

## Experiment No : 4

**Aim :** To Study various 8051 addressing modes.

**Theory :**

### 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. These are some examples of Immediate Addressing Mode.

```
MOV A, #0AFH;  
MOV R3, #45H;  
MOV DPTR, #FE00H;
```

In these instructions, the # symbol is used for immediate data. In the last instruction, there is DPTR. The DPTR stands for Data Pointer. Using this, it points the external data memory location. In the first instruction, the immediate data is AFH, but one 0 is added at the beginning. So, when the data is starting with A to F, the data should be preceded by 0.

### 2. Register addressing mode

In the register addressing mode the source or destination data should be present in a register (R0 to R7). These are some examples of Register Addressing Mode.

```
MOV A, R5;  
MOV R2, #45H;  
MOV R0, A;
```

In 8051, there is no instruction like **MOVR5, R7**. But we can get the same result by using this instruction **MOV R5, 07H**, or by using **MOV 05H, R7**. But these two instructions will work when the selected register bank is **RB0**. To use another register bank and to get the same effect, we have to add the starting address of that register bank with the register number. For an example, if the RB2 is selected, and we want to access R5, then the address will be (10H + 05H = 15H), so the instruction will look like this **MOV 15H, R7**. Here 10H is the starting address of Register Bank 2.

### 3. Direct Addressing Mode

In the Direct Addressing Mode, the source or destination address is specified by using 8-bit data in the instruction. Only the internal data memory can be used in this mode. Here some of the examples of direct Addressing Mode.

```
MOV 80H, R6;  
MOV R2, 45H;  
MOV R0, 05H;
```

The first instruction will send the content of register R6 to port P0 (Address of Port 0 is 80H). The second one is forgetting content from 45H to R2. The third one is used to get data from Register R5 (When register bank RB0 is selected) to register R5.

#### 4. Register indirect addressing Mode

In this mode, the source or destination address is given in the register. By using register indirect addressing mode, the internal or external addresses can be accessed. The R0 and R1 are used for 8-bit addresses, and DPTR is used for 16-bit addresses, no other registers can be used for addressing purposes. Let us see some examples of this mode.

```
MOV 0E5H, @R0;  
MOV @R1, 80H
```

In the instructions, the @ symbol is used for register indirect addressing. In the first instruction, it is showing that the R0 register is used. If the content of R0 is 40H, then that instruction will take the data which is located at location 40H of the internal RAM. In the second one, if the content of R1 is 30H, then it indicates that the content of port P0 will be stored at location 30H in the internal RAM.

```
MOVX A, @R1;  
MOV @DPTR, A;
```

In these two instructions, the X in MOVX indicates the external data memory. The external data memory can only be accessed in register indirect mode. In the first instruction if the R0 is holding 40H, then A will get the content of external RAM location 40H. And in the second one, the content of A is overwritten in the location pointed by DPTR.

#### 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. These are some examples of Indexed addressing mode.

```
MOVC A, @A+PC;  
MOVC A, @A+DPTR;
```

The C in MOVC instruction refers to code byte. For the first instruction, let us consider A holds 30H. And the PC value is 1125H. The contents of program memory location 1155H (30H + 1125H) are moved to register A.

#### 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. Here are some examples of Implied Addressing Mode.

```
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.

1. Place the number 3Bh in internal RAM locations 30h to 32h.

```
00000000 | mov a, #3Bh
00000002 | mov r0, #30h
00000004 | mov r2, #3
00000006 | again: mov @r0, a
00000007 | inc r0
00000008 | djnz r2, again
```

[illegible]

2. Copy the data at internal RAM location 70h to R0 and R3.

```

0000 | mov r0,70h
0002 | mov r3,70h

```

[illegible]

3. Set the SP at the byte address just above the last working register address.

