Keil?

It provides step execution, breakpoints, register view, memory watch, and I/O simulation to test and debug programs.

Q20. What is the use of the directive ORG in assembly language?

The ORG directive specifies the starting address for program or data storage in memory.

Experiment Based Question

Write an assembly language program to find the GCD and LCM of a given two bytes in a memory locations.

Question	**Answer**
1. What is GCD?	The Greatest Common Divisor (GCD) of two numbers is the largest number that divides both without leaving a remainder.
2. What is LCM?	The Least Common Multiple (LCM) of two numbers is the smallest number that is a multiple of both numbers.
3. What is the relation between GCD and LCM?	GCD × LCM = (Number1 × Number2).
4. Which algorithm is commonly used to find GCD?	**Euclid's algorithm**, which uses repeated subtraction or modulo operation.
5. What are the memory addresses used for in this program?	To store the two input numbers and the results (GCD and LCM).

Question	Answer
6. Why do we use registers B and C in this program?	Registers B and C are used to store and manipulate the two numbers during the GCD calculation.
7. Can we find GCD for negative numbers using this program?	No, this program is designed for **unsigned 8-bit positive integers** only.
8. What is the function of the CMP instruction?	It compares the accumulator with another register and sets flags based on the result.
9. What is the significance of the STA instruction?	It stores the contents of the accumulator into a specified memory address.
10. What is the difference between JC and JZ?	JC jumps if the Carry flag is set (A < operand), while JZ jumps if the Zero flag is set (A = operand).

```
ORG 0000H        ; Program starts at memory address 0000H
MOV A, 50H       ; Load first number (from memory 50H) into accumulator A
MOV B, 51H       ; Load second number (from memory 51H) into register B
BACK: MOV R1, B  ; Store B temporarily in R1 (used for later reference)
DIV AB           ; Divide A by B → A = Quotient, B = Remainder
MOV A, B         ; Move remainder into A to check if it is zero

CJNE A, #00H, L1  ; Compare A with 0 → if not equal, jump to L1
L1: JZ LAST       ; If remainder = 0, jump to LAST (GCD found)
MOV A, R1         ; Move previous divisor into A (A = previous B)
SJMP BACK         ; Repeat loop with new values until remainder = 0
LAST: MOV A, R1   ; When remainder = 0, the GCD is stored in R1
MOV 52H, A        ; Store GCD into memory location 52H
; -------- Now find LCM --------
MOV A, 50H        ; Load first number into A
MOV B, 51H        ; Load second number into B
MUL AB            ; Multiply A and B → result = A × B (A = low byte, B = high byte)
```

```
MOV B, 52H      ; Load GCD from memory 52H into B
DIV AB          ; Divide (A:B) by GCD → A = LCM
MOV 53H, A      ; Store LCM result into memory 53H
END             ; End of program
```

Experiment 02:- Write a program to convert the given BCD number into its equivalent seven segment value.

Question Answer

1. What is a BCD number?	BCD (Binary Coded Decimal) represents each decimal digit separately in 4 bits. For example, 25 = 0010 0101.
2. What is a seven-segment display?	A seven-segment display is an arrangement of 7 LEDs (segments) used to display digits 0–9.
3. How many segments are used to display all digits?	Seven segments (a, b, c, d, e, f, g).
4. What is the difference between common anode and common cathode displays?	In a **common cathode**, all cathodes are connected to ground; logic '1' lights a segment. In a **common anode**, all anodes are connected to VCC; logic '0' lights a segment.
5. What instruction is used to access lookup table data?	The **MOVC A, @A+DPTR** instruction is used to read data from the lookup table in program memory.
6. What is the use of DPTR in this program?	DPTR (Data Pointer) holds the 16-bit address of the lookup table in program memory.
7. What does the SJMP $ instruction do?	It creates an infinite loop — halting the program at the last instruction.
8. Why is a lookup table used here?	Because each digit (0–9) has a unique 7-segment code; using a table simplifies conversion.
9. What is the advantage of using BCD instead of binary?	BCD makes it easy to display numeric digits directly on seven-segment displays.
10. What is the address range of program memory in 8051?	From **0000H to FFFFH** (64 KB).

