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 location40H. 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 is1125H. 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.

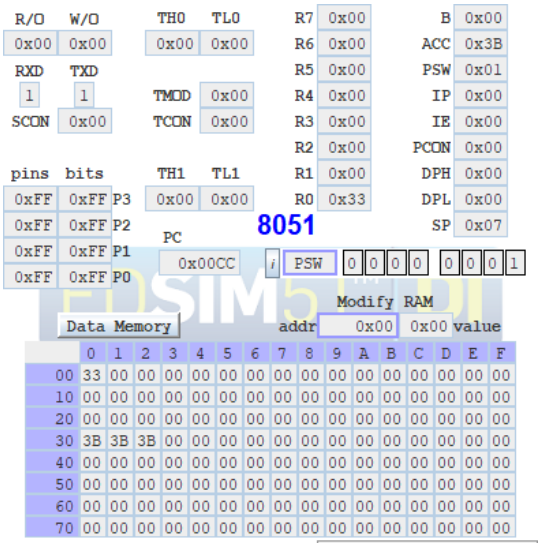
```
RL A;
SWAP A;
```

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.

Programs :

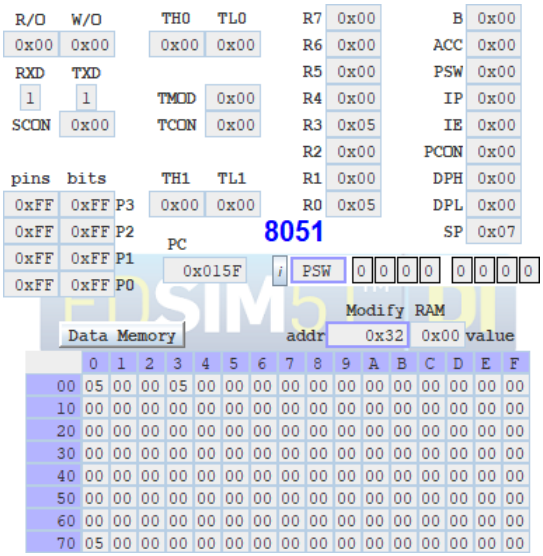
1. Place the number 3Bh in internal RAM locations 30h to 32h.

```
ØØØØ| mov a,#3Bh
ØØØ2| mov rØ,#3Øh
ØØØ4| mov r2,#3
ØØØ6| again:mov @rØ,a
ØØØ7| inc rØ
ØØØ8| djnz r2,again
```



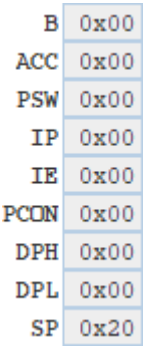
2. Copy the data at internal RAM location 70h to R0 and R3.

```
ØØØØ| mov rØ,7Øh
ØØØ2| mov r3,7Øh
```



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

```
ØØØØ| mov sp,#2Øh
```



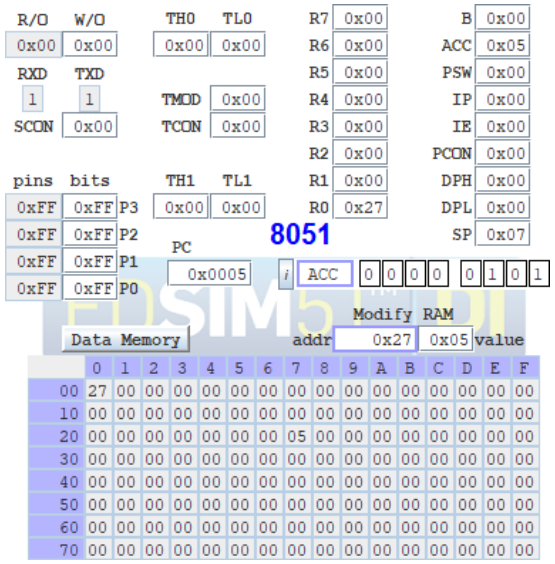
4. Exchange the contents of the SP and the PSW.

```
ØØØØ| mov sp,#2Øh
ØØØ3| mov psw,#1Øh
ØØØ6| mov a,sp
ØØØ8| xch a,psw
ØØØA| mov sp,a
```

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

5. Copy the byte at internal RAM address 27h to external RAM address 27h.

```
ØØØØ| mov a,27h
ØØØ2| mov rØ,#27h
ØØØ4| movx @rØ,a
```