```
00000000 | mov sp, #20h
```

B	0x00
ACC	0x00
PSW	0x00
IP	0x00
IE	0x00
PCON	0x00
DPH	0x00
DPL	0x00
SP	0x20

```
0000| mov sp,#20h
0003| mov psw,#10h
0006| mov a,sp
0008| xch a,psw
000A| mov sp,a
```

B	0x00
ACC	0x11
PSW	0x20
IP	0x00
IE	0x00
PCON	0x00
DPH	0x00
DPL	0x00
SP	0x11

```

0000 | mov a,27h
0002 | mov r0,#27h
0004 | movx @r0,a

```

[illegible]

```
00000000 | mov th1, #0A2h
00000003 | mov tl1, #3Dh
```

TH0	TL0
0x00	0x00
TMOD	0x00
TCON	0x00
TH1	TL1
0xA2	0x3D

```
0000| mov dptr,#37D6h
0003| mov r0,dpl
0005| mov r1,dph
```

R7	0x00	B	0x00
R6	0x00	ACC	0x00
R5	0x00	PSW	0x00
R4	0x00	IP	0x00
R3	0x00	IE	0x00
R2	0x00	PCON	0x00
R1	0x37	DPH	0x37
R0	0xD6	DPL	0xD6
		SP	0x07

```
0000| mov dptr,#123h
0003| movx a,@dptr
0004| mov t10,a
0006| mov dptr,#234h
0009| movx a,@dptr
000A| mov th0,a
```

R/O

W/O

TH0

TLO

R7

0x00

B

0x00

0x00

0x00

0x00

0x00

R6

0x00

ACC

0x00

RXD

TXD

R5

0x00

PSW

0x00

1

1

TMOD

0x00

R4

0x00

IP

0x00

SCON

0x00

TCON

0x00

R3

0x00

IE

0x00

R2

0x00

PCON

0x00

pins

bits

TH1

TL1

R1

0x37

DPH

0x02

0xFF

0xFF

P3

0x00

R0

0x23

DPL

0x34

0xFF

0xFF

P2

SP

0x07

0xFF

0xFF

P1

0xFF

0xFF

P0

PC

0x0072

r

ACC

0

0

0

0

0

0

0

0

0

0

8051

Data Memory

addr

0x00

0x23

value

0

1

2

3

4

5

6

7

8

9

A

B

C

D

E

F

00

23

37

00

00

00

00

00

00

00

00

00

00

00

00

00

10

00

00

00

00

00

00

00

00

00

00

00

00

00

00

00

20

00

00

00

00

00

00

00

00

00

00

00

00

00

00

00

30

00

00

00

00

00

00

00

00

00

00

00

00

00

00

00

40

00

00

00

00

00

00

00

00

00

00

00

00

00

00

00

50

00

00

00

00

00

00

00

00

00

00

00

00

00

00

00

60

00

00

00

00

00

00

00

00

00

00

00

00

00

00

00

70

00

00

00

00

00

00

00

00

00

00

00

00

00

00

00

```

00000000| mov r2,#4
00000002| mov r0,#12h
00000004| mov r1,#20h
00000006| loop: mov a,@r0
00000007| mov @r1,a
00000008| inc r0
00000009| inc r1
0000000A| djnz r2,loop

```

The diagram illustrates the 8051 microcontroller's pins and internal registers. The pins are organized into four groups: P (Parallel Port), T (Timer/Counter), R (Reset), and S (Serial Port). The internal registers are organized into four groups: P (Parallel Port), T (Timer/Counter), R (Reset), and S (Serial Port).

**8051 Pins:**

Pin	Function
1	NC
2	NC
3	NC
4	NC
5	NC
6	NC
7	NC
8	NC
9	NC
10	NC
11	NC
12	NC
13	NC
14	NC
15	NC
16	NC
17	NC
18	NC
19	NC
20	NC
21	NC
22	NC
23	NC
24	NC
25	NC
26	NC
27	NC
28	NC
29	NC
30	NC
31	NC
32	NC
33	NC
34	NC
35	NC
36	NC
37	NC
38	NC
39	NC
40	NC
41	NC
42	NC
43	NC
44	NC
45	NC
46	NC
47	NC
48	NC
49	NC
50	NC
51	NC
52	NC
53	NC
54	NC
55	NC
56	NC
57	NC
58	NC
59	NC
60	NC
61	NC
62	NC
63	NC
64	NC
65	NC
66	NC
67	NC
68	NC
69	NC
70	NC
71	NC
72	NC
73	NC
74	NC
75	NC
76	NC
77	NC
78	NC
79	NC
80	NC
81	NC
82	NC
83	NC
84	NC
85	NC
86	NC
87	NC
88	NC
89	NC
90	NC
91	NC
92	NC
93	NC
94	NC
95	NC
96	NC
97	NC
98	NC
99	NC
100	NC

**8051 Registers:**

Register	Value
R0	0x00
R1	0x01
R2	0x02
R3	0x03
R4	0x04
R5	0x05
R6	0x06
R7	0x07
B	0x08
ACC	0x09
PSW	0x0A
IP	0x0B
IE	0x0C
PCON	0x0D
DPH	0x0E
DPL	0x0F
SP	0x10

