



SNS COLLEGE OF TECHNOLOGY
An Autonomous Institution
Coimbatore-35



Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

19ITT204 - MICROCONTROLLER AND EMBEDDED SYSTEMS

II YEAR/ IV SEMESTER

UNIT III EMBEDDED SYSTEM CONCEPTS AND PROCESSORS

TOPIC – 8051 INSTRUCTION SET



Instruction Groups

- The 8051 has 255 instructions
 - Every 8-bit opcode from 00 to FF is used except for A5.
 - The instructions are grouped into 5 groups
 - Arithmetic
 - Logic
 - Data Transfer
 - Boolean
 - Branching
-



Arithmetic Instructions

- **ADD**
 - 8-bit addition between the accumulator (A) and a second operand.
 - The result is always in the accumulator.
 - The CY flag is set/reset appropriately.
 - **ADDC**
 - 8-bit addition between the accumulator, a second operand and the previous value of the CY flag.
 - Useful for 16-bit addition in two steps.
 - The CY flag is set/reset appropriately.
-



Example – 16-bit Addition

Add 1E44H to 56CAH

CLR	C	; Clear the CY flag
MOV	A, 44H	; The lower 8-bits of the 1 st number
ADD	A, CAH	; The lower 8-bits of the 2 nd number
MOV	R1, A	; The result 0EH will be in R1. CY = 1.
MOV	A, 1EH	; The upper 8-bits of the 1 st number
ADDC	A, 56H	; The upper 8-bits of the 2 nd number
MOV	R2, A	; The result of the addition is 75H

The overall result: 750EH will be in R2:R1. CY = 0.



Arithmetic Instructions

- DA
 - Decimal adjust the accumulator.
 - Format the accumulator into a proper 2 digit packed BCD number.
 - Operates only on the accumulator.
 - Works only after the ADD instruction.

 - SUBB
 - Subtract with Borrow.
 - Subtract an operand and the previous value of the borrow (carry) flag from the accumulator.
 - $A \leftarrow A - \langle \text{operand} \rangle - CY$.
 - The result is always saved in the accumulator.
 - The CY flag is set/reset appropriately.
-



Example – BCD addition

Add 34 to 49 BCD

CLR	C	; Clear the CY flag
MOV	A, #34H	; Place 1 st number in A
ADD	A, #49H	; Add the 2 nd number.
		; A = 7DH
DA	A	; A = 83H



Arithmetic Instructions

- **INC**
 - Increment the operand by one.
 - The operand can be a register, a direct address, an indirect address, the data pointer.
- **DEC**
 - Decrement the operand by one.
 - The operand can be a register, a direct address, an indirect address.
- **MUL AB / DIV AB**
 - Multiply A by B and place result in A:B.
 - Divide A by B and place result in A:B.



Logical Operations

- ANL / ORL
 - Work on byte sized operands or the CY flag.
 - ANL A, Rn
 - ANL A, direct
 - ANL A, @Ri
 - ANL A, #data
 - ANL direct, A
 - ANL direct, #data

 - ANL C, bit
 - ANL C, /bit



Logical Operations

- XRL
 - Works on bytes only.
- CPL / CLR
 - Complement / Clear.
 - Work on the accumulator or a bit.
 - CLR P1.2



Logical Operations

- **RL / RLC / RR / RRC**
 - Rotate the accumulator.
 - RL and RR without the carry
 - RLC and RRC rotate through the carry.
- **SWAP A**
 - Swap the upper and lower nibbles of the accumulator.
- **No compare instruction.**
 - Built into conditional branching instructions.



Data Transfer Instructions

- MOV
 - 8-bit data transfer for internal RAM and the SFR.
 - MOV A, Rn
 - MOV A, direct
 - MOV A, @Ri
 - MOV A, #data
 - MOV Rn, A
 - MOV Rn, direct
 - MOV Rn, #data
 - MOV direct, A
 - MOV direct, Rn
 - MOV direct, direct
 - MOV direct, @Ri
 - MOV direct, #data
 - MOV @Ri, A
 - MOV @Ri, direct
 - MOV @Ri, #data



Data Transfer Operations

- MOV
 - 1-bit data transfer involving the CY flag
 - MOV C, bit
 - MOV bit, C

- MOV
 - 16-bit data transfer involving the DPTR
 - MOV DPTR, #data



Data Transfer Instructions

- **MOVC**
 - **Move Code Byte**
 - Load the accumulator with a byte from program memory.
 - Must use indexed addressing
 - **MOVC A, @A+DPTR**
 - **MOVC A, @A+PC**



Data Transfer Instructions

- MOVX
 - Data transfer between the accumulator and a byte from external data memory.
 - MOVX A, @Ri
 - MOVX A, @DPTR
 - MOVX @Ri, A
 - MOVX @DPTR, A



Data Transfer Instructions

- **PUSH / POP**
 - Push and Pop a data byte onto the stack.
 - The data byte is identified by a direct address from the internal RAM locations.

