# 8051 Microcontroller Moving Data

Prepared by : A. B. Modi

Target Audience : 5th Semester Students

# Learning Objectives

After learning this chapter, Students should be able to:

- Use different commands for data manipulation
- Describe Data Addressing Modes
- Describe the execution of POP and PUSH instructions
- Write simple data movement programs

# Introduction

Data is stored in memory location.

Data is stored at a source address and moved to a destination address.

The ways by which this addresses are specified are called addressing modes.

Three types of instruction:

*MOV  destination, source*
*PUSH  source or POP destination*
*XCH  destination, source*

# Addressing Modes

1. Immediate addressing mode
2. Register addressing mode
3. Direct addressing mode
4. Indirect addressing mode

# Immediate Addressing Mode

- Instruction using #
- The program counter is automatically incremented to point to the byte(s) following the opcode byte in the program memory.

```
MOV  a, #n          ; copy the immediate data byte n to the A register
MOV  Rr,#n          ; copy the immediate data byte n to register Rr
MOV  DPTR,#nn       ; copy the immediate 16-bit number nn to the DPTR register
```

# Register Addressing Mode

- Register A, DPTR, R0-R7
- The registers used in the opcode as R0 - R7 are the ones that are currently chosen by the bank-select bits, RS0 and RS1 in PSW.

```
MOV  A, Rr     ;copy data from register Rr to register A
MOV  Rr, A     ; copy data from register A to Register Rr
```

# Direct Addressing Mode

- 128 bytes of Internal RAM and the SFRs may be addressed directly using the single byte address assigned to each RAM location and each SFR.

```
MOV 90h, #0a5h  ;move a constant number into port 1
MOV p1, #0a5h   ;move a constant number into port 1
```

*MOV A, add ;copy data from direct address to register A*

*MOV add, A ;copy data from register A to direct address*

*MOV Rr, add ;copy data from direct address to register Rr*

*MOV add, Rr ;copy data from register Rr to register A*

*MOV add, #n ;copy immediate data n to direct address*

*MOV add1, add2 ;copy data from direct address 1 to direct address 2*

*MOV A, 80h ;copy data from the port 0 pins to register A*

*MOV 80h, A ;copy data from register A to the port 0 latch*

*MOV 3Ah, #3Ah ;copy immediate data byte 3Ah to RAM location 3Ah*

*MOV Rr,12h ;copy data from RAM location 12h to register R0*

*MOV 8Ch, R7 ;copy data from register R7 to timer 0 high byte*

*MOV 5ch, A ;copy data from register A to RAM location 5Ch*

*MOV 0A8h, 77h ;copy data from RAM location 77h to IE register*

| SFR | Address (Hex) |
|---|---|
| A | 0E0 |
| B | 0F0 |
| DPL | 82 |
| DPH | 83 |
| IE | 0A8 |
| IP | 0B8 |
| P0 | 80 |
| P1 | 90 |
| P2 | 0A0 |
| P3 | 0B0 |

| SFR | Address (Hex) |
|---|---|
| PCON | 87 |
| PSW | 0D0 |
| SBUF | 99 |
| SCON | 98 |
| SP | 81 |
| TCON | 88 |
| TMOD | 89 |
| TH0 | 8C |
| TL0 | 8A |
| TH1 | 8D |
| TL1 | 8B |

| RS1=0, RS0=0 | | RS1=0, RS0=1 | | RS1=1, RS0=0 | | RS1=1, RS0=1 | |
|---|---|---|---|---|---|---|---|
| BANK 0 | | BANK 1 | | BANK 2 | | BANK 3 | |
| 0 | R0 | 8 | R0 | 10 | R0 | 18 | R0 |
| 1 | R1 | 9 | R1 | 11 | R1 | 19 | R1 |
| 2 | R2 | A | R2 | 12 | R2 | 1A | R2 |
| 3 | R3 | B | R3 | 13 | R3 | 1B | R3 |
| 4 | R4 | C | R4 | 14 | R4 | 1C | R4 |
| 5 | R5 | D | R5 | 15 | R5 | 1D | R5 |
| 6 | R6 | E | R6 | 16 | R6 | 1E | R6 |
| 7 | R7 | F | R7 | 17 | R7 | 1F | R7 |

# Indirect Addressing Mode

- Instruction using @
- The actual address is going to be used in data manipulation is stored in a register.
- Register itself is not the address, but data stored in the register is the address.
- Register R0 and R1, often called data pointer, used to store data locations of RAM (00h - 7Fh).

```
MOV @Rp, #n      ;copy the immediate byte n to the address in Rp
MOV @Rp, add     ;copy the contents of add to the address in Rp
MOV @Rp, A       ;copy the data in A to the address in Rp
MOV add, @Rp     ;copy the contents of the address in Rp to add
MOV A,@Rp        ;copy the contents of the address in Rp to A
```

