e-PG Pathshala

Subject : Computer Science

Paper: Embedded System
Module: Instruction set
Module No: CS/ES/6

Quadrant 1 – e-text

In this lecture different Addressing mode in 8051 instruction set will be discussed. Instruction set of 8051 will be visited. Data transfer instructions in the instruction set will be discussed in detail with examples.

1.1 Addressing modes

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. we can explain this with a simple data move instruction of 8051.

MOV A,#4AH

Here the data 4A is the operand, often known as source data. When this instruction is executed, the data 4AH is moved to accumulator A. There are 5 different ways to execute this instruction and hence we say, we have got 5 addressing modes for 8051. They are

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

1.1.1 Immediate Addressing Mode.

In general we can write MOV A, #data

This addressing mode is named as ***"immediate"*** because it transfers an 8-bit data immediately to the accumulator (destination operand). An example is given below

MOV A, #6AH

Immediate Addressing Mode

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

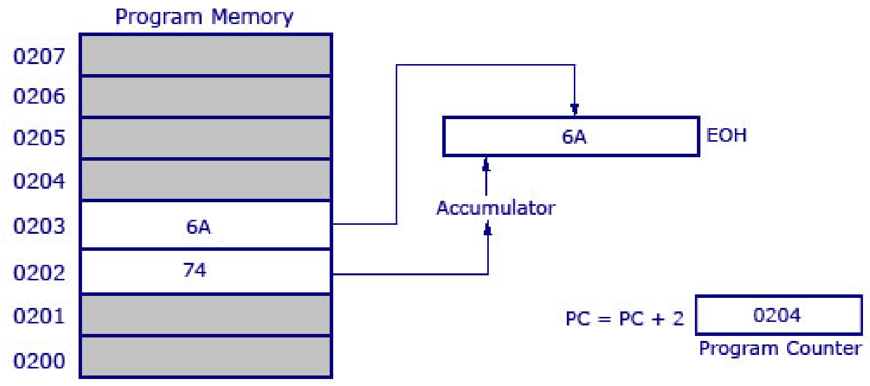


Figure 1.1 Immediate Addressing Modes

The picture above describes the above instruction and its execution. 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. (See, 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 o204 of program memory.The **'#'** symbol before 6AH indicates that operand is a data (8 bit). If **'#'** is not present then the hexadecimal number would be taken as address.

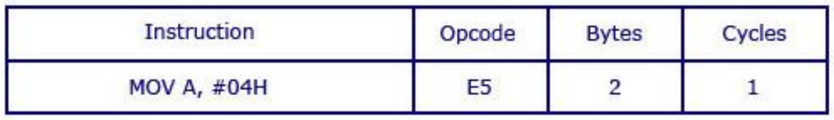
1.1.2 Direct Addressing Mode

This is another way of addressing an operand. Here the address of the data (source data) is given as operand. An example is given below.

MOV A, 04H

Here 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 picture below we can see, register 04H holds the data 1FH. So the data 1FH is moved to accumulator.