3-Toggling of p1.5 with the delay of 5ms without interrupts using timer 1

Question Answer

1. What is the purpose of this program?	To toggle the output pin **P1.5** every 5 ms using **Timer 1** in **Mode 1** (without interrupts).
2. Which timer mode is used here?	**Mode 1** – 16-bit timer mode.
3. What does TMOD = #10H mean?	It configures **Timer 1** in **Mode 1 (16-bit)**, while Timer 0 remains unaffected.
4. What is the function of TR1 and TF1 bits?	TR1 (Timer Run 1) starts/stops Timer 1. TF1 (Timer Flag 1) becomes 1 when Timer 1 overflows.
5. What is the frequency of the 8051 crystal oscillator typically used?	Commonly **12 MHz**.
6. How is the delay generated using the timer?	By loading a starting value into the timer registers and allowing it to count up to overflow (FFFFH).
7. Why do we clear TF1 and TR1 after overflow?	To stop the timer and reset it for the next delay cycle.
8. What instruction toggles the bit P1.5?	CPL P1.5 (complement P1.5).
9. What happens if the clock frequency is changed?	The delay time changes — it depends on the clock frequency, so timer values must be recalculated.
10. Can the same task be done using interrupts?	Yes, by enabling Timer 1 interrupt and placing toggle logic inside the ISR, but here we use polling (no interrupts).

4-Toggling of p1.2 with the delay of 5ms with interrupts using timer 0

Question Answer

1. What is the purpose of this program?	To toggle output pin **P1.2** every 5 ms using **Timer 0 interrupt**.
2. Which timer mode is used here?	**Mode 1 (16-bit timer mode)**.

3. What is the vector address for Timer 0 interrupt?	**000BH**.
4. What instruction returns from an interrupt?	RETI (Return from Interrupt).
5. How is the delay generated?	By allowing Timer 0 to overflow after counting 5000 µs (5 ms).
6. What happens when Timer 0 overflows?	The **TF0 flag** is set, triggering the **Timer 0 ISR** automatically.
7. What is the role of the EA bit?	It enables or disables **all interrupts** globally.
8. What is the role of the ET0 bit?	It enables **Timer 0 interrupt** specifically.
9. What is the function of CPL P1.2?	It **toggles** the logic state of pin **P1.2**.
10. What is the advantage of using interrupts over polling?	Interrupts allow the CPU to perform other tasks while waiting, improving efficiency.
11. What happens if RETI is not used at the end of ISR?	The CPU will not return to the main program correctly, causing unpredictable behavior.
12. How often will P1.2 toggle?	Once every **5 ms**, meaning a full ON-OFF cycle of **10 ms** (100 Hz frequency).
13. Can the same delay be achieved using Timer 1?	Yes, but the interrupt vector and TMOD bits will change accordingly.
14. What happens if the clock frequency changes?	The delay will change; new timer reload values must be calculated.
15. Why is reloading of TH0/TL0 done inside ISR?	Because the timer resets to 0000H after overflow; reloading ensures a constant 5 ms period.

5-Count no. 0f pulses without interrupts using counter 0

Question	Answer
1. What is the purpose of this program?	To count external pulses applied to **T0 (P3.4)** pin using **Counter 0** without using interrupts.

Question	Answer
2. How is a counter different from a timer in 8051?	A **timer** counts **machine cycles** (internal clock), while a **counter** counts **external pulses** on T0 or T1 pins.
3. What is the function of the C/T bit in TMOD register?	It selects **Timer (0)** or **Counter (1)** operation mode.
4. What does TMOD = #05H mean?	It sets **Counter 0** in **Mode 1 (16-bit counter)**.
5. Which pin is used for Counter 0 input?	**P3.4** (T0).
6. What is the range of Counter 0 in Mode 1?	It can count **0 to 65535 pulses (FFFFH)**.
7. What happens when the counter overflows?	The **TF0 flag** becomes **1**.
8. How is overflow detected here?	By **polling TF0** flag using JNB TF0, HERE instruction (no interrupt).
9. Why do we clear TF0 after overflow?	To make the counter ready for the next counting cycle.
10. What happens if the TR0 bit is cleared?	Counter 0 stops counting external pulses.
11. How are the counts read in the program?	Using the **TH0** (high byte) and **TL0** (low byte) registers.
12. What is the maximum number of counts before overflow?	**65,536 (2^{16})** counts.
13. Can Counter 1 also count pulses?	Yes, by configuring **TMOD** for **Counter 1** and using **T1 (P3.5)** pin.
14. What happens if no pulses are applied to P3.4?	The count will remain unchanged — counter increments only on external pulses.
15. What is the difference between polling and interrupt counting?	In **polling**, CPU continuously checks TF0; in **interrupt**, CPU is automatically notified on overflow — more efficient.