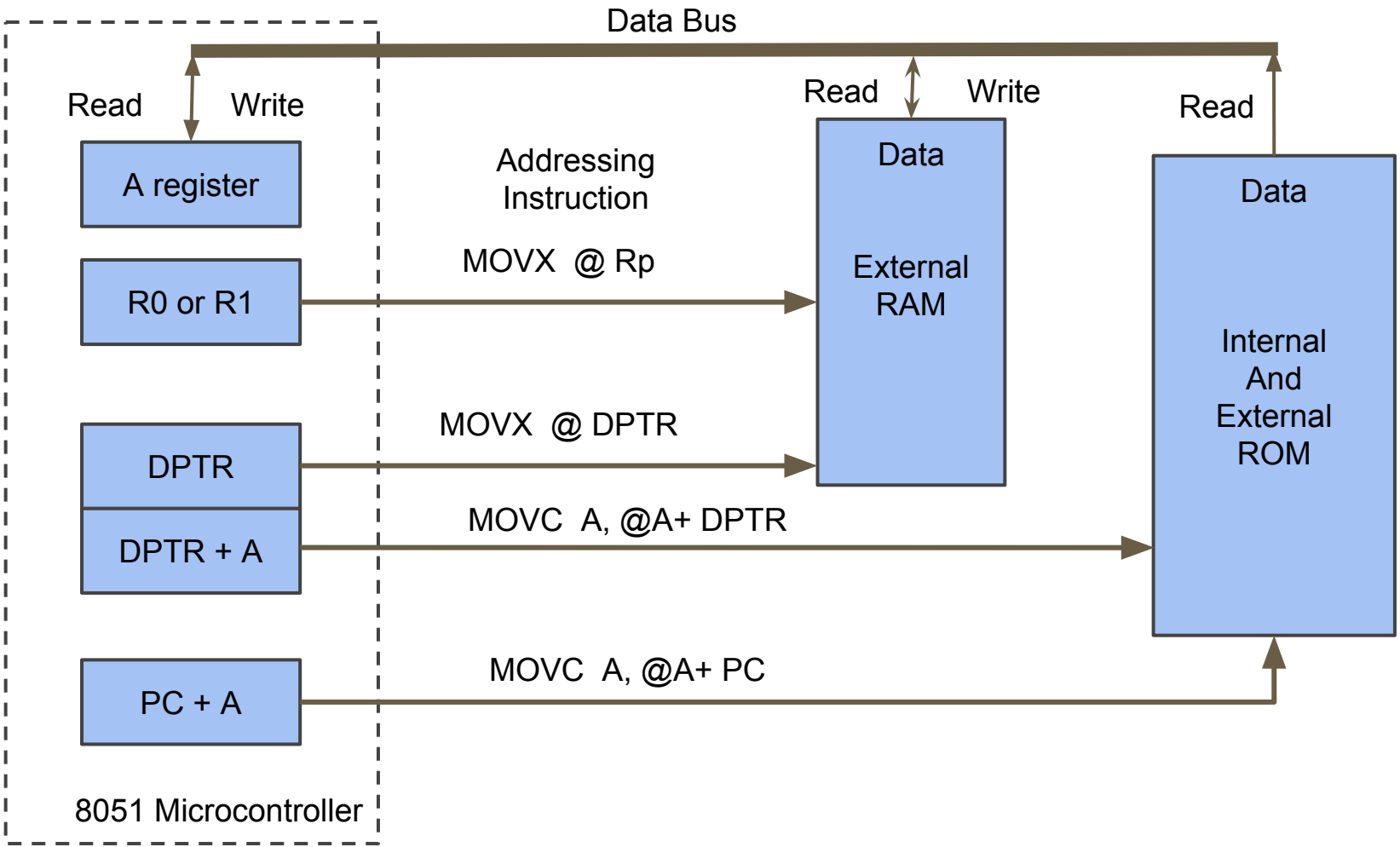
```
MOV @R1, #35h   ;copy the immediate byte 35h to the address in R1
MOV @R0, 80h    ;copy the contents of 80h to the address in R0
MOV @R1, A      ;copy the data in A to the address in R1
MOV 80h, @R0    ;copy the contents of the address in R0 to 80h
MOV A,@R0       ;copy the contents of the address in R0 to A
```

***Note:***

1. the number in register Rp must be a RAM address.
2. Only register R0 and R1 may be used for indirect addressing.