Direct Addressing Mode



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

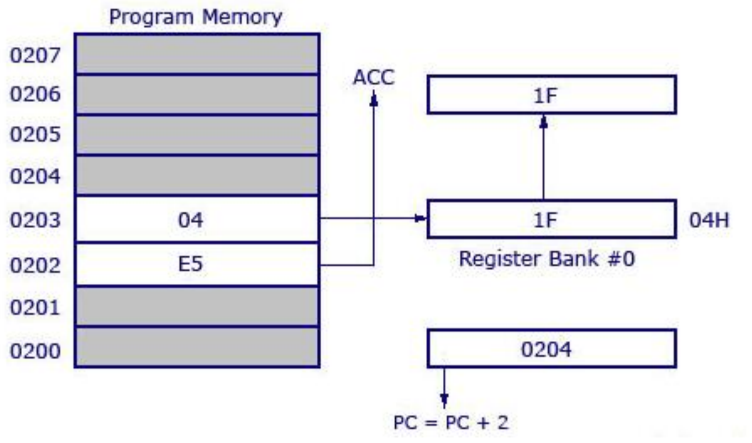


Figure 1.2 Direct Addressing modes

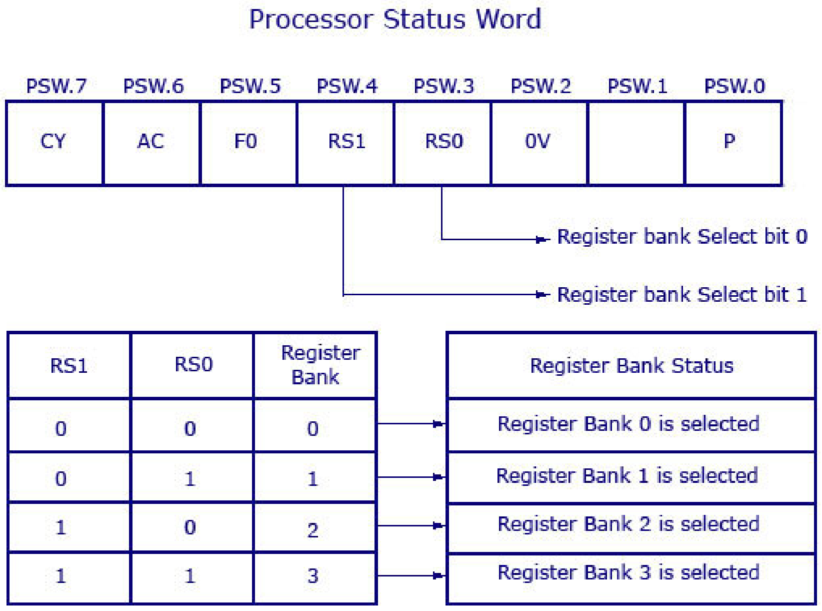
As shown in picture above this is a 2 byte instruction which requires 1 cycle to complete. Program counter will increment by 2 and stand in 0204. The opcode for instruction MOV A, address is E5H. When the instruction at 0202 is executed (E5H), accumulator is made active and ready to receive data. Then 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.

1.1.3 Register Direct Addressing Mode

In this addressing mode the register name directly (as source operand) used. An example is shown below.

MOV A, R4

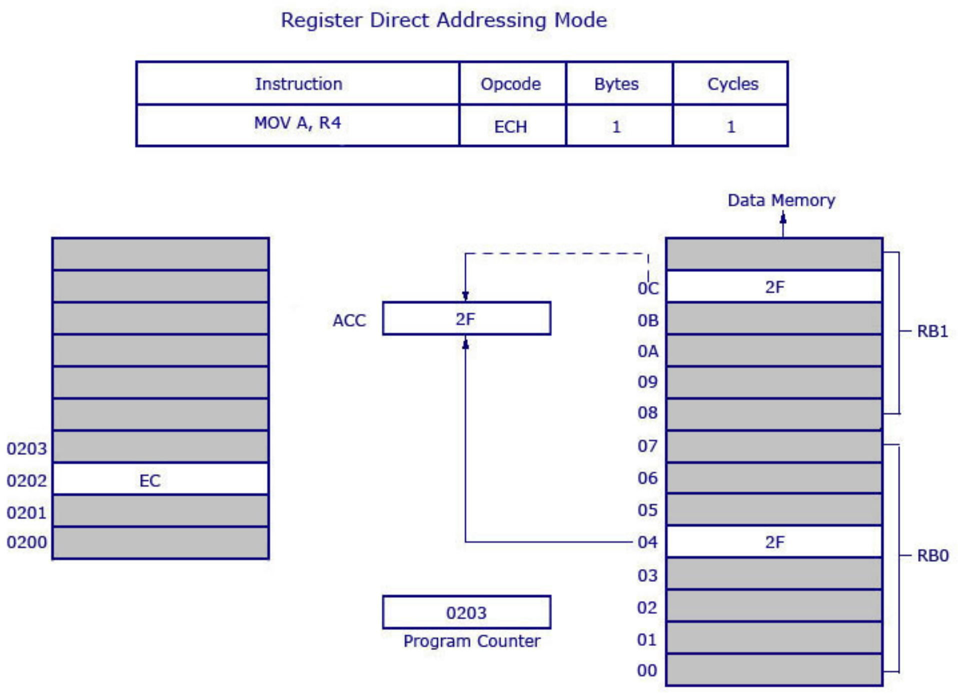
At a time registers can take value from R0,R1…to R7.There are 32 registers. There are 4 register banks named 0,1,2 and 3. Each bank has 8 registers named from R0 to R7. At a time only one register bank can be selected. Selection of register bank is made possible through a Special Function Register (SFR) 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. A picture below shows the PSW register and the Register Bank Select bits with status.