```

00000000 | mov sp, #07h
00000003 | push 81h
00000005 | mov a, 08h

```

The diagram illustrates the internal structure of the 8051 microcontroller, divided into three main sections: Special Function Registers (SFRs), Pins, and Memory.

**Special Function Registers (SFRs):**

- Register Pairs:** R0-R7, R8-R15, R16-R31, and R32-R39 are shown in pairs, each with a value of 0x00.
- Accumulator (ACC):** 0x08
- Program Status Word (PSW):** 0x01
- Instruction Pointer (IP):** 0x01
- Interrupt Enable (IE):** 0x00
- Program Counter (PCON):** 0x00
- Data Pointer High (DPH):** 0x00
- Data Pointer Low (DPL):** 0x00
- Stack Pointer (SP):** 0x08

**Pins:**

- P0:** 0xFF
- P1:** 0xFF
- P2:** 0xFF
- P3:** 0xFF

**Memory:**

- PC (Program Counter):** 0x0029
- PSW (Program Status Word):** 0x00
- RAM (Random Access Memory):** A table showing memory addresses from 00 to FF and their corresponding values (all 00).

```
0000| mov b,#20h
0003| mov r0,b
0005| mov dptr,#2CFh
0008| movx a,@dptr
0009| xch a,r0
000A| movx @dptr,a
000B| mov b,r0
```

B	0x00
ACC	0x20
PSW	0x01
IP	0x00
IE	0x00
PCON	0x00
DPH	0x02
DPL	0xCF
SP	0x07

```
00000000 | mov b, #20h
00000003 | mov r0, b
00000005 | mov dptr, #2CFh
00000008 | movx a, @dptr
00000009 | xch a, r0
0000000A | movx @dptr, a
0000000B | mov b, r0
```

```
0000| clr ea
0002| mov dptr,#7Dh
0005| movc a,@a+dptr
0006| mov sp,a
```

B	0x00
ACC	0x00
PSW	0x00
IP	0x00
IE	0x00
PCON	0x00
DPH	0x00
DPL	0x7D
SP	0x00

```
0000| mov r5,#28h
0002| mov dptr,#32Fh
0005| mov a,r5
0006| movx @dptr,a
```

R7	0x00	B	0x00
R6	0x00	ACC	0x28
R5	0x28	PSW	0x00
R4	0x00	IP	0x00
R3	0x00	IE	0x00
R2	0x00	PCON	0x00
R1	0x00	DPH	0x03
R0	0x20	DPL	0x2F
51		SP	0x07

```
00000000 | mov  dptr,#3000h
00000003 | mov  a,#0
00000005 | movc a,@a+dptr
00000006 | mov  dptr,#3000h
00000009 | movx @dptr,a
```

R7	0x00	B	0x00
R6	0x00	ACC	0x00
R5	0x00	PSW	0x00
R4	0x00	IP	0x00
R3	0x00	IE	0x00
R2	0x00	PCON	0x00
R1	0x00	DPH	0x03
R0	0x20	DPL	0x00
		SP	0x07

