INSTRUCTION SET OF 8051

The instructions of 8051 can be broadly classified under the following headings.

1. Data transfer instructions
2. Arithmetic instructions
3. Logical instructions
4. Branch instructions
5. Subroutine instructions
6. Bit manipulation instructions

1. Data transfer instructions.

In this group, the instructions perform data transfer operations of the following types.

a. Move the contents of a register Rn to A

- *i.* MOV A,R2

b. Move the contents of a register A to Rn

- *i.* MOV R4,A

c. Move an immediate 8 bit data to register A or to Rn or to a memory location(direct or indirect)

i.	MOV A, #45H	(Move value 45 to A)
ii.	MOV R6, #51H	(Move value 51 to R6)
iii.	MOV 30H, #44H	(Move value 44 to RAM location 30)
iv.	MOV @R0, #0E8	(Move value E8 to RAM location addressed by R0)
v.	MOV DPTR, #0F5A2H	(Move value F5A2 to DPTR)

d. Move the contents of a memory location to A or A to a memory location using direct and indirect addressing

i.	MOV A, 65H	(Move data at RAM location 65 to A)
ii.	MOV A, @R0	(Move data at RAM location addressed by R0 to A)
iii.	MOV 45H, A	(Move data at A to RAM location 45)
iv.	MOV @R1, A	(Move data at A to RAM location addressed by R1)

e. Move the contents of a memory location to Rn or Rn to a memory location using direct addressing

i.	MOV R3, 65H	(Move data at RAM location 65 to R3)
ii.	MOV 45H, R2	(Move data at R2 to RAM location 45)

f. Move the contents of memory location to another memory location using direct and indirect addressing

i.	MOV 47H, 65H	(Move data at RAM location 65 to RAM location 47)
ii.	MOV 45H, @R0	(Move data at RAM location addressed by R0 to RAM location 45)

g. Move the contents of an external memory to A or A to an external memory

i.	MOVX A,@DPTR	(Move data at external RAM location addressed by DPTR to A)
ii.	MOVX@DPTR,A	(Move data at A to external RAM location addressed by DPTR)

h. Move the contents of program memory to A

- *i.* MOVC A, @A+PC
- *ii.* MOVC A, @A+DPTR

MOVC moves a byte from Code Memory into the Accumulator. The Code Memory address from which the byte will be moved is calculated by summing the value of the Accumulator with either DPTR or the Program Counter (PC). In the case of the Program Counter, PC is first incremented by 1 before being summed with the Accumulator.

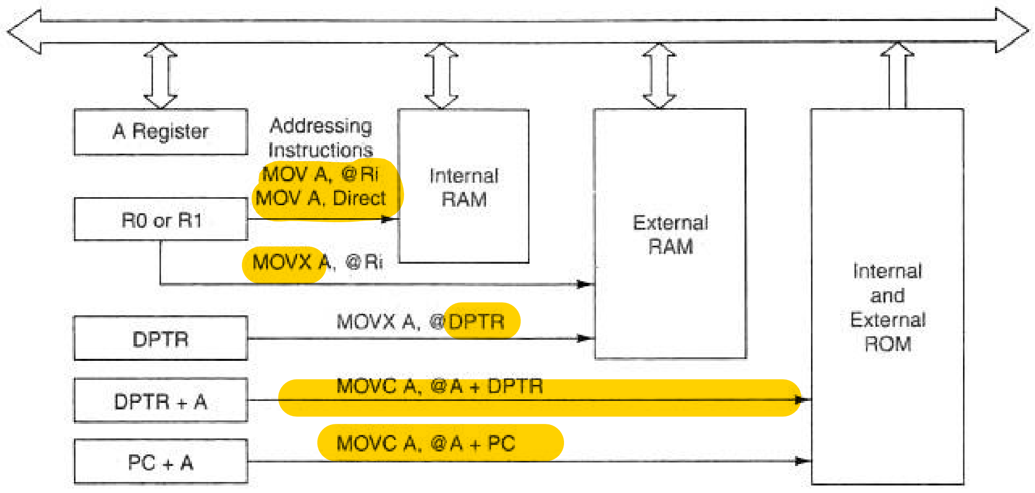


FIG. Addressing Using MOV, MOVX and MOVC

i. Push and Pop instructions

PUSH 6	Increment SP (SP=SP+1) then, data at RAM location 06 is moved to stack memory addressed by SP.
POP 7	Data at stack memory addressed by SP is moved to RAM location 07 then decrement SP (SP=SP-1).