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

Figure 1.3 Program Status word

Hence in register direct addressing mode, data is transferred to accumulator from the register (based on which register bank is selected).



Instruction	Opcode	Bytes	Cycles
MOV A, R4	ECH	1	1

Figure 1.4 Register Direct Addressing modes

Since we see that 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. Now please take a look at the dotted line. Here 2F is getting transferred to accumulator from data memory location 0C H. Now understand that 0C H is the address location of Register 4 (R4) of register bank #1. Programmers usually get confused with register bank selection. Also keep in mind that data at R4 of register bank #0 and register bank #1 (or even other banks) will not be same. So wrong selection of register banks will result in undesired output. Also note that the instruction above is 1 byte and requires 1 cycle for complete execution. This means using register direct addressing mode can save program memory.

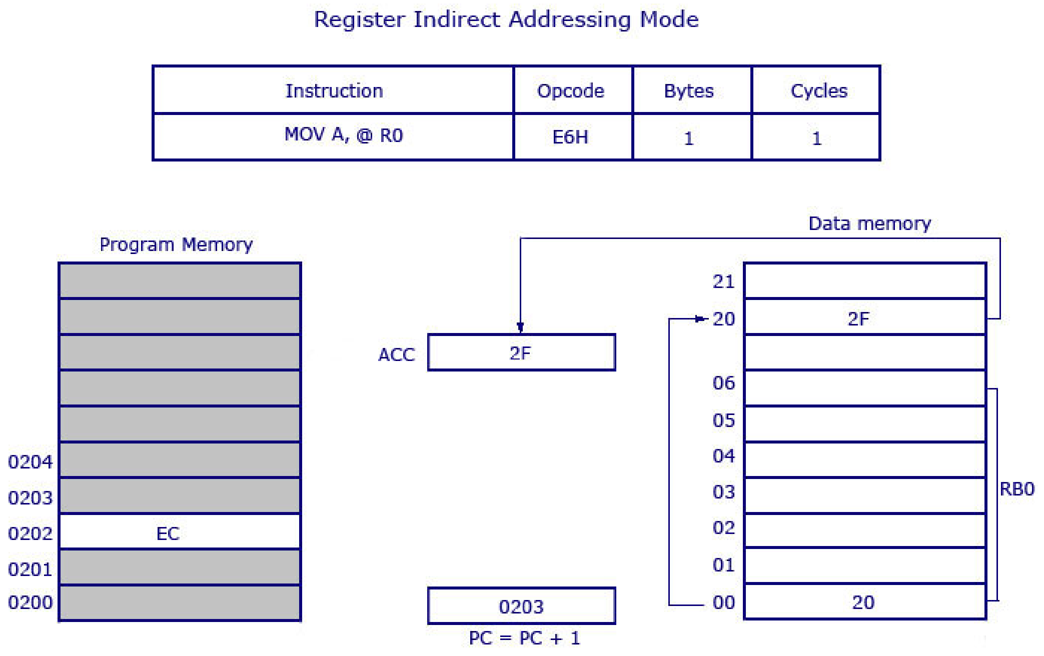
1.1.4 Register Indirect Addressing Mode

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

MOV A, @R0

Here the value inside R0 is considered as an address, which holds the data to be transferred to accumulator.

Example: 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 picture is shown below.


