MICROCONTROLLER & EMBEDDED SYSTEMS

UNIT-2

2.1 8051 INSTRUCTION SET

Mnemonic		Description	Byte
ARITHMETIC OPERATIONS			
ADD	A,Rn	Add register to Accumulator	1
ADD	A,direct	Add direct byte to Accumulator	2
ADD	A,@R_i	Add indirect RAM to Accumulator	1
ADD	A,#data	Add immediate data to Accumulator	2
ADDC	A,R_n	Add register to Accumulator with Carry	1
ADDC	A,direct	Add direct byte to Accumulator with Carry	2
ADDC	A,@R_i	Add indirect RAM to Accumulator with Carry	1
ADDC	A,#data	Add immediate data to Accumulator with Carry	2
SUBB	A,R_n	Subtract Register from Acc with Borrow	1
SUBB	A,direct	Subtract direct byte from Acc with Borrow	2
SUBB	A,@R_i	Subtract indirect RAM from Acc with Borrow	1
SUBB	A,#data	Subtract immediate data from Acc with Borrow	2
INC	A	Increment Accumulator	1
INC	R_n	Increment Register	1
INC	direct	Increment direct byte	2
INC	@R_i	Increment indirect RAM	1
DEC	A	Decrement Accumulator	1
DEC	R_n	Decrement Register	1
DEC	direct	Decrement direct byte	2
DEC	@R_i	Decrement indirect RAM	1
INC	DPTR	Increment Data Pointer	1
MUL	AB	Multiply A and B	1
DIV	AB	Divide A by B	1
DA	A	Decimal Adjust Accumulator	1

Mnemonic		Description	Byte
LOGICAL OPERATIONS			
ANL	A,R_n	AND Register to Accumulator	1
ANL	A,direct	AND direct byte to Accumulator	2
ANL	A,@R_i	AND indirect RAM to Accumulator	1
ANL	A,#data	AND immediate data to Accumulator	2
ANL	direct,A	AND Accumulator to direct byte	2
ANL	direct,#data	AND immediate data to direct byte	3
ORL	A,R_n	OR Register to Accumulator	1
ORL	A,direct	OR direct byte to Accumulator	2
ORL	A,@R_i	OR indirect RAM to Accumulator	1
ORL	A,#data	OR immediate data to Accumulator	2
ORL	direct,A	OR Accumulator to direct byte	2
ORL	direct,#data	OR immediate data to direct byte	3
XRL	A,R_n	Exclusive-OR Register to Accumulator	1
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2
XRL	A,@R_i	Exclusive-OR indirect RAM to Accumulator	1
XRL	A,#data	Exclusive-OR immediate data to Acc	2
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3
CLR	A	Clear Accumulator	1
CPL	A	Complement Accumulator	1
RL	A	Rotate Accumulator Left	1
RLC	A	Rotate Accumulator Left through Carry	1
RR	A	Rotate Accumulator Right	1
RRC	A	Rotate Accumulator Right through the Carry	1
SWAP	A	Swap nibbles within the Accumulator	1

Mnemonic		Description	Byte
DATA TRANSFER			
MOV	A,R_n	Move Register to Accumulator	1
MOV	A,direct	Move direct byte to Accumulator	2
MOV	A,@Ri	Move indirect RAM to Accumulator	1
MOV	A,#data	Move immediate data to Accumulator	2
MOV	Rn,A	Move Accumulator to register	1
MOV	Rn,direct	Move direct byte to register	2
MOV	Rn,#data	Move immediate data to register	2
MOV	direct,A	Move Accumulator to direct byte	2
MOV	direct,Rn	Move register to direct byte	2
MOV	direct,direct	Move direct byte to direct	3
MOV	direct,@Ri	Move indirect RAM to direct byte	2
MOV	direct,#data	Move immediate data to direct byte	3
MOV	@Ri,A	Move Accumulator to indirect RAM	1
MOV	@Ri,direct	Move direct byte to indirect RAM	2
MOV	@Ri,#data	Move immediate data to indirect RAM	2
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to Acc	1
MOVC	A,@A+PC	Move Code byte relative to PC to Acc	1
MOVX	A,@R_i	Move external RAM(8-bit address) to Acc	1
MOVX	A,@DPTR	Move external RAM(16-bit address) to Acc	1
MOVX	@Ri,A	Move Acc to external RAM(8-bit address)	1
MOVX	@DPTR,A	Move Acc to external RAM(16-bit address)	1
PUSH	direct	Push direct byte onto stack	2
POP	direct	Pop direct byte from stack	2
XCH	A,R_n	Exchange Register with Accumulator	1
XCH	A,direct	Exchange direct byte with Accumulator	2
XCH	A,@R_i	Exchange indirect RAM with Accumulator	1
XCHD	A,@R_i	Exchange low order Digit indirect RAM with Accumulator	1