j. Exchange instructions

The content of source ie, register, direct memory or indirect memory will be exchanged with the contents of destination ie, accumulator.

i.	XCH A,R3	Exchange the data at A and register R3
ii.	XCH A,@R1	Exchange the data at A and RAM location addressed by R1
iii.	XCH A,54h	Exchange the data at A and RAM location 54

k. Exchange digit. Exchange the lower order nibble of Accumulator (A0-A3) with lower order nibble of the internal RAM location which is indirectly addressed by the register.

i. XCHD A,@R1

ii. XCHD A,@R0

2. Arithmetic instructions.

The 8051 can perform addition, subtraction. multiplication and division operations on 8 bit numbers.

Addition

In this group, we have instructions to

i. Add the contents of A with immediate data with or without carry.
   - i. ADD A, #45H (A = A + 45)
   - ii. ADDC A, #OB4H (A = A + 45 + CY Flag)

ii. Add the contents of A with register Rn with or without carry.
   - i. ADD A, R5 (A = A + Data at R5)
   - ii. ADDC A, R2 (A = A + Data at R2+ CY Flag)

iii. Add the contents of A with contents of memory with or without carry using direct and indirect addressing

i. ADD A, 51H (A = A + Data at RAM location 51)
ii. ADDC A, 75H (A = A + Data at RAM location 75 + CY Flag)
iii. ADD A, @R1 (A = A + Data at RAM location addressed by R1)
iv. ADDC A, @R0 (A = A + Data at RAM location addressed by R0 + CY Flag)

Subtraction

In this group, we have instructions to

i. Subtract the contents of A with immediate data with carry.
   SUBB A, #45H (A = A - 45 - CY Flag)

ii. Subtract the contents of A with register Rn with carry.
   SUBB A, R5 (A = A - Data at R2 - CY Flag)

i. Subtract the contents of A with contents of memory with or without carry using direct and indirect addressing
   - i. SUBB A, 51H (A = A + Data at RAM location 51 - CY Flag)
   - ii. SUBB A, @R1 (A = A - Data at RAM location addressed by R1 - CY Flag)

Multiplication

MUL A B.

- ➢ This instruction multiplies two 8 bit unsigned numbers which are stored in A and B registers.
- ➢ After multiplication the lower byte of the result will be stored in accumulator and higher byte of result will be stored in B register.

```
Eg.     MOV A,#45H          ;[A]=45H
        MOV B,#0F5H         ;[B]=F5H
        MUL AB              ;[A] x [B] = 45 x F5 = 4209
                            ;[A]=09H, [B]=42H
```

Division

DIV A B.

- ➢ This instruction divides the 8 bit unsigned number which is stored in A by the 8 bit unsigned number which is stored in B register.
- ➢ After division the result will be stored in accumulator and remainder will be stored in B register.

```
Eg.    ~~MOV A,#45H~~        ;[A]=0E8H                                          MOV A,#0E8H
       ~~MOV B,#0F5H~~       ;[B]=1BH                                           MOV B,#1BH
       ~~DIV AB~~            ;[A] / [B] = E8 /1B = 08 H with remainder  10H     DIV AB
                          ;[A] = 08H, [B]=10H
```

DA A (Decimal Adjust After Addition).

When two BCD numbers are added, the answer is a non-BCD number. To get the result in BCD, we use DA A instruction after the addition. DA A works as follows.

- If lower nibble is greater than 9 or auxiliary carry is 1, 6 is added to lower nibble.
- If upper nibble is greater than 9 or carry is 1, 6 is added to upper nibble.

Increment: *increments the operand by one.*

- ➢ INC increments the value of source by 1.
- ➢ If the initial value of register is FFh, incrementing the value will cause it to reset to 0.