- PUSH DPL
- POP 40H



Data Transfer Instructions

- XCH
 - Exchange accumulator and a byte variable
 - XCH A, Rn
 - XCH A, direct
 - XCH A, @Ri

 - XCHD
 - Exchange lower digit of accumulator with the lower digit of the memory location specified.
 - XCHD A, @Ri

 - The lower 4-bits of the accumulator are exchanged with the lower 4-bits of the internal memory location identified indirectly by the index register.
 - The upper 4-bits of each are not modified.
-



Boolean Operations

- This group of instructions is associated with the single-bit operations of the 8051.
 - This group allows manipulating the individual bits of bit addressable registers and memory locations as well as the CY flag.
 - The P, OV, and AC flags cannot be directly altered.
 - This group includes:
 - Set, clear, and, or complement, move.
 - Conditional jumps.
-



Boolean Operations

- CLR
 - Clear a bit or the CY flag.
 - CLR P1.1
 - CLR C

 - SETB
 - Set a bit or the CY flag.
 - SETB A.2
 - SETB C

 - CPL
 - Complement a bit or the CY flag.
 - CPL 40H ; Complement bit 40 of the bit addressable memory
-



Boolean Operations

- **ORL / ANL**
 - **OR / AND a bit with the CY flag.**
 - `ORL C, 20H` ; OR bit 20 of bit addressable memory with the CY flag
 - `ANL C, /34H` ; AND complement of bit 34 of bit addressable memory with the CY flag.
 - **MOV**
 - **Data transfer between a bit and the CY flag.**
 - `MOV C, 3FH` ; Copy the CY flag to bit 3F of the bit addressable memory.
 - `MOV P1.2, C` ; Copy the CY flag to bit 2 of P1.
-



Boolean Operations

- **JC / JNC**
 - Jump to a **relative address** if CY is set / cleared.
 - **JB / JNB**
 - Jump to a **relative address** if a bit is set / cleared.
 - JB ACC.2, <label>
 - **JBC**
 - Jump to a relative address if a bit is set and clear the bit.
-



Branching Instructions

- The 8051 provides four different types of unconditional jump instructions:
 - Short Jump – SJMP
 - Uses an 8-bit signed offset relative to the 1st byte of the next instruction.
 - Long Jump – LJMP
 - Uses a 16-bit address.
 - 3 byte instruction capable of referencing any location in the entire 64K of program memory.



Branching Instructions

- Absolute Jump – AJMP
 - Uses an **11-bit address**.
 - 2 byte instruction
 - The upper 3-bits of the address combine with the 5-bit opcode to form the 1st byte and the lower 8-bits of the address form the 2nd byte.
 - The 11-bit address is substituted for the lower 11-bits of the PC to calculate the 16-bit address of the target.
 - The location referenced must be within the 2K Byte memory page containing the AJMP instruction.

- Indirect Jump – JMP
 - `JMP @A + DPTR`



Branching Instructions

- The 8051 provides 2 forms for the CALL instruction:
 - Absolute Call – ACALL
 - Uses an 11-bit address similar to AJMP
 - The subroutine must be within the same 2K page.
 - Long Call – LCALL
 - Uses a 16-bit address similar to LJMP
 - The subroutine can be anywhere.
 - Both forms push the 16-bit address of the next instruction on the stack and update the stack pointer.
-



Branching Instructions

- Compare and Jump if Not Equal – CJNE
 - Compare the magnitude of the two operands and jump if they are not equal.
 - The values are considered to be unsigned.
 - The Carry flag is set / cleared appropriately.

 - CJNE A, direct, rel
 - CJNE A, #data, rel
 - CJNE Rn, #data, rel
 - CJNE @Ri, #data, rel
-



Branching Instructions

- Decrement and Jump if Not Zero – DJNZ
 - Decrement the first operand by 1 and jump to the location identified by the second operand if the resulting value is not zero.
 - DJNZ Rn, rel
 - DJNZ direct, rel
- No Operation
 - NOP



THANK YOU