```

00000000 | mov th0, #20h
00000003 | mov tl0, #10h
00000006 | mov a, th0
00000008 | xch a, tl0
0000000A | xch a, th0

```

R/O	W/O	TH0	TL0	R7	0x00	B	0x00
0x00	0x00	0x10	0x20	R6	0x00	ACC	0x20
RXD	TXD			R5	0x28	PSW	0x01
1	1	TMOD	0x00	R4	0x00	IP	0x00
SCON	0x00	TCON	0x00	R3	0x00	IE	0x00
				R2	0x00	PCON	0x00
pins	bits	TH1	TL1	R1	0x00	DPH	0x00
0xFF	0xFF	0x00	0x00	R0	0x20	DPL	0x00
0xFF	0xFF					SP	0x07
0xFF	0xFF	PC					
0xFF	0xFF	0x00F4					
				i	PSW	000000	00001

```
00000000| mov dptr,#5F3h
00000003| mov r0,dph
00000005| mov a,dpl
00000007| mov dptr,#123h
0000000A| movx @dptr,a
0000000B| mov a,r0
0000000C| mov dptr,#2BCh
0000000F| movx @dptr,a
```

R7	0x00	B	0x00
R6	0x00	ACC	0x05
R5	0x28	PSW	0x00
R4	0x00	IP	0x00
R3	0x00	IE	0x00
R2	0x00	PCON	0x00
R1	0x00	DPH	0x02
R0	0x05	DPL	0xBC
		SP	0x07

```
0000 | mov r0,#12h
0002 | mov r1,#34h
0004 | mov a,r0
0005 | mov r0,#1h
0007 | xchd a,@r0
0008 | mov r0,a
```

R7	0x00	B	0x00
R6	0x00	ACC	0x14
R5	0x28	PSW	0x00
R4	0x00	IP	0x00
R3	0x00	IE	0x00
R2	0x00	PCON	0x00
R1	0x32	DPH	0x00
R0	0x14	DPL	0x00
51		SP	0x07

```

0000 | mov 20h, #10h
0003 | mov 08h, #25h
0006 | mov r0, 08h
0008 | mov @r0, 20h

```

[illegible]

```

0000 | mov a, #22h
0002 | mov r0, a
0003 | mov @r0, a

```

The diagram illustrates the internal structure of the 8051 microcontroller. It is divided into three main sections: CPU, Memory, and Pins.

**CPU Section:**

- Registers:** R0, R1, R2, R3, R4, R5, R6, R7. R0-R7 are 8-bit registers. R0-R7 are labeled with their addresses: R0 (0x00), R1 (0x01), R2 (0x02), R3 (0x03), R4 (0x04), R5 (0x05), R6 (0x06), R7 (0x07).
- Special Function Registers (SFRs):** ACC (0x00), PSW (0x01), IP (0x02), IE (0x03), PCON (0x07), DPH (0x08), DPL (0x09), SP (0x0D).
- Control Registers:** TH0, TL0, TH1, TL1, TMOD, TCON, PC (Program Counter).
- Other Registers:** B (0x00), ACC (0x02), PSW (0x01), IP (0x02), IE (0x03), PCON (0x07), DPH (0x08), DPL (0x09), SP (0x0D).

**Memory Section:**

- Data Memory:** 0000-FFFF. It is divided into 256 bytes (00-FF).
- Program Memory:** 0000-FFFF. It is divided into 65536 bytes (0000-FFFF).

**Pins Section:**

- P0:** 8-bit bus, multiplexed with address and data.
- P1:** 8-bit bus, multiplexed with address and data.
- P2:** 8-bit bus, multiplexed with address and data.
- P3:** 8-bit bus, multiplexed with address and data.
- P4:** 8-bit bus, multiplexed with address and data.

The CPU is labeled **8051** in the center. The memory system is labeled **Memory** at the bottom. The pins are labeled **P0**, **P1**, **P2**, **P3**, and **P4** on the left side.



21. Copy program bytes 0100h to 0102h to internal RAM locations 20h to 22h.