# External Data Moves

- 8051 microcontroller RAM and ROM memory space can be expanded by adding external memory chips.
- These external memory addresses are always accessed by using Indirect Addressing Mode.
- R0, R1, DPTR are used to store address of external RAM.
- R0, R1 can store memory addresses up to 00h to FFh.
- DPTR can address the maximum RAM space of 0000h to FFFFh.


Data Bus
Read
Write
A register
R0 or R1
DPTR
DPTR + A
PC + A
8051 Microcontroller
Addressing
Instruction
MOVX @ Rp
MOVX @ DPTR
MOVC A, @A+ DPTR
MOVC A, @A+ PC
Read
Write
Data
External
RAM
Read
Data
Internal
And
External
ROM

```
MOVX A, @Rp       ;copy the contents of the external address in Rp to A
MOVX A, @DPTR     ;copy the contents of the external address in DPTR to A
MOVX @Rp, A       ;copy the contents of A to the external address in Rp
MOVX @DPTR, A     ;copy the contents of A to the external address in DPTR
```

Note :

All external data moves must involve the A register.
MOVX is normally used with external RAM or I/O addresses.
There are two sets of RAM addresses between 00h and FFh : one internal and one external.

# Code Memory Read-Only Data Moves

- When access to a preprogrammed mass of data is needed, such as data table. This data must be permanent to be of repeated use and is stored in the program ROM.
- Data is accessed by using Indirect Addressing Mode, using registers A, PC and DPTR.
- Pointing address  = pointing register (PC or DPTR) + number in register A.
- The data fetched is fetched from pointing address and it is placed in A.

*MOVC A, @A+DPTR* *;copy the code byte, found at the ROM address formed by adding A and the DPTR, to A*

*MOVC A, @A+PC* *;copy the code byte, found at the ROM address formed by adding A and the PC, to A*

Example:

```
MOV DPTR, #1234h    ;DPTR     = 1234h
MOV A, #56h         ;A        =     56h
MOVC A,@A+DPTR;copy the contents of address 128Ah to A
MOVC A, @A+PC       ; if PC = 4000h, A = 56h then
                    ; copy the contents of address 4057h to A
```

Note :

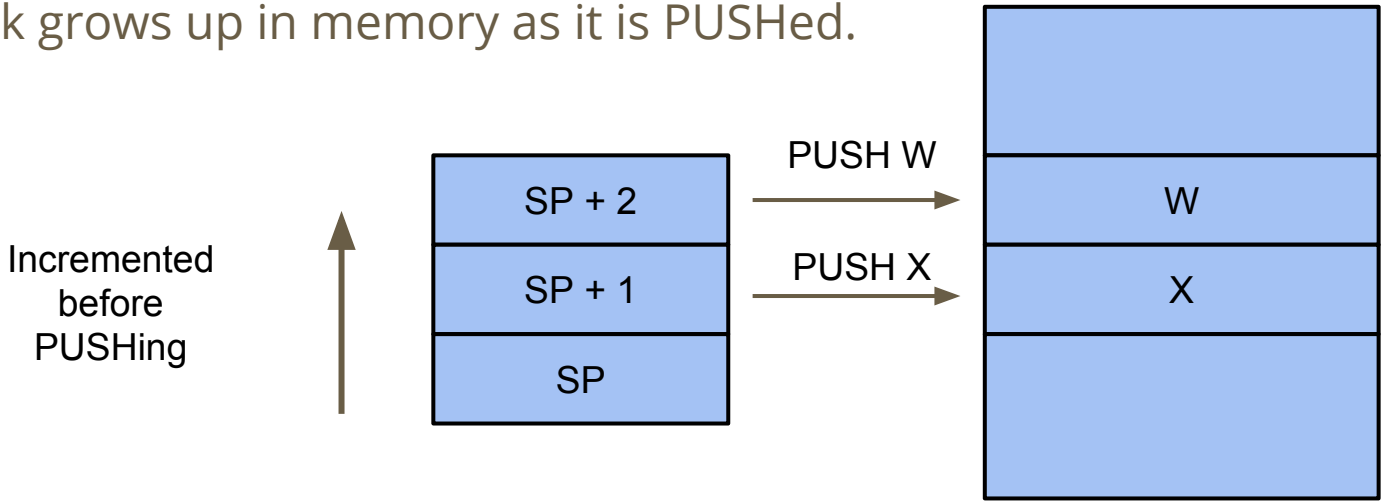
MOVC is normally used with internal and external ROM.