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	27	00	00	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	00
20	00	00	00	00	00	00	00	00	05	00	00	00	00	00	00	00
30	00	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	00
50	00	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	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

6. Set Timer 1 to A23Dh.

```
ØØØØ| mov th1,#ØA2h
ØØØ3| mov tl1,#3Dh
```

TH0	TL0
0x00	0x00

TMOD	0x00
TCON	0x00

TH1	TL1
0xA2	0x3D

7. Copy the contents of DPTR to registers R0 (DPL) and R1 (DPH).

```
ØØØØ| mov dptr,#37D6h
ØØØ3| mov rØ,dpl
ØØØ5| mov r1,dph
```

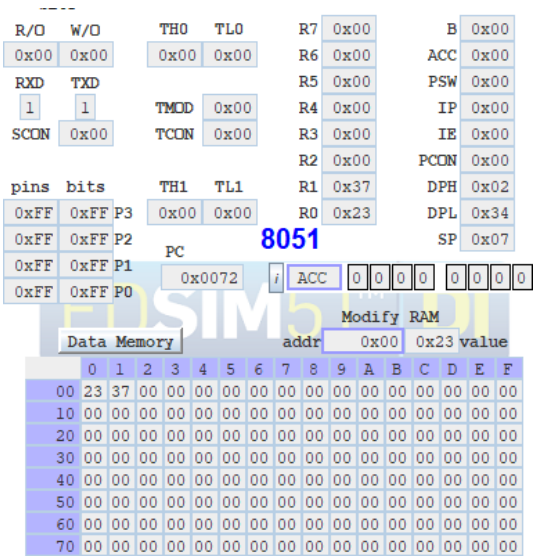
R7	0x00
R6	0x00
R5	0x00
R4	0x00
R3	0x00
R2	0x00
R1	0x37
R0	0xD6

51

B	0x00
ACC	0x00
PSW	0x00
IP	0x00
IE	0x00
PCON	0x00
DPH	0x37
DPL	0xD6
SP	0x07

8. Copy the data in external RAM location 0123h to TL0 and the data in external RAM location 0234h to TH0.

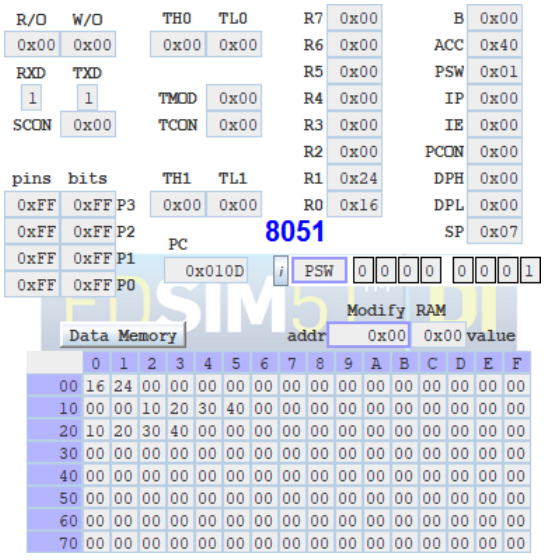
```
0000| mov dptr,#123h
0003| movx a,@dptr
0004| mov tl0,a
0006| mov dptr,#234h
0009| movx a,@dptr
000A| mov th0,a
```



	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	00
10	00	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	00
30	00	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	00
50	00	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	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

9. Copy the data in internal RAM locations 12h to 15h to internal RAM locations 20h to 23h: Copy 12h to 20h. 13h to 21h, etc.

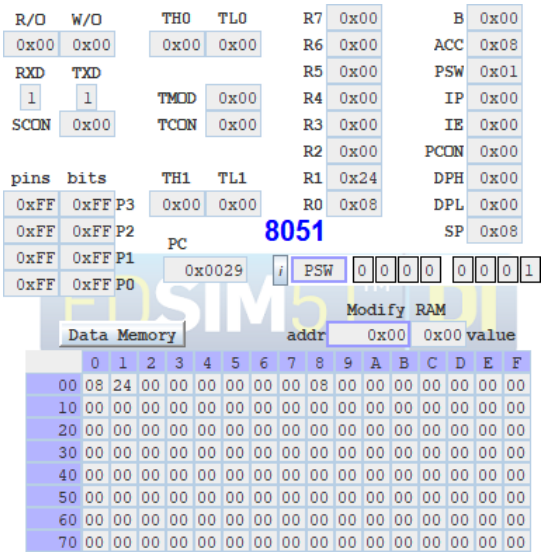
```
0000| mov r2,#4
0002| mov r0,#12h
0004| mov r1,#20h
0006| loop: mov a,@r0
0007| mov @r1,a
0008| inc r0
0009| inc r1
000A| djnz r2,loop
```



