

## LESSON 2: DATA TRANSFER INSTRUCTIONS

---

Objectives:

Study the operation of data transfer instructions

- MOV A,<source>
- MOV <destination>, A
- MOV <destination>, <source>
- MOV DPTR,#data16
- MOVX A, @DPTR
- MOVX @DPTR, A
- PUSH <source>
- POP <destination>
- XCH A, <byte>
- XCHD A, @Ri

Most of operations of the CPU are data transferring. When we write a program in assembly language, we will use many instructions that move 8-bit data between register and memory. This lesson we will learn the operation of data transfer instruction using single step mode. We can examine the content of registers after execution easily.

### MOV A, <source>

The addressing modes are:

Direct	MOV A,30H
Indirect	MOV A,@R0
Register	MOV A,R0
Immediate	MOV A,#1FH

**EXERCISE 2-1:** Enter the opcode shown below started from address 8100. Single step and answer the question.

ADDR	OPCODE	
8100	E530	1 mov a,30h
8102	7831	2 mov r0,#31h
8104	E6	3 mov a,@r0
8105	E8	4 mov a,r0
8106	741F	5 mov a,#1Fh

1. What is the content of Accumulator after the instruction mov a,30h was executed?  
We can use command 'i' to dump internal memory.

ADDR:8121> Hex dump internal memory

```

00: 00 67 C0 59 79 00 21 81 B6 08 7B 95 90 AF 99 D8
10: 04 00 1B 00 1B 00 00 00 00 00 00 00 00 00 00
20: 81 A7 83 D3 48 F8 2B 7A 0D 81 C2 E5 9F 3C 87 D7
30: F4 DF 01 82 07 07 84 01 79 B3 0E EA 0A 00 87 00
40: 87 00 80 F7 A3 24 46 BB FB D0 4E 6E 99 32 EE 77
50: AA 61 F7 F0 63 8F F2 2C 3F 64 26 7D E1 85 3B 98
60: F2 33 F0 65 5C ED 9A 14 56 19 E2 CC A2 66 0C 63
70: 0E 83 F3 E4 34 0D C3 89 E7 36 AF FA 20 95 11 AD
80: 21 48 05 F6 25 DE 12 69 94 AB 1C 72 C2 4B 00 19
90: 63 46 DB 29 54 E1 5B CF DD 56 5C 80 1A 6E 04 E7
A0: 30 CB 66 0A 6C 5D B3 11 34 5E 03 4B 9B D3 FE 1C
B0: 42 1A B5 51 2A F5 B0 DA 73 FB 8B E7 3C E1 11 FF
C0: 45 35 DE 8A AF AC D3 4C C7 B3 FE 77 35 EE EB 91
D0: 77 DE 1A A2 84 91 63 24 F8 DE A6 BD 8D 87 F2 BF
E0: E6 4F 6E BD AF 75 1C D4 87 C2 CB DE 55 50 78 D5
F0: E9 5C 80 DC 1F 61 B3 4C 99 0B C5 9D 2C 3E DB C6

```

ADDR:8121>

2. Where is the location of a byte that was copied into Accumulator?
3. After line3 was executed, what is the content of Accumulator?
4. The same question for line 4 and line 5.

### MOV <destination>,A

This instruction is the same operation of above description but in opposite direction. However only three addressing modes are available. The content of Accumulator will be copied to destination.

The addressing modes are:

Direct	MOV 30H, A ; copy content of Accumulator to memory at address 30H
Indirect	MOV @R0, A ; copy content of Accumulator to memory pointed to by R0
Register	MOV R7, A ; copy content of Accumulator to register R7

**EXERCISE 2-2:** Find the value after execution for each instruction.

ADDR OPCODE

8100	E530	8	mov a,30h ; memory address 30h = ? A = ?
8102	7430	9	mov a,#30h; A=?
8104	F530	10	mov 30h,a ; memory address 30h = ?
		11	
8106	7931	12	mov r1,#31h ; R1 = ?
8108	F7	13	mov @r1,a ; memory address 31h = ?
8109	F8	14	mov r0,a ; R0 = ?

