3

Moving Data

★ Introduction

⇒ A Computer typically Spends more time moving data from one location to another than it Spends on any other operation.

⇒ In 8051, mnemonics are written ~~first~~ with the destination address named first, followed by the Source address.

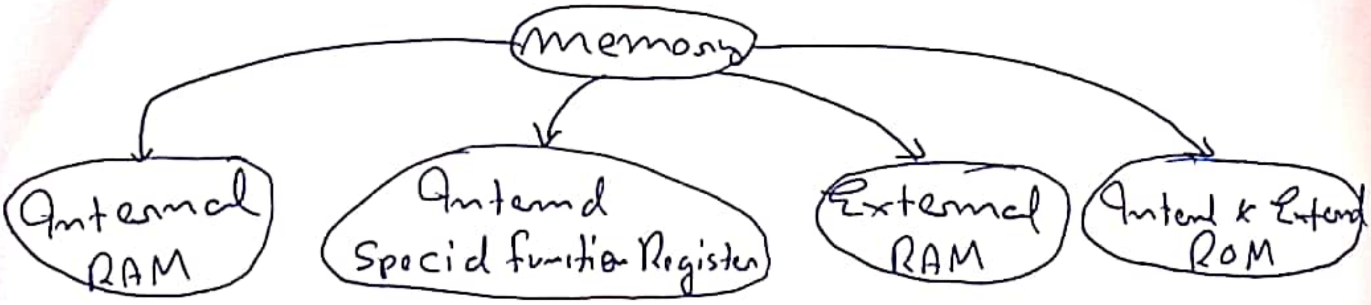
⇒ There are 28 distinct mnemonics that Copy data from a Source to a destination, they may be divided into the following three main type.

1. MOV destination, source
2. PUSH source or POP destination
3. XCH destination, Source.

⇒ The following four address modes are used to access data:-

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

⇒ The MOV opCodes involve data transfers within the 8051 memory.



⇒ Findly, the following five types of opcodes are used to move data:-

1. MOV
2. MOVX
3. MOVC
4. PUSH and POP
5. XCH

★ Addressing Modes

1. Immediate Addressing Mode

OpCode(#n)	Next Byte(s)	Source Only
Instruction	Data	

2. Register Addressing Mode

⇒ Certain register names may be used as part of the opcode mnemonic as source or destinations of data.
{A, DPTR, R0 to R7}

Example:-

MOV A, #n // Copy the immediate data byte n to the A register.

Note

It is impossible to have immediate data as a destination.

All numbers must start with a decimal number (0-9), or the assembler assumes the number is a label.

Register to register moves using the register addressing mode occur between register A and R0 to R7.

3. Direct Addressing Mode

⇒ All 128 bytes of internal RAM and the SFRs may be addressed directly using the single-byte address assigned to each RAM location and each special function register.

Example

	Address (HEX)
A ———	0E0
B ———	0F0
DPL ———	82
DPL ———	83

4. Indirect Addressing Mode

⇒ Indirect addressing for MOV opcode uses register R0 or R1, often called "data pointer" to hold the address of one of the data locations which could be RAM or an SFR address.

⇒ The mnemonic symbol used for indirect addressing is the "at" sign, which is printed as @.

Example

MOV @RP, #n // Copy the immediate byte n to the address in RP.

Note

The number in register Rp must be a RAM or an SFR address.

Only registers R0 or R1 may be used for indirect addressing.

★ Extemal Data Moves

⇒ The extemal memory can be as large as 64K bytes for each of the RAM and ROM memory areas.

⇒ Opcodes that access this extemal memory always use indirect addressing to specify the extemal memory.

⇒ An X is added to the MOV mnemonics to serve as a reminder that the data move is extemal to the 8051, as shown in the following table.

Example

MOVX A,@Rp // Copy the Content of extemal address in Rp to A.

Note

There are two set of RAM address between 00 and 0FFh: One intemal and one extemal to the 8051.

★ Code Memory Read-Only Data Moves

⇒ Access to this data is made possible by using indirect addressing and the A register in conjunction with either the PC or the DPTR.

⇒ The letter C is added to the MOV mnemonic to highlight the use of the opcodes for moving data from the source address in the Code ROM to the A register in the 8051.

Example

MOVC A,@A+PC // Copy the code byte, found at the ROM address formed by adding A and the PC to A.

★ PUSH and POP opcode

⇒ The data moves between an area of internal RAM, Known as stack, and the specified direct address.

⇒ The Stack pointer - SFR Contains the address in RAM where data from the source address will be PUSHed, or where data to be POPed to the destination address is found.

PUSH add //Increment SP; 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; decrease the SP.

Note

When the SP reaches FFh it "rolls over" to 00h (R0).

★ Data Exchange {→All exchange use register A}

⇒ MOV, PUSH and POP opcodes all involve copying the data found in the source address to the ~~ad~~ destination address; the original data in the source is not changed.

⇒ Exchange instructions actually move data in two directions; from source to destination and from destination to source.

⇒ All addressing mode except immediate may be used in the XCH (exchange) opcode.

Example

XCH A, Rr // Exchange data bytes between register Rr and A.

★ Example Programs

Unsolved Problems

1. 3Bh → 30h
31h
32h

```
MOV 30h,#3Bh
MOV 31h, 30h
MOV 32h, 31h
```

2.
```
MOV 0F1h , R0
MOV R3 , R0
```

Binary to Hex & Hex to Binary

$(234)_{16} = (?)_2$

2 3 4

0010 0011 0100

$\Rightarrow (001000110100)_2$

$(\underline{1000}\,\underline{1101}\,\underline{0001})_2 = (?)_{16}$

8 13 (D) 1

← from Right

$\Rightarrow (8D1)_{16}$

~~D1~~ ~~13~~ 1

110101

3 5 ✓

Hex to Dec

$(A69.8)_{16} = (?)_{10}$

A 6 9 . 8

$(A\times16^2) + (6\times16^1) + (9\times16^0) \cdot (8\times16^{-1})$

$\Rightarrow 2665.5$

Dec to Hex

$(2482)_{16} = (?)_{16}$

16	2482	
16	155	2
	9	11

$(9\ 11\ 2)_{16} \Rightarrow (9B2)_{16}$

3) MOV 081h,#80h

6)

4) ~~XCH 81h, PSW~~

XCH A, 81h

XCh A, PSW

XCH A, 81h

5) 27h → 27h

intl ext

MOV A, #27h

MOVX @A, 27h

*

- Logical Operations
- Arithmetic Operations
- Jump and Call OpCode