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	16	24	00	00	00	00	00	00	00	00	00	00	00	00	00	00
10	00	00	10	20	30	40	00	00	00	00	00	00	00	00	00	00
20	10	20	30	40	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	00
40	00	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	00
60	00	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	00

10. Set the SP register to 07h and PUSH the SP register on the stack; predict what number is pushed to address 08h.

```
ØØØØ| mov sp,#Ø7h
ØØØ3| push 81h
ØØØ5| mov a,Ø8h
```



11. Exchange the contents of the B register and external RAM address 02CFh.

```
ØØØØ| mov b,#2Øh
ØØØ3| mov rØ,b
ØØØ5| mov dptr,#2CFh
ØØØ8| movx a,@dptr
ØØØ9| xch a,rØ
ØØØA| movx @dptr,a
ØØØB| mov b,rØ
```

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

12. Rotate the bytes in registers R0 to R3; copy the data in R0 to R 1, R 1 to R2, R2 to R3, and R3 to R0.

```
ØØØØ| mov b,#2Øh
ØØØ3| mov rØ,b
ØØØ5| mov dptr,#2CFh
ØØØ8| movx a,@dptr
ØØØ9| xch a,rØ
ØØØA| movx @dptr,a
ØØØB| mov b,rØ
```

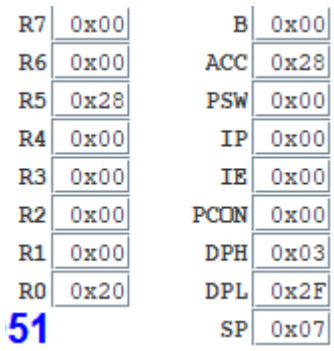
13. Copy the external code byte at address 007Dh to the SP.

```
ØØØØ| clr ea
ØØØ2| mov dptr,#7Dh
ØØØ5| movc a,@a+dptr
ØØØ6| mov sp,a
```

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

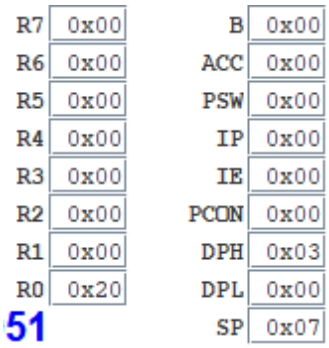
14. Copy the data in register R5 to external RAM address 032Fh.

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



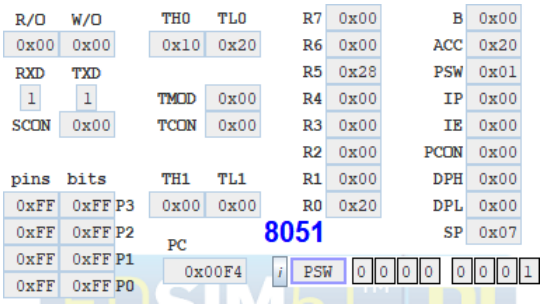
15. Copy the internal code byte at address 0300h to external RAM address 0300h.

```
0000| mov dptr,#300h
0003| mov a,#0
0005| movc a,@a+dptr
0006| mov dptr,#300h
0009| movx @dptr,a
```



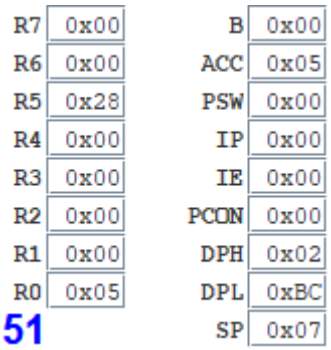
16. Swap the bytes in timer 0; put TL0 in TH0 and TH0 in TL0.

```
0000| mov th0,#20h
0003| mov tl0,#10h
0006| mov a,th0
0008| xch a,tl0
000A| xch a,th0
```