Question: Why do we use counters in microcontrollers?
Answer: Counters are used to **measure external events or frequencies** (like pulse counting, speed measurement, RPM detection) independent of the CPU clock.

Question: What is the advantage of using counter mode instead of manually counting pulses?
Answer: The hardware counter is **fast and accurate** — it counts external transitions without software delay.

6- Count no. 0f pulses with interrupts using counter 1

Question	Answer
1. What is the purpose of this program?	To count external pulses applied to **P3.5 (T1)** using **Counter 1 with interrupt handling**.
2. What is the difference between Timer and Counter in 8051?	A **Timer** counts internal clock cycles, while a **Counter** counts **external pulses** on T0 or T1 pins.
3. What is the vector address for Timer/Counter 1 interrupt?	**001BH**.
4. What does TMOD = #50H do?	It sets Counter 1 in **Mode 1 (16-bit counter mode)**; C/T1=1 selects counter operation.
5. Which pin of 8051 is used for Counter 1 input?	**P3.5 (T1)**.
6. What instruction enables global interrupts?	SETB EA.
7. What instruction enables only Timer 1 interrupt?	SETB ET1.
8. What happens when Counter 1 overflows?	The **TF1 flag** is set, and the **interrupt service routine (ISR)** is executed.
9. Why is TF1 cleared inside the ISR?	To acknowledge the interrupt and prepare for the next counting cycle.
10. What is the function of RETI?	It marks the **end of the ISR** and returns program control to the main routine.
11. What is the maximum count value before overflow?	**65536 counts (2^{16})**.

Question	Answer
12. What happens if no external pulses are applied to T1?	The counter will not increment.
13. How can you calculate the total pulse count?	**Total = (R0 × 65536) + (TH1 × 256 + TL1)**.
14. What is the difference between using interrupts and polling here?	Interrupts free the CPU from continuously checking flags, making the system more efficient.
15. Can Counter 0 also count pulses with interrupts?	Yes — by enabling the Timer 0 interrupt and using **P3.4 (T0)** pin.

Q: Why prefer interrupts instead of polling for pulse counting?
A: Interrupts allow the CPU to handle other tasks and respond only when overflow occurs — much more efficient than continuously polling TF1.

Q: What real-world applications use hardware counters?
A: Frequency measurement, RPM counters, event counting, digital tachometers, and production line monitoring systems.

7-Program to transfer data serially with baud rate of 9600

Question Answer

1. What is the purpose of this program?	To transmit data serially through the 8051's UART at 9600 bps.
2. What is the function of SBUF?	**SBUF** is the **Serial Buffer Register** — used to send or receive serial data.
3. What does the instruction SCON = #50H do?	Configures **serial mode 1 (8-bit UART)** and enables **reception (REN=1)**.
4. Why is Timer 1 used here?	Timer 1 generates the **baud rate clock** for serial communication.
5. Why is TH1 = -3 used?	It sets **Timer 1** to overflow at a rate that produces **9600 baud** with an 11.0592 MHz crystal.

6. What is the baud rate?	The **number of bits transmitted per second** (here, 9600 bps).
7. Which pins are used for serial communication in 8051?	**P3.0 (RxD)** for receiving and **P3.1 (TxD)** for transmitting data.
8. What is the role of the TI flag?	The **Transmit Interrupt flag (TI)** becomes **1** when a byte is completely transmitted.
9. What happens when we write data to SBUF?	The microcontroller automatically starts serial transmission of that byte.
10. What is the difference between mode 0, 1, 2, and 3 in serial communication?	- Mode 0: Shift register (8-bit synchronous) - Mode 1: 8-bit UART (variable baud) - Mode 2: 9-bit UART (fixed baud) - Mode 3: 9-bit UART (variable baud).
11. What happens if we don't clear TI after sending a byte?	The next transmission will not start properly — TI must be cleared before sending the next byte.
12. How can we change the baud rate?	By changing the **TH1 value** or by setting the **SMOD bit** in the **PCON** register.
13. What is REN in SCON register?	**REN (bit 4)** enables or disables **reception** of serial data.
14. Can we receive and transmit simultaneously?	Yes, 8051 UART supports **full-duplex communication**.
15. What is the advantage of serial communication?	It requires fewer lines (TxD and RxD) and is ideal for long-distance communication.

