Chapter 8: THE MICROCONTROLLER.

SECTION-II (SHORT QUESTIONS)

1. Define microcontroller.
A microcontroller is a small computing device built on a single integrated circuit (chip) that combines a CPU, on-chip memory (RAM and ROM/Flash), I/O ports, timers, serial communication, interrupt logic, and other peripherals
It is basically a "self-contained computer" often used for controlling embedded tasks (e.g., sensors, motors, appliances).
Because it comes with built-in memory and peripherals, it avoids the need for many external components — making it compact and cost-efficient for embedded applications.

2. Define microprocessor.
A microprocessor is a CPU — a central processing unit — implemented as an integrated circuit that performs the processing and executes programs, but does *not* include built-in memory or I/O peripherals; such external components must be added separately.
Thus a microprocessor on its own is just the "brain," and relies on external memory (RAM/ROM), I/O ports, timers, etc., to build a full system.
Microprocessors are suited to systems that need flexibility, external memory expansion, high performance — like desktop computers — rather than compact embedded control.

3. Enlist two differences between a microprocessor & microcontroller.

- A microcontroller has built-in memory (ROM + RAM) and peripherals (I/O ports, timers, serial ports), but a microprocessor requires external memory and external peripherals.
- Microcontroller-based systems tend to be compact, lower-cost and aimed at embedded control tasks; microprocessor-based systems can support larger memory, more external components, and higher performance, suitable for PCs or general computing.

4. Give three applications of microcontrollers.

- Embedded control in household appliances — e.g. washing machines, microwave ovens, remote controls.
- Automotive control systems — e.g. controlling engine operations, sensors, or other car electronics.
- Industrial automation — e.g. managing machinery, timers, sensors, data acquisition, or simple robotics.

5. How much memory space is built in 8051 microcontroller?
The standard 8051 includes 4 KB of on-chip program memory (ROM) for storing the program.

It also has 128 bytes of internal data memory (RAM) for variables, stack, temporary data.

6. Name the interrupts of 8051 microcontroller.
The 8051 offers five interrupt sources:

- Timer 0 overflow interrupt (TF0)
- Timer 1 overflow interrupt (TF1)
- External hardware interrupt 0 (INT0)
- External hardware interrupt 1 (INT1)
- Serial communication interrupt (RI/TI) for Rx/Tx on serial port

7. Describe the sizes of 8051 microcontroller.
The 8051 is an 8-bit microcontroller: its CPU processes data 8 bits at a time.
It has an 8-bit data bus and a 16-bit address bus (so it can address up to 64 KB memory space).

8. How much timers / counters are available with 8051 microcontroller?
There are two 16-bit timers/counters: Timer 0 and Timer 1.

9. Name three groups of instruction set of 8051 microcontroller?
Three main groups of 8051 instruction set include:

- Data Transfer Instructions (e.g. MOV, etc.)
- Arithmetic Instructions (add, subtract, multiply, divide)

- Logical / Bit-Manipulation Instructions (e.g. bit-set, bit-clear, boolean operations) — because 8051 supports bit-addressable memory

10. Enlist various addressing modes of 8051 microcontroller.
8051 supports multiple addressing modes:

- Immediate Addressing (e.g. `MOV A, #0x25`)
- Register Addressing (e.g. operations using registers R0–R7, A, B)
- Direct Addressing (access internal RAM or SFR by explicit address)
- Indirect Addressing (via registers R0 or R1, e.g. `MOV A, @R0`) for internal RAM or external data memory
- Indexed Addressing (for accessing program memory; e.g. via DPTR + A) (on some 8051 variants)

SECTION-III (LONG QUESTIONS — Short Explanations with Examples)

1. Describe the detailed comparison of microprocessor and microcontroller.
A microcontroller integrates CPU + memory + I/O + peripherals (timers, serial, ports) on one chip, enabling it to function as a compact embedded controller — for example, a microcontroller can directly drive LEDs, sensors, or motors without needing external memory or interface chips. In contrast, a microprocessor only provides the CPU; to build a full system you must connect external RAM, ROM, I/O chips, timers, and other peripherals, making the design larger and more flexible (e.g., building a personal computer). Because microcontrollers come as "all-in-one," they are cost-effective, low-power, and ideal for dedicated tasks, while microprocessors suit applications requiring high processing power, memory, and expandability.

2. Describe the architecture of 8051 microcontroller.
The 8051 architecture comprises an 8-bit CPU, 8-bit data bus, and 16-bit address bus, with separate program memory (on-chip 4 KB ROM) and data memory (on-chip 128 bytes RAM), following Harvard architecture (separate code and data spaces).

It also includes built-in peripherals: four 8-bit I/O ports (total 32 I/O pins), two 16-bit timers/counters, a serial communication port (UART), interrupt logic for external, timer and serial interrupts, and special function registers (SFRs) to control these.

Internal RAM is organized into register banks (R0–R7), bit-addressable space, and general data storage; program execution is fetched from ROM (or external code memory if used), and external memory can be added if needed up to 64 KB for code or data.

3. Explain the various addressing modes of 8051 microcontroller.

- **Immediate Addressing**: the operand is given explicitly in the instruction. Example: `MOV A, #0x55` : loads constant 0x55 into accumulator.
- **Register Addressing**: operand is inside one of the internal registers (R0–R7, A, B). Example: `MOV A, R3` moves content of R3 into accumulator.
- **Direct Addressing**: operand located at a specific internal RAM or SFR address. Example: `MOV A, 30h` loads from internal RAM address 0x30.
- **Indirect Addressing**: register (usually R0 or R1) holds the address pointing to data. Example: `MOV A, @R0` : if R0 = 50h, moves data from internal RAM at 50h into A.
- **Indexed Addressing** (for accessing program memory): uses a pointer plus index (e.g. DPTR + accumulator) to fetch code or data from program memory. Example: `MOVC A, @A+DPTR`. This is typically used when fetching lookup tables from code memory.