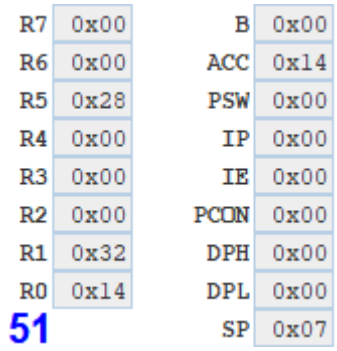
17. Store DPTR in external RAM locations 0123h (DPL) and 02BCH (DPH).

```
0000| mov dptr,#5F3h
0003| mov r0,dph
0005| mov a,dpl
0007| mov dptr,#123h
000A| movx @dptr,a
000B| mov a,r0
000C| mov dptr,#2BCh
000F| movx @dptr,a
```



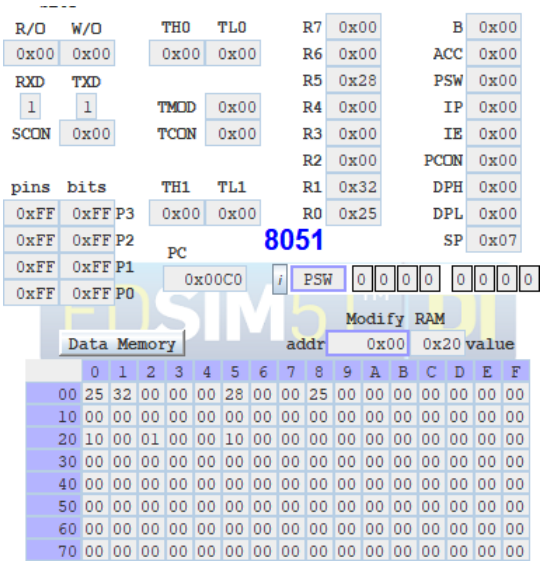
18. Exchange both low nibbles of registers R0 and R1; put the low nibble of R0 in R1, and the low nibble of R1 in R0.

```
ØØØØ| mov rØ,#12h
ØØØ2| mov r1,#34h
ØØØ4| mov a,rØ
ØØØ5| mov rØ,#1h
ØØØ7| xchd a,@rØ
ØØØ8| mov rØ,a
```



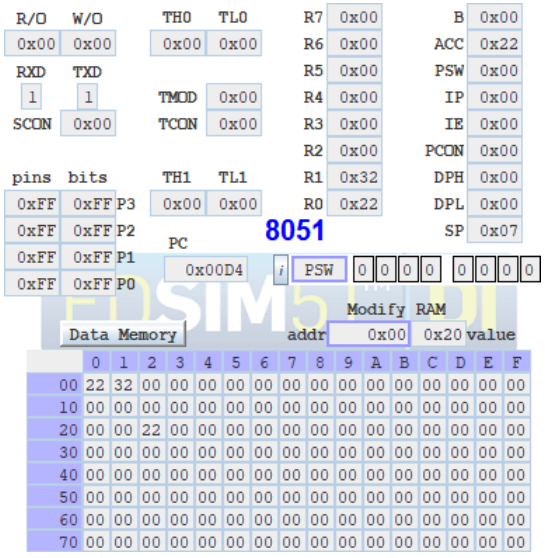
19. Store the contents of RAM location 20h at the address contained in RAM location 08h.

```
ØØØØ| mov 2Øh,#1Øh
ØØØ3| mov Ø8h,#25h
ØØØ6| mov rØ,Ø8h
ØØØ8| mov @rØ,2Øh
```



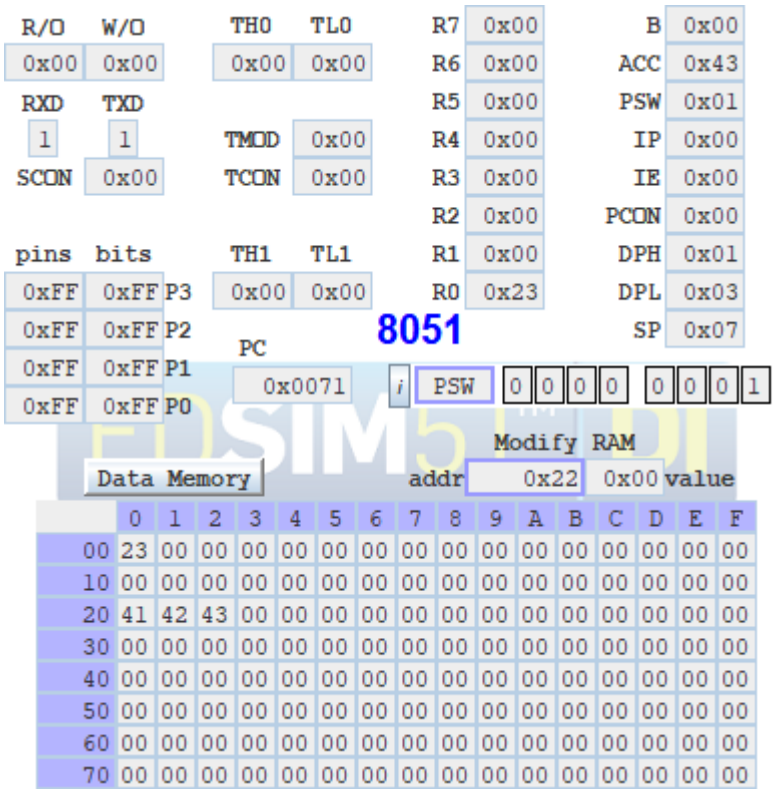
20. Store register A at the internal RAM location address in register A.

```
ØØØØ| mov a,#22h
ØØØ2| mov rØ,a
ØØØ3| mov @rØ,a
```