```

org 0000h
0000| mov dptr,#100h
0003| mov r0,#20h
0005| mov r2,#3
0007| loop: clr a
0008| movc a,@a+dptr
0009| mov @r0,a
000A| inc dptr
000B| inc r0
000C| djnz r2,loop
      org 100h
      PD:db "ABC"

```

R/O	W/O	TH0	TL0	R7	0x00	B	0x00
0x00	0x00	0x00	0x00	R6	0x00	ACC	0x43
RXD	TXD			R5	0x00	PSW	0x01
1	1	TMOD	0x00	R4	0x00	IP	0x00
SCON	0x00	TCON	0x00	R3	0x00	IE	0x00
				R2	0x00	PCON	0x00
pins	bits	TH1	TL1	R1	0x00	DPH	0x01
0xFF	0xFF	P3	0x00	0x00		DPL	0x03
0xFF	0xFF	P2		R0	0x23	SP	0x07
0xFF	0xFF	P1					
0xFF	0xFF	P0					
		PC	0x0071	8051			
				PSW	0	0	0
					0	0	1
Modify RAM							
Data Memory		addr	0x22	0x00	value		
		0	1	2	3	4	5
00	23	00	00	00	00	00	00
10	00	00	00	00	00	00	00
20	41	42	43	00	00	00	00
30	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00

22. PUSH the contents of the B register to TMOD.

```

0000| mov b,#25h
0003| mov sp,#88h
0006| push 0F0h

```

R/O	W/O	TH0	TL0	R7	0x00	B	0x25
0x00	0x00	0x00	0x00	R6	0x00	ACC	0x00
RXD	TXD			R5	0x00	PSW	0x00
1	1	TMOD	0x25	R4	0x00	IP	0x00
SCON	0x00	TCON	0x00	R3	0x00	IE	0x00
				R2	0x00	PCON	0x00
pins	bits	TH1	TL1	R1	0x00	DPH	0x00
0xFF	0xFF	P3	0x00	0x00		DPL	0x00
0xFF	0xFF	P2				SP	0x89
0xFF	0xFF	P1					
0xFF	0xFF	P0					
		PC	0x017F	8051			
				PSW	0	0	0
					0	0	0

23. Copy the contents of external code memory address 0040h to IE.

```

0000| clr ea
0002| mov dptr,#40h
0005| clr a
0006| movc a,@a+dptr
0007| mov ie,a

```

R/O	W/O	TH0	TL0	R7	0x00	B	0x00
0x00	0x00	0x00	0x00	R6	0x00	ACC	0x00
RXD	TXD			R5	0x00	PSW	0x00
1	1	TMOD	0x00	R4	0x00	IP	0x00
SCON	0x00	TCON	0x00	R3	0x00	IE	0x00
				R2	0x00	PCON	0x00
pins	bits	TH1	TL1	R1	0x00	DPH	0x00
0xFF	0xFF	P3	0x00	0x00		DPL	0x40
0xFF	0xFF	P2				SP	0x07
0xFF	0xFF	P1					
0xFF	0xFF	P0					
		PC	0x022A	8051			
				PSW	0	0	0
					0	0	0

24. Write a program to copy 10 bytes of data stored from 30H to another location starting from 50H.

```
00000000| mov r0,#300h
00000002| mov r1,#500h
00000004| mov r2,#10
00000006| loop: mov a,@r0
00000007| mov @r1,a
00000008| inc r0
00000009| inc r1
0000000A| djnz r2,loop
```

R/O	W/O	TH0	TL0	R7	0x00	B	0x00
0x00	0x00	0x00	0x00	R6	0x00	ACC	0x39
RXD	TXD			R5	0x00	PSW	0x00
1	1	TMOD	0x00	R4	0x00	IP	0x00
SCON	0x00	TCON	0x00	R3	0x00	IE	0x00
				R2	0x00	PCON	0x00
pins	bits	TH1	TL1	R1	0x5A	DPH	0x00
0xFF	0xFF	0x00	0x00	R0	0x3A	DPL	0x00
0xFF	0xFF	PC				SP	0x07
0xFF	0xFF						
0xFF	0xFF						
0xFF	0xFF						

25. Add the BCD numbers found in internal RAM locations 25h, 26h and 27h together and put the result in RAM location 31h(MSB) and 30h(LSB).

```
0000 | clr a
0001 | add a, 25h
0003 | add a, 26h
0005 | add a, 27h
0007 | mov r0, #30h
0009 | xchd a, @r0
000A | swap a
000B | inc r0
000C | xchd a, @r0
```

[illegible]

26. Place any number in internal RAM location 3Ch and increment it until the number equals 2Ah. (Use cjne)

```
00000001 | mov r0, #3Ch
00000002 | mov @r0, #25h
00000004 | com: inc @r0
00000005 |     inc a
00000006 |     cjne @r0, #2Ah, com
```

[illegible]

27. Write code to push R0, R1, R3 of bank 0 onto the stack and pop them back into R5, R6, and R7 of bank 3.