## MOV <destination>, <source>

This instruction allows 15-combinations of source and destination including the Accumulator.

MOV A,Rn  
MOV A,direct  
MOV A,@Ri  
MOV A,#data  
MOV Rn,A  
MOV Rn,direct  
MOV Rn,#data

MOV direct, A  
MOV direct, Rn  
MOV direct,direct  
MOV direct,@Ri  
MOV direct,#data

MOV @Ri,A  
MOV @Ri,direct  
MOV @Ri,#data

Note: Ri can be only R0 or R1. Rn are R0, R1, R2, R3, R4, R5, R6, R7  
Direct is 8-bit onchip memory location. Data is 8-bit constant.

**EXERCISE 2-3:** Find the value after execution for each instruction.

ADDR OPCODE			
8100	783F	9	mov r0,#3fH ; R0 = ?
8102	E8	10	mov a,r0 ; A = ?
8103	E540	11	mov a,40h; A = ?
8105	7840	12	mov r0,#40h
8107	E6	13	mov a,@r0; A = ?
8108	7440	14	mov a,#40h; A =?
		15	
810A	FE	16	mov r6,a ; R6 = ?
810B	AF41	17	mov r7,41h ; R7 = ?
810D	7F40	18	mov r7,#40h; R7 = ?
		19	
810F	F540	20	mov 40h,a; [40h] = ?
8111	8D41	21	mov 41h,r5; [41h] = ?
8113	854443	22	mov 43h,44h; [43h] = ?
8116	7545FF	23	mov 45h,#0ffh ; [45h] = ?
		24	

8119	7850	25	mov r0,#50h
811B	F6	26	mov @r0,a ; [50h] = ?
811C	A651	27	mov @r0,51h; [50h] = ?
811E	75527F	28	mov 52h,#7fh; [52h] = ?

### MOV DPTR,#data16

**MOVX A, @DPTR**

**MOVX @DPTR, A**

**MOVX A, @Ri**

**MOVX @Ri, A**

We can use MOVX for external memory reading/writing. The location pointer can be 16-bit using DPTR or 8-bit using R0 or R1. With DPTR, we can point any locations within 64kB space. Instead with R0 or R1 we can point only address from 0 to FF or 256 locations.

Note: The onboard I/O devices are mapped into external memory having its location as follows;

Address	I/O
0000h	LCD command write register
0001h	LCD data write register
0002h	LCD command read
0003h	LCD data read
0100h	GPIO1, 8-bit output port
0200h	GPIO2, 8-bit input port
0300h-07FFh	expansion I/O space

We see that we can use R0 and R1 to point the address of LCD registers. But for GPIO and expansion I/O space, we will need DPTR.

**EXERCISE 2-3:** Find the value after execution for each instruction.

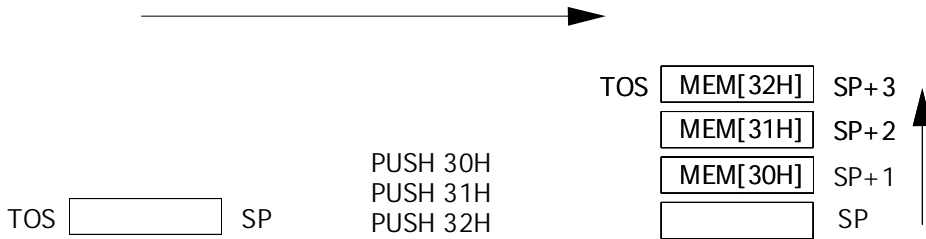
ADDR OPCODE

8100	909000	9	main: mov dptr,#9000h ; content of location 9000h=?
8103	E0	10	movx a,@dptr ; A = ?
8104	90A000	11	mov dptr,#0a000h ; content of location 0A000h=?
8107	F0	12	movx @dptr,a ; content of location 0A000h=?