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

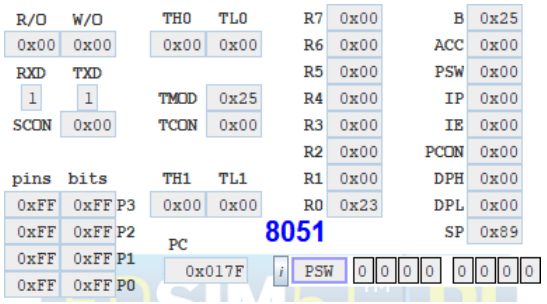
```
      org ØØØØh
ØØØØ| mov dptr,#1ØØh
ØØØ3| mov rØ,#2Øh
ØØØ5| mov r2,#3
ØØØ7| loop: clr a
ØØØ8| movc a,@a+dptr
ØØØ9| mov @rØ,a
ØØØA| inc dptr
ØØØB| inc rØ
ØØØC| djnz r2,loop
      org 1ØØh
      PD:db "ABC"
```



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	23	00	00	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	00
20	41	42	43	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	00
40	00	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	00
60	00	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	00

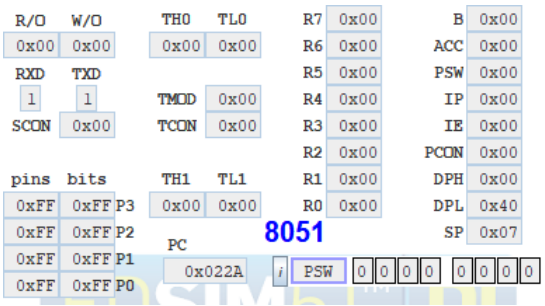
22. PUSH the contents of the B register to TMOD.

```
ØØØØ| mov b,#25h
ØØØ3| mov sp,#88h
ØØØ6| push ØF9h
```



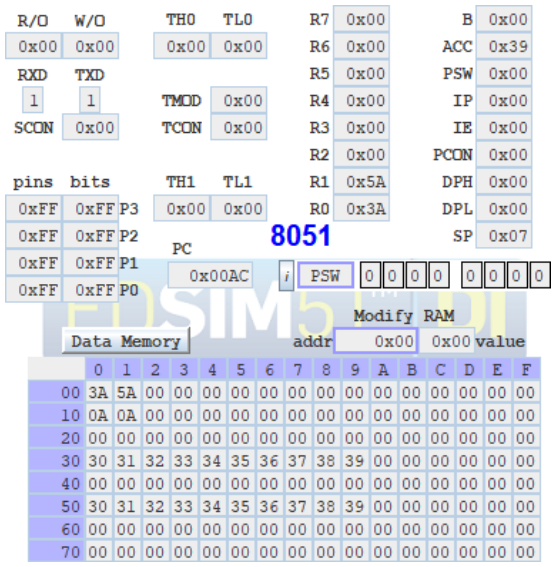
23. Copy the contents of external code memory address 0040h to IE.

```
ØØØØ| clr ea
ØØØ2| mov dptr,#4Øh
ØØØ5| clr a
ØØØ6| movc a,@a+dptr
ØØØ7| mov ie,a
```