Mnemonic		Description	Byte
BOOLEAN VARIABLE MANIPULATION			
CLR	C	Clear Carry	1
CLR	bit	Clear direct bit	2
SETB	C	Set Carry	1
SETB	bit	Set direct bit	2
CPL	C	Complement Carry	1
CPL	bit	Complement direct bit	2
ANL	C,bit	AND direct bit to CARRY	2
ANL	C,/bit	AND complement of direct bit to Carry	2
ORL	C,bit	OR direct bit to Carry	2
ORL	C,/bit	OR complement of direct bit to Carry	2
MOV	C,bit	Move direct bit to Carry	2
MOV	bit,C	Move Carry to direct bit	2
JC	rel	Jump if Carry is set	2
JNC	rel	Jump if Carry not set	2
JB	bit,rel	Jump if direct Bit is set	3
JNB	bit,rel	Jump if direct Bit is Not set	3
JBC	bit,rel	Jump if direct Bit is set & clear bit	3

Mnemonic		Description	Byte
PROGRAM BRANCHING			
ACALL	addr11	Absolute Subroutine Call	2
LCALL	addr16	Long Subroutine Call	3
RET		Return from Subroutine	1
RETI		Return from interrupt	1
AJMP	addr11	Absolute Jump	2
LJMP	addr16	Long Jump	3
SJMP	rel	Short Jump (relative addr)	2
JMP	@A+DPTR	Jump indirect relative to the DPTR	1
JZ	rel	Jump if Accumulator is Zero	2
JNZ	rel	Jump if Accumulator is Not Zero	2
CJNE	A,direct,rel	Compare direct byte to Acc and Jump if Not Equal	3
CJNE	A,#data,rel	Compare immediate to Acc and Jump if Not Equal	3
CJNE	Rn,#data,rel	Compare immediate to register and Jump if Not Equal	3
CJNE	@Ri,#data,rel	Compare immediate to indirect and Jump if Not Equal	3
DJNZ	Rn,rel	Decrement register and Jump if Not Zero	2
DJNZ	direct,rel	Decrement direct byte and Jump if Not Zero	3
NOP		No Operation	1

2.2 8051 ADDRESSING MODES

What is an addressing mode ?

Addressing mode is a way to address an operand.

Operand means the data we are operating upon (in most cases source data). It can be a direct address of memory, it can be register names, it can be any numerical data etc.

For example,

MOV A,#2AH

Here, Operand= 2A

On execution of this instruction, the data 2AH is moved to accumulator A.

There are five different ways of executing this instruction. Thus, we can say that there are 5 addressing modes of 8051.

Types of Addressing Modes in 8051

1. Immediate addressing mode
2. Direct addressing mode
3. Register direct addressing mode
4. Register indirect addressing mode
5. Indexed addressing mode.

1. Immediate addressing mode

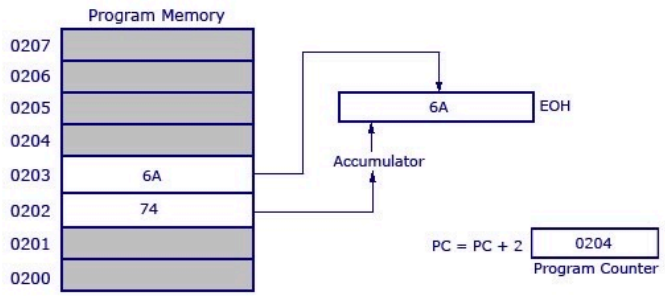
- In this Immediate Addressing Mode, the data is provided in the instruction itself.
- The data is provided immediately after the opcode.

Example: MOV A,#6AH

- The opcode for MOV A, # data is 74H.
- The opcode is saved in program memory at 0202 address.
- The data 6AH is saved in program memory 0203 (Any part of the program memory can be used, this is just an example).
- When the opcode 74H is read, the next step taken would be to transfer whatever data at the next program memory address (here at 0203) to accumulator A (E0H is the address of accumulator).
- This instruction is of two bytes and is executed in one cycle.
- So after the execution of this instruction, program counter will add 2 and move to 0204 of program memory.

Immediate Addressing Mode

Instruction	Opcode	Bytes	Cycles
MOV A, #6AH	74H	2	1



2. Direct Addressing Mode

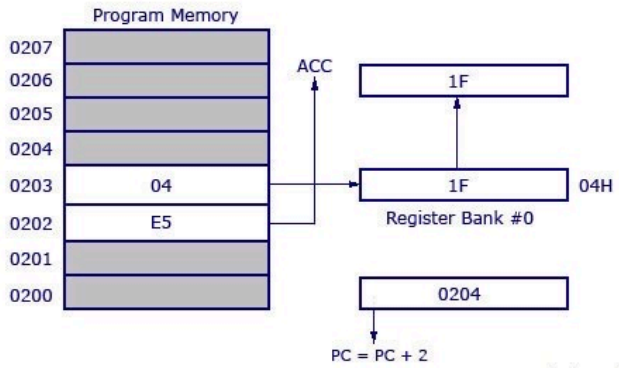
Here the address of the data (source data) is given as operand.