```
0000| mov r0,#10h
0002| mov r1,#20h
0004| mov r3,#30h
0006| push 0
0008| push 1
000A| push 3
000C| pop 1Dh
000E| pop 1Eh
0010| pop 1Fh
0012| setb rs1
0014| setb rs0
```

R7	0x30	B	0x00
R6	0x20	ACC	0x00
R5	0x10	PSW	0x18
R4	0x00	IP	0x00
R3	0x00	IE	0x00
R2	0x00	PCON	0x00
R1	0x00	DPH	0x00
R0	0x00	DPL	0x00
8051		SP	0x07
PSW 0 0 0 1 1 0 0 0			

28. Write a program to copy 10 bytes of data starting at ROM address 400H to RAM locations starting at 30H.

```
0000| mov r2,#10
0002| mov dptr,#400h
0005| mov r0,#30h
0007| loop:clr a
0008| movc a,@a+dptr
0009| inc dptr
000A| mov @r0,a
000B| inc r0
000C| djnz r2,loop
      org 400h
      d:db "1234567899"
```

R/O	W/O	TH0	TL0	R7	0x00	B	0x00
0x00	0x00	0x00	0x00	R6	0x00	ACC	0x39
RxD	TxD			R5	0x00	PSW	0x00
1	1	TMOD	0x00	R4	0x00	IP	0x00
SCON	0x00	TCON	0x00	R3	0xFF	IE	0x00
				R2	0x00	PCON	0x00
pins bits		TH1	TL1	R1	0x00	DPH	0x04
0xFF	0xFF	P3	0x00	R0	0x3A	DPL	0x0A
0xFF	0xFF	P2				SP	0x07
0xFF	0xFF	P1					
0xFF	0xFF	P0					
		PC	0x025B	i	ACC	0	0
						1	1
						1	0
						0	1
Data Memory				Modify RAM			
addr				0x00 value			
	0	1	2	3	4	5	6
	7	8	9	A	B	C	D
	E	F					
00	3A	00	00	FF	00	00	00
10	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00
30	31	32	33	34	35	36	37
40	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00

29. Assume that the word INDIA is burned on ROM location starting at 200H and that the program is burned into ROM locations starting at 0. Get this word and store each character in Registers r0-r4.

```
      org 000h
0000| mov dptr,#200h
0003| mov r7,#5
0005| mov r0,#1
0007| loop:clr a
0008| inc dptr
0009| movc a,@a+dptr
000A| mov @r0,a
000B| inc r0
000C| djnz r7,loop
000E| clr a
000F| mov dptr, #200h
0012| movc a,@a+dptr
0013| mov r0,a
      org 200h
      data: db "INDIA"
```

R7	0x00	B	0x00
R6	0x00	ACC	0x49
R5	0x00	PSW	0x01
R4	0x41	IP	0x00
R3	0x49	IE	0x00
R2	0x44	PCON	0x00
R1	0x4E	DPH	0x02
R0	0x49	DPL	0x00
51		SP	0x07

30. Write the same program in question 29 using

a. a counter

```

    org 0000h
0000| mov dptr,#2000h
0003| mov tmod,#01h
0006| mov r0,#1
0008| mov th0,#0FFh
000B| mov tl0,#0EAh
000E| setb tr0
0010| loop:clr a
0011|   inc dptr
0012|   movc a,@a+dptr
0013|   mov @r0,a
0014|   inc r0
0015|   jnb tf0,loop
0018| clr a
0019| mov dptr, #2000h
001C| movc a,@a+dptr
001D| mov r0,a
    org 2000h
    data: db "INDIA"

```

R/O	W/O	TH0	TL0	R7	0x00	B	0x00
0x00	0x00	0x00	0x07	R6	0x00	ACC	0x49
RXD	TXD			R5	0x00	PSW	0x01
1	1	TMOD	0x01	R4	0x41	IP	0x00
SCON	0x00	TCON	0x20	R3	0x49	IE	0x00
				R2	0x44	PCON	0x00
pins	bits	TH1	TL1	R1	0x4E	DPH	0x02
0xFF	0xFF P3	0x00	0x00	R0	0x49	DPL	0x00
0xFF	0xFF P2					SP	0x07
0xFF	0xFF P1	PC					
0xFF	0xFF P0	0x0043					

8051

ACC 01001001

b. using null char at the end of the string