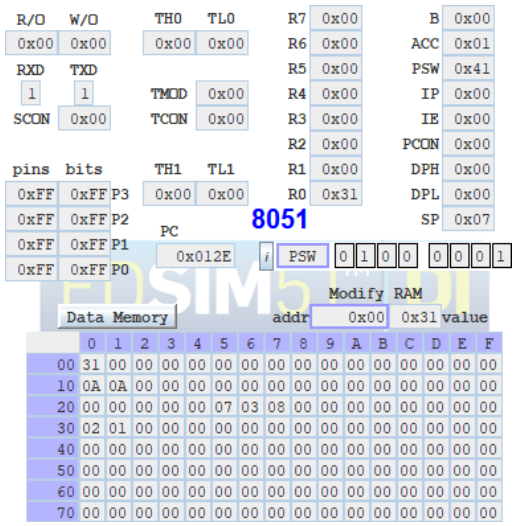
24. Write a program to copy 10 bytes of data stored from 30H to another location staring from 50H.

```
ØØØØ| mov rØ,#3Øh
ØØØ2| mov r1,#5Øh
ØØØ4| mov r2,#1Ø
ØØØ6| loop: mov a,@rØ
ØØØ7| mov @r1,a
ØØØ8| inc rØ
ØØØ9| inc r1
ØØØA| djnz r2,loop
```



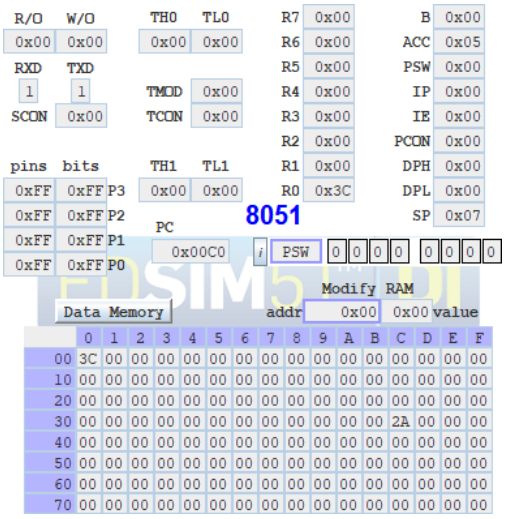
25. Add the BCD numbers found in internal RAM locations 25h, 26h and 27h together and put the result in RAM location 31h(MSB) and 30h(LSB).

```
ØØØØ| clr a
ØØØ1| add a,25h
ØØØ3| add a,26h
ØØØ5| add a,27h
ØØØ7| mov rØ,#3Øh
ØØØ9| xchd a,@rØ
ØØØA| swap a
ØØØB| inc rØ
ØØØC| xchd a,@rØ
```



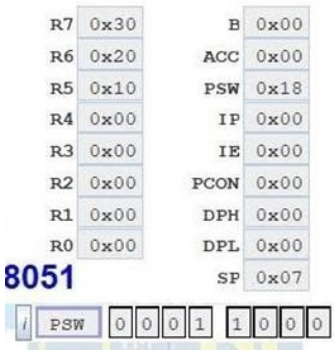
26. Place any number in internal RAM location 3Ch and increment it until the number equals 2Ah. (Use cjne)

```
ØØØØ| mov rØ,#3Ch
ØØØ2| mov @rØ,#25h
ØØØ4| com:inc @rØ
ØØØ5|    inc a
ØØØ6|    cjne @rØ,#2Ah,com
```



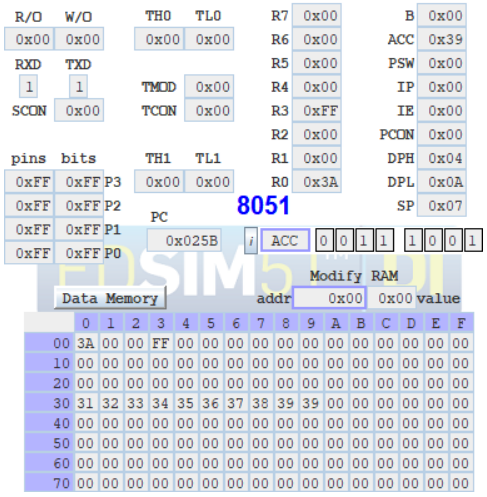
27. Write code to push R0, R1, R3 of bank 0 onto the stack and pop them back into R5, R6, and R7 of bank 3.

```
ØØØØ| mov rØ,#1Øh
ØØØ2| mov r1,#2Øh
ØØØ4| mov r3,#3Øh
ØØØ6| push Ø
ØØØ8| push 1
ØØØA| push 3
ØØØC| pop 1Dh
ØØØE| pop 1Eh
ØØ1Ø| pop 1Fh
ØØ12| setb rs1
ØØ14| setb rsØ
```