INC A	A=A+1
INC Rn	Eg; INC R4; R4= R4+1
INC DIRECT	Eg; INC 54; RAM location 54= RAM location 54 +1
INC @Ri	Eg; INC @R3; RAM location addressed by R3 = RAM location addressed by R3 +1
INC DPTR	DPTR=DPTR+1

Decrement: *decrements the operand by one.*

- ➢ DEC decrements the value of *source* by 1.
- ➢ If the initial value of is 0, decrementing the value will cause it to reset to FFh.

DEC A	A=A-1
DEC Rn	Eg; DEC R4; R4= R4-1
DEC DIRECT	Eg; DEC54; RAM location 54= RAM location 54 -1
DEC @Ri	Eg; DEC @R3; RAM location addressed by R3 = RAM location addressed by R3 -1

3. Logical Instructions

Logical AND

ANL destination, source:

- ANL does a bitwise "AND" operation between *source* and *destination*, leaving the resulting value in *destination*.

ANL A,#DATA	eg:- ANL A,#09 (A = A *and* value 09)
ANL A, Rn	eg:- ANL A,R1 (A = A *and* Data at R1)
ANLA, DIRECT	eg:- ANL A,46 (A = A *and* Data at RAM location 41)
ANL A,@Ri	eg:- ANL A,@R4 (A = A *and* Data at RAM location addressed by R4)
ANL DIRECT,A	eg:- ANL 45,A (RAM location 45 = RAM location 45 *and* A)
ANL DIRECT, #DATA	eg:- ANL 45,#FF (RAM location 45 = RAM location 45 *and* value FF)

Logical OR

- **ORL** destination, source: ORL does a bitwise "OR" operation between *source* and *destination*, leaving the resulting value in *destination*.

ORL A,#DATA	eg:- ORL A,#09 (A = A *or* value 09)
ORL A, Rn	eg:- ORL A,R1 (A = A *or* Data at R1)
ORL A,DIRECT	eg:- ORL A,46 (A = A *or* Data at RAM location 41)
ORL A,@Ri	eg:- ORL A,@R4 (A = A *or* Data at RAM location addressed by R4)
ORL DIRECT,A	eg:- ORL 45,A (RAM location 45 = RAM location 45 *or* A)
ORL DIRECT, #DATA	eg:- ORL 45,#FF (RAM location 45 = RAM location 45 *or* value FF)

Logical Ex-OR

- XRL destination, source: XRL does a bitwise "EX-OR" operation between *source* and *destination*, leaving the resulting value in *destination*.

XRL A,#DATA	eg:- XRL A,#09 (A = A x*or* value 09)
XRL A,Rn	eg:- XRL A,R1 (A = A x*or* Data at R1)
XRL A,DIRECT	eg:- XRL A,46 (A = A *xor* Data at RAM location 41)
XRL A,@Ri	eg:- XRL A,@R4 (A = A x*or* Data at RAM location addressed by R4)
XRL DIRECT,A	eg:- XRL 45,A (RAM location 45 = RAM location 45 x*or* A)
XRL DIRECT, #DATA	eg:- XRL 45,#FF (RAM location 45 = RAM location 45 x*or* value FF)

Logical NOT

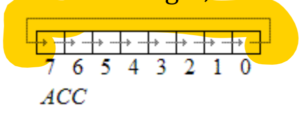
- CPL complements *operand*, leaving the result in *operand*. If *operand* is a single bit then the state of the bit will be reversed.

CPL A	Invert all the bits of A.
CPL C	Invert Carry flag.
CPL bit address	Eg:- CPL 07 07 is the address of 8th bit of first bit addressable memory location . Invert it.

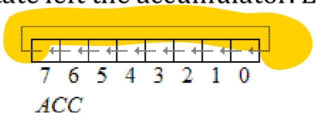
SWAP A – Swap the upper nibble and lower nibble of A.

<u>Rotate Instructions</u>

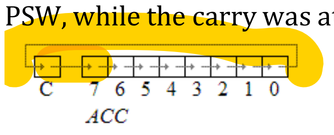
RR A
This instruction is rotate right the accumulator. Its operation is illustrated below. Each bit is shifted one location to the right, with bit 0 going to bit 7.