Register Indirect Addressing Mode

Instruction	Opcode	Bytes	Cycles
MOV A, @ R0	E6H	1	1

Figure 1.5 Register Indirect Addressing modes

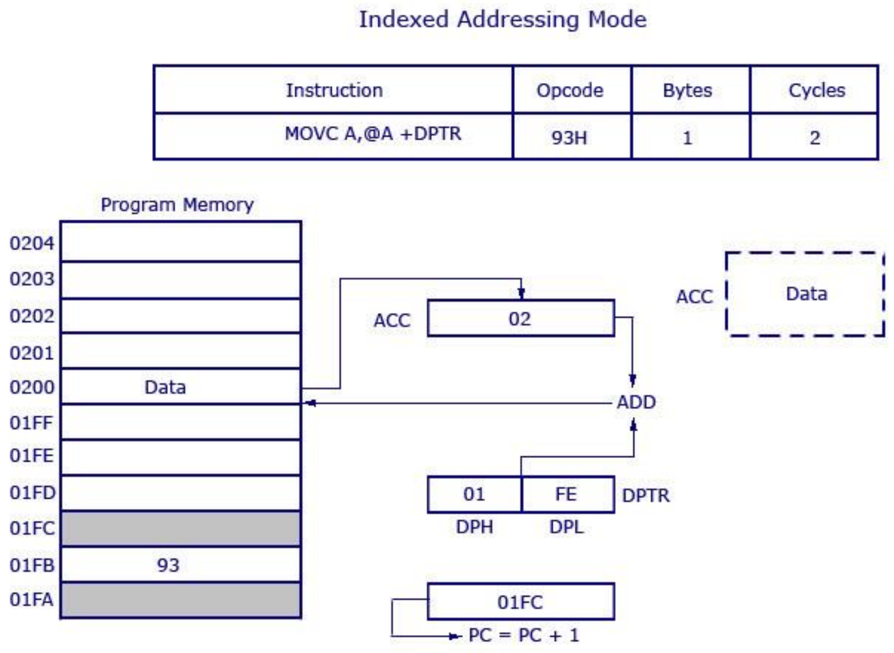
Since the opcode for **MOV A, @R0** is E6H. Assuming that register bank #0 is selected. So the R0 of register bank #0 holds the data 20H. Program control moves to 20H where it locates the data 2FH and it transfers 2FH to accumulator. This is a single byte instruction and the program counter increments 1 and moves to 0203 of program memory.

1.1.5 Indexed Addressing Mode

MOVC A, @A+DPTR and MOVC A, @A+PC
where DPTR is data pointer and PC is program counter (both are 16 bit registers).

MOVC A, @A+DPTR

The source operand is @A+DPTR and we know we will get the source data (to transfer) from this location. It is nothing but 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. Take a look at the picture below.



Indexed Addressing Mode

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

Figure 1.5 Indexed Addressing modes

The opcode for the instruction is 93H. 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 (which all were 1 cycle).

The other example **MOVC A, @A+PC** 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.

Representing addressing modes in program is shown in following table

R_n	Register R7-R0 of the currently selected bank
direct	8- bit Internal Data location's address. This could be an Internal DAta RAM location(0-127) or a SFR[i.e I/O port, control register, status register, etc.(128-255)]
@R_i	8-bit Internal data RAM location(0-255) addressed indirectly through register R1 or R0.
#data	8-bit constant included in instruction.
#data 16	16-bit constant included in instruction.
Addr 16	16-bit destination address. Used by LCALL and LJMP. A branch can be anywhere within the 64K byte Program Memory address space.

Table .1 Register Addressing modes

In table.1 'R' indicates the register , @R indicates the address stored in register and # represents it is data.

2. Instruction Types

The 8051 instruction set has different type of instructions based on their operation. They are Data transfer, Arithmetic, Logical, Boolean, and Branching instructions. In this section all kind of data transfer instruction will be discussed.

2.1 Data Transfer Instructions

Data transfer instructions move the content of one register to another. The register the content of which is moved remains unchanged. If they have the suffix "X" (MOVX), the data is exchanged with external memory.

DATA TRANSFER INSTRUCTIONS			
Mnemonic	**Description**	**Byte**	**Cycle**
MOV A,Rn	Moves the register to the accumulator	1	1
MOV A,direct	Moves the direct byte to the accumulator	2	2
MOV A,@Ri	Moves the indirect RAM to the accumulator	1	2
MOV A,#data	Moves the immediate data to the accumulator	2	2