Q: Why use serial communication instead of parallel?
A: Serial communication reduces pin usage and allows data transmission over longer distances with less interference.

Q: What real-life devices use serial communication with microcontrollers?
A: Devices like GPS modules, GSM modems, Bluetooth modules (HC-05), serial LCDs, and PCs communicate via UART.

8- Program to receive data serially with baud rate of 9600

Question Answer

Question	Answer
1. What is the purpose of this program?	To receive serial data through 8051 UART at 9600 bps.
2. Which pin receives serial data in 8051?	**P3.0 (RxD)**.
3. What is the function of SBUF in reception?	**SBUF** stores the received byte from serial communication.
4. What does SCON = #50H configure?	Sets **Serial mode 1 (8-bit UART)** and enables reception (REN=1).
5. What is the role of RI flag?	**Receive Interrupt flag (RI)** indicates a byte has been completely received.
6. Why is Timer 1 used in serial communication?	Timer 1 generates **baud rate clock** required for UART transmission and reception.
7. What happens if we don't clear RI after reading SBUF?	UART will not receive the next byte correctly — RI must be cleared after each reception.
8. Can the 8051 receive and transmit data simultaneously?	Yes, 8051 supports **full-duplex serial communication**.
9. How is baud rate calculated?	Using: **Baud = (Oscillator / 12) / 32 × (256 - TH1)** (for mode 1).
10. What is the difference between polling and interrupts for reception?	**Polling:** CPU continuously checks RI flag. **Interrupt:** CPU responds only when RI=1, more efficient.
11. What is REN in SCON?	**REN (bit 4)** enables/disables reception.
12. What is the range of baud rate with 11.0592 MHz crystal?	Standard rates like 9600, 4800, 19200 can be achieved by changing **TH1**.
13. What is the use of echoing received data?	To verify data reception is correct by sending it back.
14. What happens if Timer 1 is not started?	UART will not function properly — baud rate clock won't be generated.

15. Can we receive multiple bytes at once?	8051 can receive bytes sequentially; each byte triggers **RI** flag.

: Why use serial communication instead of parallel?
A: Serial communication requires **fewer pins**, is suitable for **long-distance communication**, and reduces signal interference.

Q: What real-world devices use serial reception?
A: GPS modules, GSM modules, sensors sending data, PCs receiving data via USB–RS232 converters

9- Sequence Generator Using Serial Interface in 8051

Question	Answer
1. What is the purpose of this program?	To generate a sequence of numbers (0–9) and transmit them serially via UART at 9600 bps.
2. Which pin is used for serial transmission?	**P3.1 (TxD)**.
3. How is baud rate configured?	Using **Timer 1** in Mode 2 and setting **TH1 = -3** for 9600 bps with 11.0592 MHz crystal.
4. Why add '0' to the counter before transmission?	To convert **number (0–9)** to its **ASCII representation** for display on terminal.
5. What is the function of SBUF?	Serial buffer for transmitting or receiving a byte.
6. What is TI flag?	Transmit interrupt flag — set when a byte is fully transmitted.
7. Why do we clear TI after each transmission?	To prepare for sending the next byte.
8. What is the role of delay in sequence generator?	To make sequence readable on terminal; prevents numbers from being sent too fast.
9. How can the sequence length be changed?	By adjusting **counter limits** and modifying CJNE comparison.
10. Can this program receive data simultaneously?	Yes, UART is **full-duplex**; SCON's REN bit must be set.

Question	Answer
11. What is the purpose of CJNE instruction?	To check if the counter reached the maximum value and reset it.
12. What is the maximum sequence number in this program?	**9**, then it resets to 0.
13. What is the significance of Timer 1 in this program?	Timer 1 generates the **baud clock** for serial transmission.
14. Can this sequence be modified to transmit alphabets?	Yes, by changing the counter and ASCII addition logic.
15. Why do we use auto-reload mode (Mode 2) for Timer 1?	So the timer reloads automatically to maintain **consistent baud rate**.