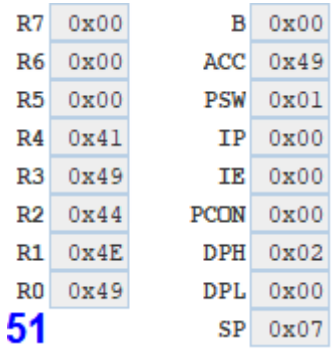
28. Write a program to copy 10 bytes of data starting at ROM address 400H to RAM locations starting at 30H.

```
ØØØØ| mov r2,#1Ø
ØØØ2| mov dptr,#4ØØh
ØØØ5| mov rØ,#3Øh
ØØØ7| loop:clr a
ØØØ8|   movc a,@a+dptr
ØØØ9|   inc dptr
ØØØA|   mov @rØ,a
ØØØB|   inc rØ
ØØØC|   djnz r2,loop
      org 4ØØh
      d:db "1234567899"
```



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.

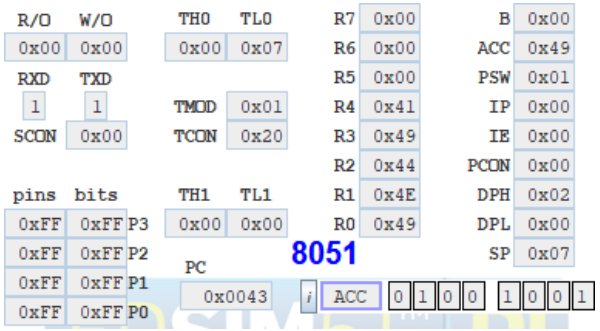
```
      org ØØØh
ØØØØ| mov dptr,#2ØØh
ØØØ3| mov r7,#5
ØØØ5| mov rØ,#1
ØØØ7| loop:clr a
ØØØ8|   inc dptr
ØØØ9|   movc a,@a+dptr
ØØØA|   mov @rØ,a
ØØØB|   inc rØ
ØØØC|   djnz r7,loop
ØØØE| clr a
ØØØF| mov dptr, #2ØØh
ØØ12| movc a,@a+dptr
ØØ13| mov rØ,a
      org 2ØØh
      data: db "INDIA"
```



30. Write the same program in question 29 using

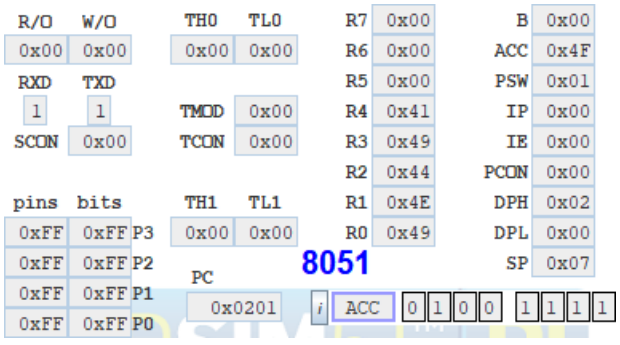
a. a counter

```
      org ØØØh
ØØØØ| mov dptr,#2ØØh
ØØØ3| mov tmod,#Ø1h
ØØØ6| mov rØ,#1
ØØØ8| mov thØ,#ØFFh
ØØØB| mov tlØ,#ØEAh
ØØØE| setb trØ
ØØ1Ø| loop:clr a
ØØ11|   inc dptr
ØØ12|   movc a,@a+dptr
ØØ13|   mov @rØ,a
ØØ14|   inc rØ
ØØ15|   jnb tfØ,loop
ØØ18| clr a
ØØ19| mov dptr, #2ØØh
ØØ1C| movc a,@a+dptr
ØØ1D| mov rØ,a
      org 2ØØh
      data: db "INDIA"
```



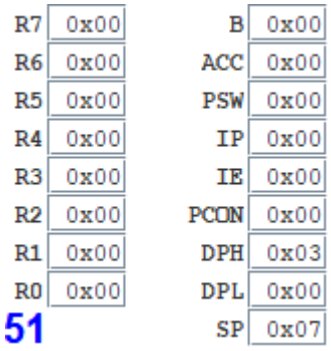
b. using null char at the end of the string

```
      org ØØØh
ØØØØ| mov dptr,#2ØØh
ØØØ3| mov rØ,#1
ØØØ5| loop:clr a
ØØØ6|   inc dptr
ØØØ7|   movc a,@a+dptr
ØØØ8|   mov @rØ,a
ØØØ9|   inc rØ
ØØØA|   cjne a,#Øh,loop
ØØØD| clr a
ØØØE| mov dptr, #2ØØh
ØØ11| movc a,@a+dptr
ØØ12| mov rØ,a
      org 2ØØh
      data: db "INDIA"
```