MOV Rn,A	Moves the accumulator to the register	1	2
MOV Rn,direct	Moves the direct byte to the register	2	4
MOV Rn,#data	Moves the immediate data to the register	2	2
MOV direct,A	Moves the accumulator to the direct byte	2	3
MOV direct,Rn	Moves the register to the direct byte	2	3
MOV direct,direct	Moves the direct byte to the direct byte	3	4
MOV direct,@Ri	Moves the indirect RAM to the direct byte	2	4
MOV direct,#data	Moves the immediate data to the direct byte	3	3
MOV @Ri,A	Moves the accumulator to the indirect RAM	1	3
MOV @Ri,direct	Moves the direct byte to the indirect RAM	2	5
MOV @Ri,#data	Moves the immediate data to the indirect RAM	2	3
MOV DPTR,#data	Moves a 16-bit data to the data pointer	3	3
MOVC A,@A+DPTR	Moves the code byte relative to the DPTR to the accumulator (address=A+DPTR)	1	3
MOVC A,@A+PC	Moves the code byte relative to the PC to the accumulator (address=A+PC)	1	3
MOVX A,@Ri	Moves the external RAM (8-bit address) to the accumulator	1	3-10
MOVX A,@DPTR	Moves the external RAM (16-bit address) to the accumulator	1	3-10
MOVX @Ri,A	Moves the accumulator to the external RAM (8-bit address)	1	4-11

MOVX @DPTR,A	Moves the accumulator to the external RAM (16-bit address)	1	4-11
PUSH direct	Pushes the direct byte onto the stack	2	4
POP direct	Pops the direct byte from the stack/td>	2	3
XCH A,Rn	Exchanges the register with the accumulator	1	2
XCH A,direct	Exchanges the direct byte with the accumulator	2	3
XCH A,@Ri	Exchanges the indirect RAM with the accumulator	1	3
XCHD A,@Ri	Exchanges the low-order nibble indirect RAM with the accumulator	1	3

Table. 2 Data Transfer Instruction

The MOV <dest>, <src> instruction allows data to be transferred between any two internal RAM or SFR locations without going through the Accumulator. PUSH and POP operation belongs to stack register operation. push will insert data into stack top and POP will fetch data from stack top. XCH helps in swapping data between accumulator and other register/memory location. XCHD helps to swap lower nibbles in accumulator and another memory location. DPTR is a 16 bit register has 2 8 bit DPL and DPH registers . These two will operates as a register bank and always hold 16 bit address only not any other data.

Address width	**Mnemonic**	**Operation**	**Execution Time @12MHz(μs)**
8 bits	MOVX A,@R_i	Read external RAM @R_i	2
8 bits	MOVX @R_i,A	Write external RAM @R_i	2
16 bits	MOVX A,@DPTR	Read external RAM @DPTR	2
16 bits	MOVX @DPTR,A	Write external RAM@DPTR	2

Table. 3 Execution Time

Table .3 discuss about moving data from external memory. MOVX indicates data at external memory. @DPTR represents 16 bit address and @Ri represents 8 bit address

Mnemonic	**Operation**	**Execution Time @ 12 MHz(μs)**
MOVC A,@A+DPTR	Read Pgm Memory at (A+DPTR)	2
MOVC A,@A+PC	Read Pgm Memory at (A+PC)	2

Table.4 Execution Time

MOVC represents program code movement from external memory. DPTR will hold the 16 bitexternal memory address.MOVX and MOVC instructions are used for accessing external memory. Whereas MOV instruction used for accessing internal memory.

The following code segments will demo a different addressing format data transfer in a detailed way.

MOV <dst>,<src>

Now assume location 30h has 59

```
MOV R0,#30H ; R0 < = 30H
MOV A,@R0 ;    A < = 59
MOV R1,A ;       R1 < = 59
```

at the end R1 is having holding data 59.

The next code segment demo the external memory data transfer

```
MOVC A,@A+DPTR ;        (A) ← ((A) + (DPTR))
  MOVX A,@DPTR ;           (A) ← ((DPTR))
```

The tables gives details of data transfer and different addressing and time taken for executing these instructions. Immediate addressing takes less time and indirect and external memory accessing takes more time to process.

3. Summary

In this lecture different Addressing formats are discussed. Data transfer instructions of 8051 instructions are discussed with examples. Time consumption by the different addressing formats also shown**.**

4. References

1. The 8051 Microcontroller and Embedded Systems Using Assembly and C Second Edition Muhammad Ali Mazidi, Janice Gillispie Mazidi and Rolin D. McKinlay.
2. Atmel 8051 manual
3. http://www.circuitstoday.com
4. http://www.mikroe.com

*********** Pictures taken from www.circuitstoday.com***********