```

    org 0000h
0000| mov dptr,#2000h
0003| mov r0,#1
0005| loop:clr a
0006|   inc dptr
0007|   movc a,@a+dptr
0008|   mov @r0,a
0009|   inc r0
000A|   cjne a,#0h,loop
000D| clr a
000E| mov dptr, #2000h
0011| movc a,@a+dptr
0012| mov r0,a
    org 2000h
    data: db "INDIA"

```

R/O	W/O	TH0	TL0	R7	0x00	B	0x00
0x00	0x00	0x00	0x00	R6	0x00	ACC	0x4F
RXD	TXD			R5	0x00	PSW	0x01
1	1	TMOD	0x00	R4	0x41	IP	0x00
SCON	0x00	TCON	0x00	R3	0x49	IE	0x00
				R2	0x44	PCON	0x00
pins	bits	TH1	TL1	R1	0x4E	DPH	0x02
0xFF	0xFF P3	0x00	0x00	R0	0x49	DPL	0x00
0xFF	0xFF P2					SP	0x07
0xFF	0xFF P1	PC					
0xFF	0xFF P0	0x0201					

8051

ACC 01001111

```

org 000h
0000| mov dptr,#300h
0003| mov a,#0FFh
0005| mov p1,a
0007| back:mov a,p1
0009|      movc a,@a+dptr
000A|      mov p2,a
000C|      sjmp back
org 300h
xsqr_table:db 0,1,4,9,16,25,36,49

```

R7	0x00	B	0x00
R6	0x00	ACC	0x00
R5	0x00	PSW	0x00
R4	0x00	IP	0x00
R3	0x00	IE	0x00
R2	0x00	PCON	0x00
R1	0x00	DPH	0x03
R0	0x00	DPL	0x00
51		SP	0x07

```

00000000 | mov r2, #0
00000002 | mov r0, #20h
00000004 | loop: mov a, r2
00000005 | mov b, r2
00000007 | mul ab ;x2
00000008 | add a, r2
00000009 | add a, r2; x2+2x
0000000A | add a, #5; x2+2x+5
0000000C | mov @r0, a
0000000D | inc r0
0000000E | inc r2
0000000F | cjne r2, #10, loop

```

[illegible]

33. Write a program to add the following data and store the result in RAM location 30H. ORG 200H

MYDATA: DB 06,09,02,05,07

```
org 000h
0000| mov dptr,#200h
0003| mov r1,#20h
0005| mov r2,#5
0007| rloop:clr a
0008| movc a,@a+dptr
0009| mov @r1,a
000A| inc r1
000B| inc dptr
000C| djnz r2,rloop
000E| mov r2,#5
0010| mov r1,#20h
0012| clr a
0013| aloop:add a,@r1
0014| inc r1
0015| djnz r2,aloop
0017| mov 30h,a
      org 200h
mydata: db 06,09,02,05,07
```

R/O	W/O	TH0	TL0	R7	0x00	B	0x00
0x00	0x00	0x00	0x00	R6	0x00	ACC	0x1D
RXD	TXD			R5	0x00	PSW	0x00
1	1	TMOD	0x00	R4	0x00	IP	0x00
SCON	0x00	TCON	0x00	R3	0x00	IE	0x00
				R2	0x00	PCON	0x00
pins	bits	TH1	TL1	R1	0x26	DPH	0x02
0xFF	0xFF	P3	0x00	R0	0x2A	DPL	0x05
0xFF	0xFF	P2				SP	0x07
0xFF	0xFF	P1					
0xFF	0xFF	P0					
				PC	8051		
					ACC	0	0
						0	1
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1
						1	0
						1	1
						0	1