**EXERCISE 2-4:** Write a program that moves 128 bytes of internal memory 00-7Fh to external memory at 9000h-907FH. (use R7 for loop counter)

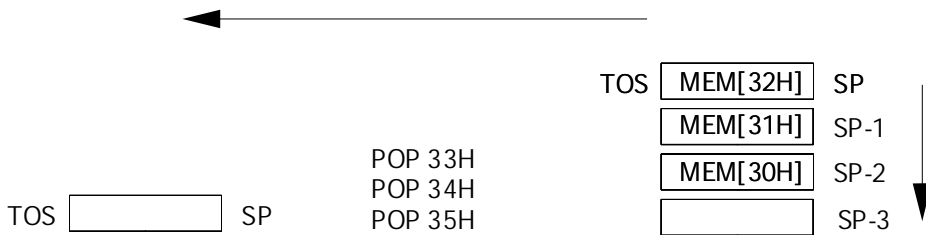
**PUSH <source>**  
**POP <destination>**

PUSH and POP instructions are used to save and retrieve direct byte to the memory area called STACK. STACK is the memory space that reserved for special purpose. SP is stack pointer register. The SP always points to the top of STACK. When we save byte to the STACK, with instruction PUSH, the STACK size will be bigger and when we retrieve it, the STACK will be smaller.



TOS = Top Of Stack

For example, PUSH 30H, the SP register will increment by one and copy the content of memory location 30H to STACK memory. Similarly for PUSH 31H and PUSH 32H.



TOS = Top Of Stack

To retrieve byte from STACK, we use POP instruction. For example, POP 33H, the top of stack will be retrieved and copied to memory location 33H. The SP register then decrements by one. When POP 33H, POP 34H and POP35H are executed, the STACK size will become zero.

**EXERCISE 2-5:** Study the operation of STACK by answer below commands.

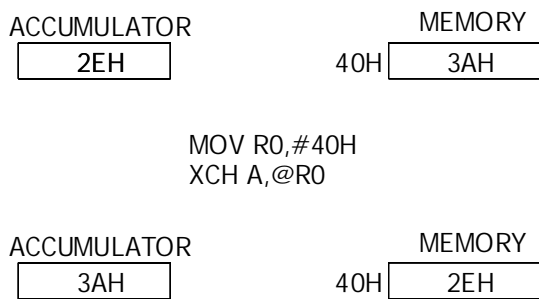
1. Before run the code below with single step, write down the content of memory locations, 30H, 31H and 32H, 33H, 34H and 35H.
2. Where is the top of stack location? Enter single step until ready to execute address 8100.

ADDR OPCODE

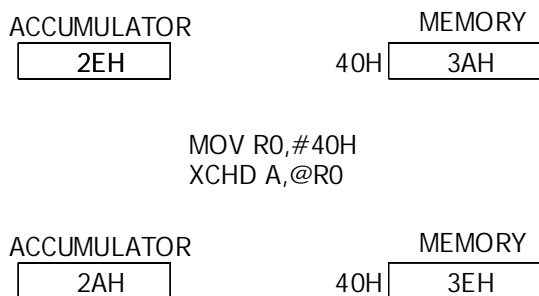
8100	C030	9	main: push 30h; SP = ? memory[SP] = ?
8102	C031	10	push 31h; SP = ? memory[SP] = ?
8104	C032	11	push 32h; SP = ? memory[SP] = ?
		12	
8106	D033	13	pop 33h ; SP = ? memory[33H] = ?
8108	D034	14	pop 34h ; SP = ? memory[34H] = ?
810A	D035	15	pop 35h ; SP = ? memory[35H] = ?

**XCH A, <byte>**  
**XCHD A, @Ri**

XCH stands for **exchange**. The instruction XCH A, < byte>, exchanges content of accumulator with byte variable. The byte variable can use register, direct, or indirect addressing modes.



XCHD stands for **exchange digit**. The instruction XCHD A, @Ri exchanges the low-order nibble of the accumulator with that of the internal memory location indirectly addressed by R0 or R1. The high-order nibbles are not effected.



**EXERCISE 2-6:** Student may now uses ASM51 for translate the source code into machine code and learn how to download the code to the 8051SBC.

1. Write a program that exchanges content of onchip RAM between locations 30H-3FH and 40H-4FH.
2. The same programs, but exchange only low-order nibble.
3. Now exchange between onchip RAM and external RAM, 50H-5FH and 9000H-900FH.