Example: MOV A,04H

- 04H is the address of register 4 of register bank#0.
- When this instruction is executed, whatever data is stored in register 04H is moved to accumulator.
- In the figure below, register 04H holds the data 1FH. So the data 1FH is moved to accumulator.
- This is a 2 byte instruction which requires 1 cycle to complete.
- Program counter will increment by 2 and stand in 0204.
- The opcode for instruction MOV A, address is E5H.
- When the instruction at 0202 is executed (E5H), accumulator is made active and ready to receive data.
- Then program control goes to next address that is 0203 and look up the address of the location (04H) where the source data (to be transferred to accumulator) is located.
- At 04H the control finds the data 1F and transfers it to accumulator and hence the execution is completed.

Direct Addressing Mode

Instruction	Opcode	Bytes	Cycles
MOV A, 04H	E5	2	1



3. Register Direct Addressing Mode

In this addressing mode we use the register name directly as source operand.

Example: MOV A,R4

- There are 32 word registers- 8 in each of the four Register Banks named 0,1,2 and 3.
- Each bank has 8 registers named from R0 to R7.
- At a time only one register bank can be selected.
- Selection of register bank is made possible through a Special Function Register (SFR) named Processor Status Word (PSW).
- PSW is an 8 bit SFR where each bit can be programmed.
- Bits are designated from PSW.0 to PSW.7
- Register banks are selected using PSW.3 and PSW.4 These two bits are known as register bank select bits as they are used to select register banks.

Processor Status Word

PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	PSW.0
CY	AC	F0	RS1	RS0	0V		P

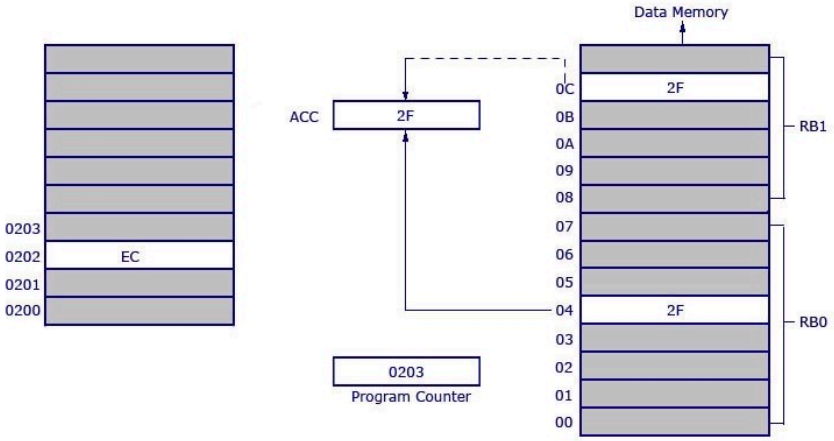
Register bank Select bit 0

Register bank Select bit 1

RS1	RS0	Register Bank	Register Bank Status
0	0	0	Register Bank 0 is selected
0	1	1	Register Bank 1 is selected
1	0	2	Register Bank 2 is selected
1	1	3	Register Bank 3 is selected

Register Direct Addressing Mode

Instruction	Opcode	Bytes	Cycles
MOV A, R4	ECH	1	1



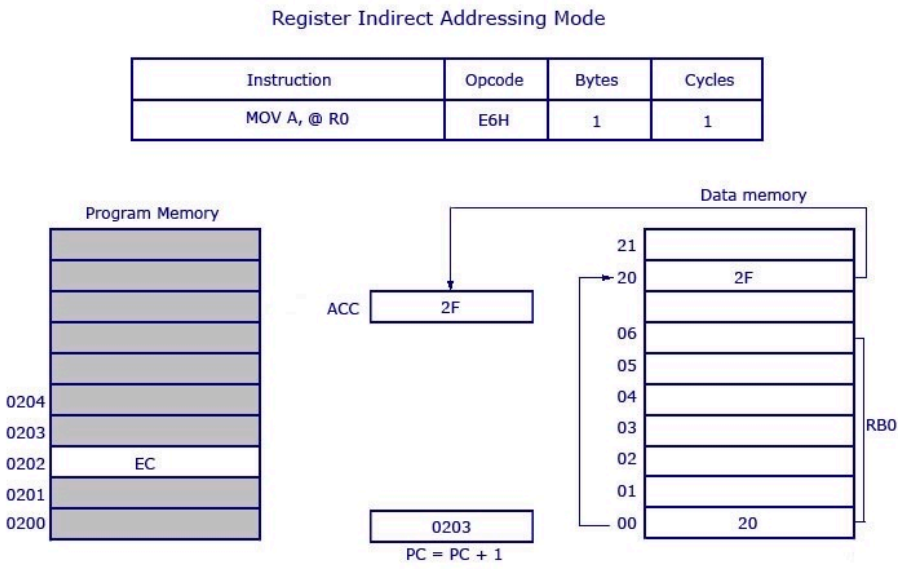
- In register direct addressing mode, data is transferred to accumulator from the register (based on which register bank is selected).
- Opcode for MOV A, R4 is EC.
- The opcode is stored in program memory address 0202 and when it is executed the control goes directly to R4 of the respected register bank (that is selected in PSW).
- If register bank #0 is selected then the data from R4 of register bank #0 will be moved to accumulator (here it is 2F stored at 04 H).
- 04 H is the address of R4 of register bank #0.
- Movement of data (2F) in this case is shown as bold line.
- The dotted line shows that 2F is getting transferred to accumulator from data memory location 0C H.
- Here, 0C H is the address location of Register 4 (R4) of register bank #1.
- The instruction above is 1 byte and requires 1 cycle for complete execution.
- This means using register direct addressing mode can save program memory.