# PUSH and POP opcodes

- Data moves between an area of internal RAM, known as STACK, and the specified direct address.
- The stack pointer (SP) is special function register.
- SP holds the indirect address whenever PUSHing or POPing.
- The SP register is set to 07h, when the 8051 microcontroller is reset.

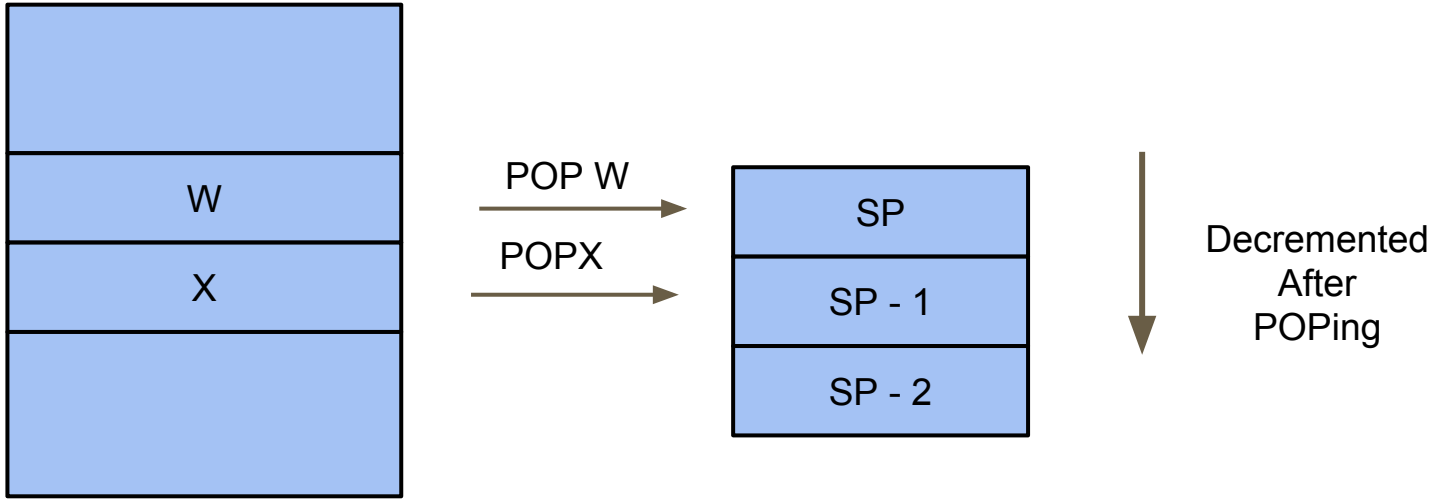
# PUSH opcode

- PUSH opcode copies data from the source address to the stack. SP is incremented by 1 before the data is copied to the internal RAM location contained in SP so that the data is stored from low addresses to high addresses in the internal RAM.
- Stack grows up in memory as it is PUSHed.



# POP opcode

- POP opcode copies data from the stack to the destination address. SP is decremented by 1 after data is copied from the stack RAM address to the direct destination to make sure that data placed on the stack is retrieved in the same order as it was placed.



*PUSH add* *; Increment SP and then copy the data in add to the internal RAM address contained in SP*

*POP add* *;Copy the data from the internal RAM address contained in SP to Add and then decrement the SP*

Stack Pointer (SP) is set to 07h when 8051 is reset.

SP value ranges from 00h-FFh, but is limited by Internal RAM Range of 7Fh.

# Data Exchanges

- Exchange instruction moves data in two directions:
  - From source to destination
  - From destination to source
- All addressing modes except immediate may be used in XCH instructions.

```
XCH A, Rr       ;exchange data bytes between A and Register Rr.
XCH A, add      ;exchange data bytes between add and A.
XCH A, @Rr      ;exchange data bytes between A and address in Rr.
XCHD A, @Rp     ;exchange lower nibble between A and address in Rr, upper
                 nibbles remains unaffected.
```

*Upper nibble of A and upper nibble of the address location in Rr do not change.*

# Now try these examples.

1. Copy the byte in TCON to register R2 using at least 4 different methods.
2. Set timer T0 to an initial setting of 1234h. Use direct address with an immediate number to set TH0 and TL0.
3. Put the number 38h in R5, R6 and R7.
4. Put the number 8Eh in RAM locations 30h to 34h.
5. Copy the contents of DPTR to register R0(DPL) and R1(DPH).
6. Rotate the bytes in registers R0 to R3: Copy the data in R0 to R1, R1 to R2, R2 to R3 and R3 to R0.
7. Store the contents of RAM location 20h at the address contained in 08h.
8. Store register A at the internal RAM location address in register A.