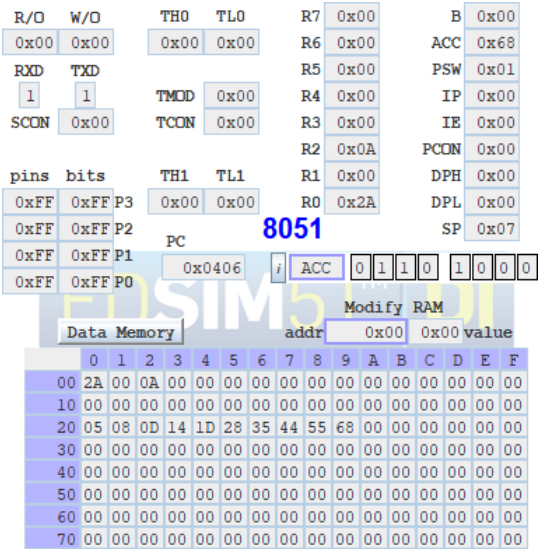
31. Write a program to get the value x from p1 and send x2 to p2 continuously using a look up table for the x2 values.

```
      org ØØØh
ØØØØ| mov dptr,#3ØØh
ØØØ3| mov a,#ØFFh
ØØØ5| mov p1,a
ØØØ7| back:mov a,p1
ØØØ9|   movc a,@a+dptr
ØØØA|   mov p2,a
ØØØC|   sjmp back
      org 3ØØh
      xsqr_table:db Ø,1,4,9,16,25,36,49
```



32. Write a program to find y where y= x2 + 2x + 5 and x is between 0 and 9.

```
ØØØØ| mov r2,#Ø
ØØØ2| mov rØ,#2Øh
ØØØ4| loop:mov a,r2
ØØØ5|   mov b,r2
ØØØ7|   mul ab  ;x2
ØØØ8|   add a,r2
ØØØ9|   add a,r2;x2+2x
ØØØA|   add a,#5;x2+2x+5
ØØØC|   mov @rØ,a
ØØØD|   inc rØ
ØØØE|   inc r2
ØØØF|   cjne r2,#1Ø,loop
```

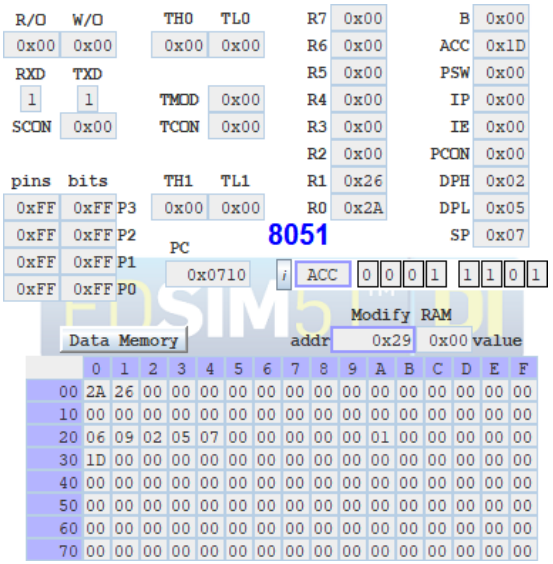


	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	2A	00	0A	00	00	00	00	00	00	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	05	08	0D	14	1D	28	35	44	55	68	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	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	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

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 ØØØh
ØØØØ| mov dptr,#2ØØh
ØØØ3| mov r1,#2Øh
ØØØ5| mov r2,#5
ØØØ7| rloop:clr a
ØØØ8|   movc a,@a+dptr
ØØØ9|   mov @r1,a
ØØØA|   inc r1
ØØØB|   inc dptr
ØØØC|   djnz r2,rloop
ØØØE| mov r2,#5
ØØ1Ø| mov r1,#2Øh
ØØ12| clr a
ØØ13| aloop:add a,@r1
ØØ14|   inc r1
ØØ15|   djnz r2,aloop
ØØ17| mov 3Øh,a
      org 2ØØH
      mydata: db Ø6,Ø9,Ø2,Ø5,Ø7
```



Conclusion :

Programs to study various 8051 Addressing Modes were successfully implemented.

Deepraj Bhosale

181105016