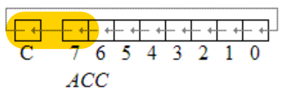
RL A
Rotate left the accumulator. Each bit is shifted one location to the left, with bit 7 going to bit 0



RRC A
Rotate right through the carry. Each bit is shifted one location to the right, with bit 0 going into the carry bit in the PSW, while the carry was at goes into bit 7



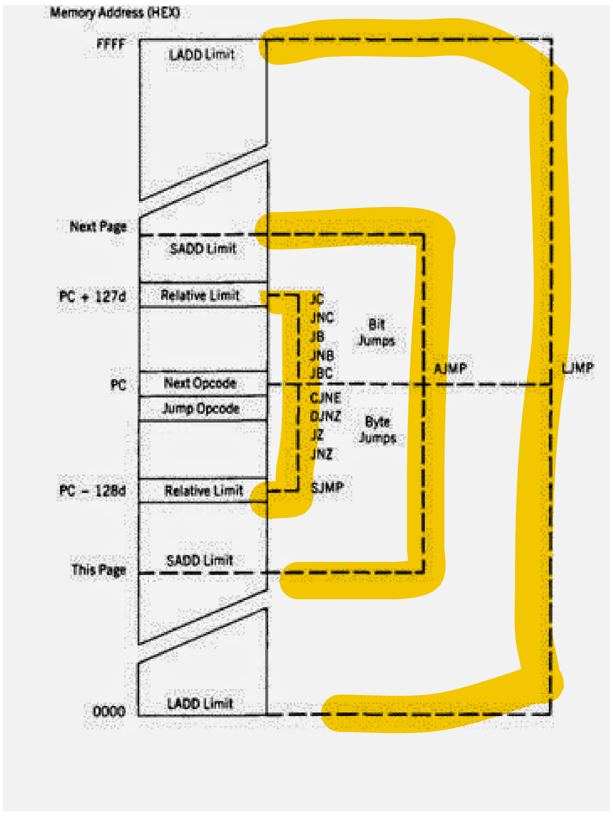
RLC A
Rotate left through the carry. Each bit is shifted one location to the left, with bit 7 going into the carry bit in the PSW, while the carry goes into bit 0.



4. **Branch (JUMP) Instructions**

- A jump or call instruction can replace the contents of program counter with a new address give in the instruction that causes the program execution to begin at the code stored at the ne address.

There are 3 types of jump instructions. They are:-
1. Relative Jump
2. Short Absolute Jump
3. Long Absolute Jump



Jump instruction ranges

1. *Relative Jump*

Jump allows only to a range of bytes within +127d or -128d from the instruction following jump.

Relative Jump Instructions:

a. *Unconditional jump:*

SJMP <relative address> ; Jump to the address specified unconditionally. Jump allows only to relative range.

b. *Conditional jump:*

- Bit level Jump Instructions:

JC <relative address>	Jump if CY = 1
JNC <relative address>	Jump if CY = 0

JB bit, <relative address>
Eg; JB 08, <relative address>
Jump to the relative address specified if the bit with address 08 = 1

JNB bit, <relative address>
Eg; JNB 08, <relative address>
Jump to the relative address specified if the bit with address 08 = 0

JBC bit, <relative address>
jump to the relative address specified if the direct bit is set and then clear the bit.

- Byte level Jump Instructions:

CJNE <destination byte>, <source byte>, <relative address>
Compare the two bytes of data(destination byte and source byte) and if these are not equal then jump to the relative address specified.

For Eg: CJNE A,56, <relative address>
Compare the data at A and data at RAM location 56 and if these are not equal then jump to the relative address specified.

DJNZ Rn <relative address>
Decrement the data at register Rn and jump to the relative address specified if the decremented value not equal to 0.

JZ <relative address>   Jump to the relative address specified if data at A equal to 0.

JNZ <relative address>   Jump to the relative address specified if data at A not equal to 0.

2. ***Short Absolute Jump***

   - In 8051, 64 K byte of program memory space is divided into 32 pages of 2K byte each.
   - Jump allows only to a range of bytes within the 2K byte from the instruction following jump.

Example of short absolute jump: -
AJMP < address > ; Jump to the address specified unconditionally. Jump allows only to 2K range.

3. ***Long Absolute Jump***

   - Applications that need to access the entire program memory from 0000H to FFFFH use long absolute jump.
   - Jump allows to any locations among the 64 K memory.