4. Register Indirect Addressing Mode

In this addressing mode, address of the data (source data to transfer) is given in the register operand.

Example: MOV A, @R0

- Here the value inside R0 is considered as an address, which holds the data to be transferred to accumulator.
- If R0 holds the value 20H, and we have a data 2F H stored at the address 20H, then the value 2FH will get transferred to accumulator after executing this instruction.



- The opcode for MOV A,@R0 is E6H.
- Assuming that register bank #0 is selected. So the R0 of register bank #0 holds the data 20H.
- Program control moves to 20H where it locates the data 2FH and it transfers 2FH to accumulator.
- This is a single byte instruction and the program counter increments 1 and moves to 0203 of program memory.
- Note: Only R0 and R1 are allowed to form a register indirect addressing instruction.

5. Indexed Addressing Mode

In the indexed addressing mode, the source memory can only be accessed from program memory only.

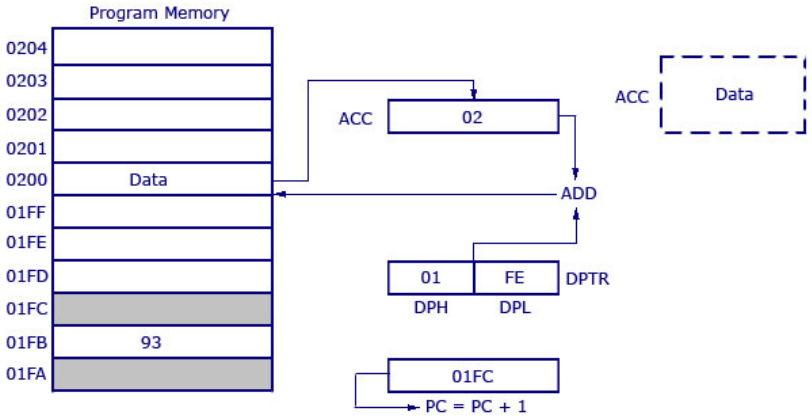
The destination operand is always the register A.

Example: MOVC A, @A+DPTR

- The source operand is @A+DPTR and we will get the source data (to transfer) from this location.
- Here we are adding contents of DPTR with present content of accumulator.
- This addition will result a new data which is taken as the address of source data (to transfer).
- The data at this address is then transferred to accumulator.
- The opcode for the instruction is 93H.

Indexed Addressing Mode

Instruction	Opcode	Bytes	Cycles
MOVC A,@A +DPTR	93H	1	2



- DPTR holds the value 01FE, where 01 is located in DPH (higher 8 bits) and FE is located in DPL (lower 8 bits).
- Accumulator now has the value 02H.
- A 16 bit addition is performed and now 01FE H+02 H results in 0200 H.
- Whatever data is in 0200 H will get transferred to accumulator.
- The previous value inside accumulator (02H) will get replaced with new data from 0200H.
- New data in the accumulator is shown in dotted line box.
- This is a 1 byte instruction with 2 cycles needed for execution.
- The execution time required for this instruction is high compared to previous instructions

Another Example: MOVC A, @A+PC

It works the same way as above example. The only difference is, instead of adding DPTR with accumulator, here data inside program counter (PC) is added with accumulator to obtain the target address.

6. Implied Addressing Mode

- In the implied addressing mode, there will be a single operand.
- These types of instruction can work on specific registers only.
- These types of instructions are also known as register specific instruction.

Examples: RL A
SWAP A

- These are 1- byte instruction.
- The first one is used to rotate the A register content to the Left.
- The second one is used to swap the nibbles in A.

Prepared by
Shweta Sharma