Q: Why use serial interface for sequence generator instead of parallel?
A: Serial communication requires fewer pins and allows data transmission over longer distances with minimal interference.

Q: How can this sequence generator be used in real applications?
A: Test equipment, LED displays, digital counters, seven-segment displays, or sending sequences to other microcontrollers for automation.

Question on Using a Virtual Lab for Embedded Systems

A **Virtual Lab** allows you to simulate microcontroller programs (8051, ARM, AVR, PIC) without physical hardware. Popular platforms include **Proteus, Keil µVision Simulator, MPLAB X Simulator, or online embedded system virtual labs**.

Step 1: Choose a Virtual Lab Platform

- **Proteus Design Suite**: Most common for 8051 and Arduino simulation.
- **Keil µVision**: For writing code and debugging; can integrate with Proteus for hardware simulation.
- **Tinkercad (Arduino)**: Online virtual lab for Arduino.
- **Online Embedded System Labs**: Many universities provide virtual labs for microcontrollers.

Step 2: Create a New Project

1. Open the virtual lab (e.g., Proteus).
2. Select **Microcontroller (8051)**.
3. Set the **crystal oscillator** frequency (e.g., 11.0592 MHz for 8051).
4. Add **necessary components**:
    - Serial interface (RS232 terminal) for UART programs.
    - LEDs or 7-segment display for visual output.

Step 3: Write/Import the Program

- Write the program in **Keil µVision** or directly in the virtual lab editor.
- Example: **Sequence generator code** (from your previous 8051 program).
- Compile the code to generate a **HEX file**.

Step 4: Load Program into Microcontroller

- In Proteus: Click on the 8051 microcontroller → Load HEX file (compiled from Keil).
- Connect **TxD pin (P3.1)** to a **Virtual Terminal** for serial output.

Step 5: Simulate the Program

1. Run the simulation.
2. Observe:
    - Serial terminal output for sequence numbers.
    - LEDs or 7-segment displays if connected.
3. Modify code or delays as needed.
4. Pause, reset, or step through instructions for debugging.

Step 6: Debug and Test

- Use **breakpoints** to check program flow.
- Monitor **register values (A, R0, SBUF, TI)** during simulation.

- Verify output matches expected sequence or behavior.

Tips for Embedded Systems Virtual Labs

- Always check **crystal frequency**; baud rates depend on it.
- Make sure **TxD/RxD connections** are correct for serial communication.
- Use **delays** for human-readable output on terminals.
- Virtual labs allow **safe testing** without burning chips or hardware.

Advantages

- Saves time and cost of hardware.
- Safe for testing programs that may have logical errors.
- Easy to observe registers and signals step-by-step.
- Perfect for **viva preparation and lab assignments**.

Question	**Answer**
What is a virtual lab in embedded systems?	A **software simulation environment** that allows testing microcontroller programs without physical hardware.
Why do we use virtual labs?	Saves hardware cost, safe testing, easy debugging, fast development.
Which microcontroller is used in this lab?	**8051 microcontroller**.
How is a program loaded into the virtual microcontroller?	By compiling the program in **Keil** to generate a **HEX file**, then loading it in Proteus.
How do you observe serial output?	Connect **TxD (P3.1)** to a **Virtual Terminal** in the simulator.
How can you debug programs in virtual labs?	Using **breakpoints, single-step execution, and register monitoring**.
What is the purpose of the crystal oscillator in 8051?	Provides the **clock frequency** to time instructions and generate baud rates.
Can multiple peripherals be simulated together?	Yes, LEDs, 7-segment displays, relays, sensors, and serial terminals can be simulated.

Question	Answer
What are the advantages over physical hardware?	Safe, cheaper, repeatable, allows visualization of internal signals.
Can virtual labs handle interrupts?	Yes, most advanced simulators can simulate **timers, serial interrupts, and external interrupts**.

Question Based on 8051 microcontroller.

Question Answer

How do you toggle an LED using 8051?	Configure port pin as output (e.g., P1.0), then **SETB/CLR** in a loop with delay.
How do you receive serial data at 9600 baud?	Set **Timer 1 for 9600 baud**, configure **SCON=50H**, poll **RI flag**, read **SBUF**.
How to generate a square wave using 8051 timer?	Configure timer in **Mode 1**, toggle output pin on overflow, adjust delay for frequency.
How do you interface a 4x4 keypad with 8051?	Connect **rows to input** and **columns to output**, scan matrix by driving columns low and reading rows.
How to find GCD/LCM using 8051 assembly?	Use **registers to implement Euclidean algorithm**, store result in RAM or port.
How do you use interrupts for pulse counting?	Configure **external interrupt (INT0/INT1)** in **edge-triggered mode**, increment counter in ISR.

