MODULE 3
PROGRAMMING 8051 AT ASSEMBLY LEVEL

3.1ADDRESSING MODES OF 8051:

The 8051 microcontroller employs several addressing modes to facilitate efficient and flexible programming. These addressing modes allow the programmer to access data and instructions in various ways, optimizing code size and execution speed. Here are the addressing modes supported by the 8051:

1. **Immediate Addressing Mode:**

- **Format:** MOV A, #data
- **Description:** In this mode, the operand (data) is directly specified in the instruction itself. The data is typically an 8-bit constant (0-255) that is immediately loaded into the accumulator (A) or a register.

2. **Direct Addressing Mode:**

- **Format:** MOV A, address
- **Description:** The operand (address) specifies the actual memory address where the data is located. The data at that memory location is then accessed or manipulated. Direct addressing is used for accessing variables stored in specific memory locations.

3. **Register Addressing Mode:**

- **Format:** MOV A, R0
- **Description:** In this mode, the operand is a register (R0, R1, R2, R3, R4, R5, R6, R7). The data stored in the specified register is accessed or manipulated. This mode is used for fast access to data stored in registers.

4. **Register Indirect Addressing Mode:**

- **Format:** MOV A, @R0
- **Description:** The operand is an indirect address contained in a register pair (DPTR or R0/R1). The value in the register pair is used as a pointer to access data in external RAM or specific memory locations. Example: MOV A, @R0 accesses the data pointed to by the address in register R0.

5. **Indexed Addressing Mode (XDATA):**

- **Format:** MOVX A, @DPTR
- **Description:** Used specifically for external data memory access. The instruction accesses data at the address specified by DPTR (Data Pointer). This mode is useful when interfacing with external RAM or memory-mapped peripherals.

6. **Relative Addressing Mode:**

- **Format:** Conditional jump instructions (e.g., JZ, JNZ, JC, etc.)
- **Description:** These instructions use a signed 8-bit offset to determine the target address for conditional branching. The offset is added to the current value of the Program Counter (PC), allowing the program to jump to a location within -128 to +127 bytes relative to the current PC.

7. **Bit Addressing Mode:**

- **Format:** SETB bit or CLR bit
- **Description:** Allows individual bits within a byte in special function registers (SFRs) or general-purpose RAM locations to be set or cleared. This mode is essential for controlling specific bits that represent flags, status bits, or control signals.

8. **Immediate Addressing for Data Memory (IDATA):**

- **Format:** MOV R0, #data
- **Description:** Similar to immediate addressing mode, but used specifically for accessing data in internal RAM (IDATA space). The data is an 8-bit constant that is immediately loaded into a register or memory location.

3.2INSTRUCTION SETS:

- The instructions of 8051 can be broadly classified under the following headings.
  1. Data transfer Instructions
  2. Arithmetic Instructions
  3. Logical Instructions
  4. Program Branching Instructions
  5. Bit Manipulation Instructions / Boolean Variable Manipulation Instructions

1. Data Transfer Instructions:

- **MOV A, data:** Move immediate data to accumulator.
- **MOV Rn, data:** Move immediate data to register Rn.
- **MOV direct, data:** Move immediate data to direct address.
- **MOV @Ri, data:** Move immediate data to indirect address pointed by register Ri.
- **MOVX A, @DPTR:** Move external data pointed by DPTR to accumulator.

2. Arithmetic Instructions:

- **ADD A, data:** Add immediate data to accumulator.
- **ADD A, Rn:** Add register Rn to accumulator.
- **ADD A, direct:** Add data at direct address to accumulator.
- **ADD A, @Ri:** Add data at indirect address pointed by register Ri to accumulator.
- **INC Rn:** Increment register Rn.
- **INC direct:** Increment data at direct address.

3. Logical Instructions:

- **ANL A, data:** Bitwise AND immediate data with accumulator.
- **ANL A, Rn:** Bitwise AND register Rn with accumulator.
- **ORL A, data:** Bitwise OR immediate data with accumulator.
- **ORL A, Rn:** Bitwise OR register Rn with accumulator.
- **XRL A, data:** Bitwise XOR immediate data with accumulator.
- **XRL A, Rn:** Bitwise XOR register Rn with accumulator.
- **CJNE A, data, rel:** Compare accumulator with immediate data and jump if not equal.
- **CJNE A, direct, rel:** Compare accumulator with data at direct address and jump if not equal.

4. Branching Instructions:

- **JMP addr:** Unconditional jump to specified address.
- **JZ addr:** Jump if accumulator is zero.
- **JNZ addr:** Jump if accumulator is not zero.
- **JC addr:** Jump if carry flag is set.
- **DJNZ Rn, rel:** Decrement register Rn and jump if not zero.

5. Bit Manipulation Instructions:

- **SETB bit:** Set specified bit in a special function register or RAM.
- **CLR bit:** Clear specified bit in a special function register or RAM.

6. Data Pointer and Stack Instructions:

- **MOV DPTR, addr:** Load DPTR (Data Pointer) with 16-bit address.
- **PUSH direct:** Push data at direct address onto stack.
- **POP direct:** Pop data from stack to direct address.

7. Control Instructions:

NOP: No operation.

- **HLT:** Halt CPU operation.
- **MOVX @DPTR, A:** Move accumulator to external RAM pointed by DPTR.

- **MOVC A, @A+DPTR:** Move code byte to accumulator from address A+DPTR.
- **LCALL addr:** Long call to subroutine at specified address.
- **RET:** Return from subroutine.
- **RETI:** Return from interrupt.

8. **Special Function Register (SFR) Instructions:**

- **MOVX A, @Ri:** Move data from external RAM pointed by register Ri to accumulator.
- **MOVX @DPTR, A:** Move accumulator to external RAM pointed by DPTR.
- **MOVX A, @DPTR:** Move data from external RAM pointed by DPTR to accumulator.
- **MOVX @Ri, A:** Move accumulator to external RAM pointed by register Ri.

9. **Serial Communication Instructions**:

- **MOV SBUF, A:** Move accumulator to serial buffer register.
- **MOV A, SBUF:** Move data from serial buffer register to accumulator.

10. **Timer and Interrupt Instructions:**

- **MOV TMOD, data:** Load Timer/Counter mode register with data.
- **MOV TCON, data:** Load Timer/Counter control register with data.
- **MOV THx, data:** Load Timer/Counter high byte register with data.
- **MOV TLx, data:** Load Timer/Counter low byte register with data.

Note 1: In Multiplication,

- A holds MSB of the result and B holds LSB of the result
- This instruction always makes CY and OV flag as '0', if result is less than 'FF'
- If CY=0 and OV=1, indicates the result is above 'FF'

Note 2: In division,

- A holds Quotient and B holds Remainder
- This instruction always makes CY and OV flag as '0', if denominator is not '0'
- If CY=0 and OV=1, indicates an error that denominator is zero