LJMP <address> ; Jump to the address specified unconditionally. Jump allows to anywhere.

5. **Subroutine CALL And RETURN Instructions**

- Subroutines are subprograms used to simplify the programs whenever a particular task is to be done repeatedly.
- Subroutines are handled by CALL and RET instructions

There are two types of CALL instructions

1. **LCALL address (16 bit)**
   This is long call instruction which unconditionally calls the subroutine located at the indicated 16 bit address. Call the subroutine located anywhere in the program memory space. Push the address of next instruction immediately after the LCALL instruction on to the stack.

2. **ACALL address**
   This is absolute call instruction which unconditionally calls the subroutine located at the specified address. Call the subroutine located on the same page. Push the address of next instruction immediately after the LCALL instruction on to the stack.

RET instruction
RET instruction pops top two contents from the stack and load it to PC

6. Bit Manipulation Instructions.

➢ 8051 has 128 bit addressable memory, bit addressable SFRs and bit addressable PORT pins.
➢ It is possible to perform following bit wise operations for these bit addressable locations.

1. LOGICAL AND
   a. ANL C,BIT(BIT ADDRESS) ; 'LOGICALLY AND' CARRY AND CONTENT OF BIT ADDRESS, STORE RESULT IN CARRY
   b. ANL C, /BIT; ; 'LOGICALLY AND' CARRY AND COMPLEMENT OF CONTENT OF BIT ADDRESS, STORE RESULT IN CARRY
2. LOGICAL OR
   a. ORL C,BIT(BIT ADDRESS) ; 'LOGICALLY OR' CARRY AND CONTENT OF BIT ADDRESS, STORE RESULT IN CARRY
   b. ORL C, /BIT; ; 'LOGICALLY OR' CARRY AND COMPLEMENT OF CONTENT OF BIT ADDRESS, STORE RESULT IN CARRY
3. CLR bit
   a. CLR bit ; CONTENT OF BIT ADDRESS SPECIFIED WILL BE CLEARED.
   b. CLR C ; CONTENT OF CARRY WILL BE CLEARED.
4. CPL bit
   a. CPL bit ; CONTENT OF BIT ADDRESS SPECIFIED WILL BE COMPLEMENTED.
   b. CPL C ; CONTENT OF CARRY WILL BE COMPLEMENTED.

2.2 ADDRESSING MODES

➢ The method by which the operand is specified in an instruction is called addressing modes.
➢ 8051 addressing modes are classified as follows.

1. Immediate addressing.
2. Register addressing.
3. Direct addressing.
4. Indirect addressing.
5. Indexed addressing

1. Immediate addressing.

➢ In this addressing mode the data is provided as a part of instruction itself.
➢ In other words data immediately follows the instruction.

Eg.	MOV A,#30H	It means that immediate data 30h provided in instruction is moved into A register.
	ADD A, #83	It means that immediate data 30h provided in instruction is added with A register.

# Symbol indicates the data is immediate.

2. ***Register addressing.***

- In this addressing mode the register will hold the data.
- One of the eight general registers (R0 to R7) can be used and specified as the operand.

Eg. MOV A,R0 Content of R0 register is copied into Accumulator.

ADD A,R6 Content of R0 register is added with Accumulator.

R0 – R7 will be selected from the current selection of register bank. The default register bank will be bank 0.

3. ***Direct addressing***

- In this mode the direct address of memory location is provided in instruction to fetch the operand.
- Only internal RAM and SFR's address can be used in this type of instruction.
  Ex: MOV A, 30H => Content of RAM address 30H is copied into Accumulator.

4. ***Indirect addressing***

- Here the address of memory location is indirectly provided by a register.
- The '@' sign indicates that the register holds the address of memory location i.e. fetch the content of memory location whose address is stored in register.
  Ex: MOV A,@R0 => Copy the content of memory location whose address is given in R0 register.

5. ***Indexed addressing***

- This addressing mode is basically used for accessing data from look up table.
- Here the address of memory is indexed i.e. added to form the actual address of memory.

Ex: MOVC A,@A+DPTR => here 'C' means Code. Here the content of A register is added with content of DPTR and the resultant is the address of memory location from where the data is copied to A register.

[redacted]