Question	Answer
What is 8051 microcontroller?	8051 is an **8-bit microcontroller** developed by Intel in 1980 with on-chip ROM, RAM, timers, serial communication, and I/O ports.
How many pins does 8051 have?	**40 pins** in standard DIP package.
What is the word length of 8051?	8-bit word length.

Question	Answer
What is the operating frequency of 8051?	Typically **12 MHz**, varies with crystal oscillator.
What are the I/O ports in 8051?	**P0, P1, P2, P3** — each 8-bit ports.
What is the purpose of SFRs?	Special Function Registers (SFRs) control timers, serial port, interrupts, and I/O operations.
What is the size of internal RAM?	**128 bytes** internal RAM in original 8051.
How many timers are there in 8051?	**Two timers/counters: Timer 0 and Timer 1**.
What is the difference between a timer and a counter?	Timer counts **time using internal clock**, counter counts **external pulses**.
What is the function of the accumulator (A)?	Main **register for arithmetic and logic operations**.

Theory Based Question

Unit I – Introduction to Embedded Systems

Question	Answer
What is an Embedded System?	A specialized computer system designed to perform dedicated functions, often with real-time constraints.
Give examples of Embedded Systems.	Microwave oven, ATM, smart thermostat, digital watch, automotive ECU.
What are the main components of an Embedded System?	CPU, Memory, I/O devices, Timer, Communication interfaces, Sensors/Actuators.
What are key characteristics of Embedded Systems?	Reliability, real-time operation, low power, small size, dedicated function.
What is the difference between general-purpose and embedded systems?	Embedded systems are task-specific; general-purpose computers can run multiple applications.

Unit II – Advanced Microcontrollers and Architecture

Question	Answer
Difference between RISC and CISC?	**RISC: simple instructions, single cycle execution; CISC: complex instructions, multiple cycles.**
Difference between Harvard and Von Neumann Architecture?	**Harvard: separate program & data memory; Von Neumann: shared memory for program & data.**
What is SoC?	**System on Chip integrates CPU, memory, I/O, and peripherals on a single chip.**
Give examples of microcontroller families.	**8051, AVR, ARM Cortex, PIC, ESP32, Arduino (ATmega).**
What are typical I/O interfaces in microcontrollers?	**GPIO, UART, SPI, I2C, ADC/DAC.**

Unit III – Programming Embedded Systems

Question	Answer
Which programming languages are used for embedded systems?	C, C++, Assembly, Embedded C, Python (for high-level boards).
What is Embedded C?	C language with extensions for hardware-level programming on microcontrollers.
Name some embedded development tools.	Compilers, Assemblers, Simulators, Debuggers, IDEs like Keil, MPLAB, Arduino IDE.
What is cross-compilation?	Compiling code on a host system for a different target microcontroller.
What is flashing firmware?	Uploading compiled code to microcontroller memory for execution.

Unit IV – Real-Time Operating Systems (RTOS)

Question	Answer
What is a Real-Time System?	A system that must respond to events within a strict time constraint.
Difference between hard and soft real-time systems?	Hard: missing deadline is catastrophic; Soft: occasional deadline misses are acceptable.
What is task scheduling?	Allocating CPU time to multiple tasks based on priority.
What is an ISR?	Interrupt Service Routine; code executed in response to an interrupt.
Name some RTOS examples.	FreeRTOS, VxWorks, ThreadX, µC/OS.

Unit V – Emerging Trends and Applications

Question	Answer
What is IoT in embedded systems?	Internet of Things: network of smart devices exchanging data.
How is AI/ML used in embedded systems?	Embedded systems can perform intelligent decisions using AI/ML algorithms.
Give examples of wearable embedded devices.	Smartwatch, fitness tracker, health monitor.
What are challenges in embedded systems?	Limited resources, power consumption, security, real-time constraints.

Question	Answer
Future scope of embedded systems?	Smart devices, autonomous vehicles, AI integration, IoT expansion.