

INSTITUTE OF COMPUTER TECHNOLOGY

B-TECH COMPUTER SCIENCE ENGINEERING 2025-26

SUBJECT: MICROCONTROLLER & APPLICATION

NAME: Rahul Prajapati

ENRLL. NO: 23162171020

BRANCH: CYBER SECURITY

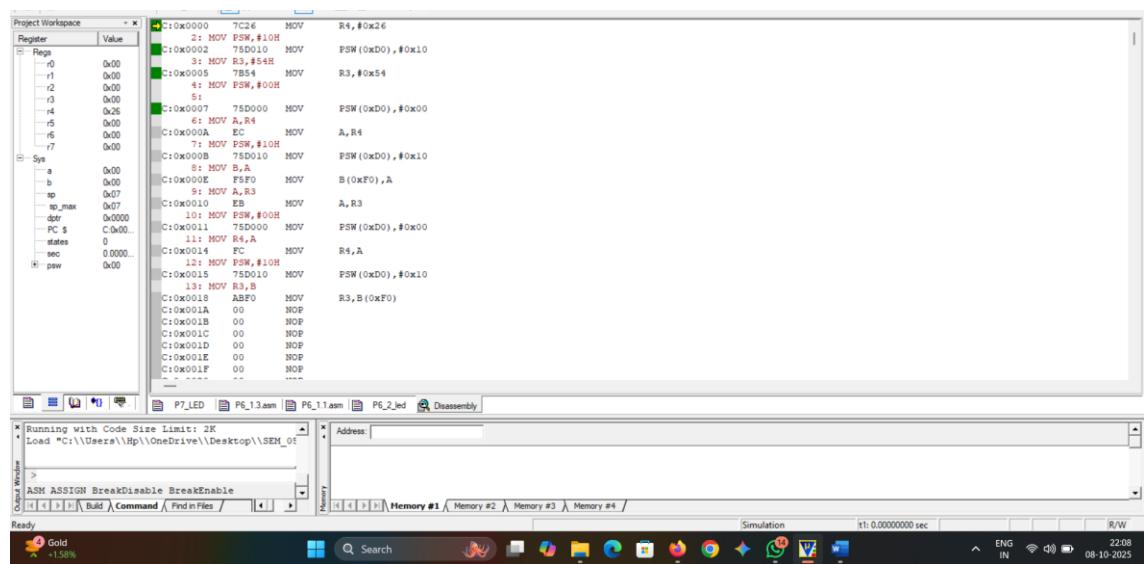
BATCH: 52

PRACTICAL_6

Aim:- Learning programs based on various Addressing modes of 8051 & learning use of Proteus Hardware Simulator for 8051 application programs.

1. Swap the contents of R4 of register bank 0 and R3 of register bank 2 using all possible addressing modes.

A. Using Register Addressing Mode:



Project Workspace

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x26
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x54
b	0x26
sp	0x07
sp_max	0x07
dptr	0x0000
PC \$	C0x0000
states	0x0000...
sec	0x0000...
paw	0x11

```

C10x0000 7C16 MOV R4,#0x26
C10x0002 75D010 MOV PSW(0xD0),#0x10
C10x0003 7B54 MOV R3,#0x54
C10x0005 7B54 MOV R3,#0x54
      5:
C10x0007 75D000 MOV PSW(0xD0),#0x00
      6: MOV A,R4
C10x0008 EC MOV A,R4
      7: MOV PSW,#10H
      8: MOV V
      9: MOV A,R3
C10x0010 EB MOV A,R3
      10: MOV PSW,#00H
C10x0011 75D000 MOV PSW(0xD0),#0x00
      11: MOV R4,A
C10x0012 75D010 MOV R4,A
      12: MOV PSW,#10H
C10x0013 75D010 MOV PSW(0xD0),#0x10
      13: MOV R3,B
C10x0014 ABF0 MOV R3,B(0xF0)
      14:
C10x0015 75D010 MOV PSW(0xD0),#0x10
      15: MOV R3,B
C10x0016 00 NOP
C10x0017 00 NOP
C10x0018 00 NOP
C10x0019 00 NOP
C10x001A 00 NOP
C10x001B 00 NOP
C10x001C 00 NOP
C10x001D 00 NOP
C10x001E 00 NOP
C10x001F 00 NOP
C10x0020 00 NOP
      --:

```

P7_LED P6_13.asm P6_11.asm P6_2_led Disassembly

Output Window

```

* Running with Code Size Limit: 2K
* Load "C:\\Users\\Hp\\OneDrive\\Desktop\\SEM_05"
*** error 65: access violation at C10x001A i f
>
ASM ASSIGN BreakDisable BreakEnable

```

Ready

AFG_BAN Live

Address:

Memory #1 Memory #2 Memory #3 Memory #4

Simulation t1: 0.00000000 sec CAP R/W

ENG IN 22:08 08-10-2025

B. Using Direct Addressing Mode:

Project Workspace

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x13
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC \$	C0x0000
states	1
sec	0x0000...
paw	0x00

```

C10x0000 7C87 MOV R4,#PCON(0x87)
      2: MOV PSW,#10H
C10x0002 75D010 MOV PSW(0xD0),#0x10
      3: MOV R3,#65H
      4:
C10x0005 7B65 MOV R3,#0x65
      5: MOV A,0x4H
C10x0007 E504 MOV A,0x04
      6: MOV B,13H
C10x0009 8513F0 MOV B(0xF0),#0x13
      7: MOV 04H,B
C10x000C 75D004 MOV 04H,B(0xF0)
      8: MOV 13H,A
C10x000F F513 MOV 0x13,A
      9:
C10x0011 00 NOP
C10x0012 00 NOP
C10x0013 00 NOP
C10x0014 00 NOP
C10x0015 00 NOP
C10x0016 00 NOP
C10x0017 00 NOP
C10x0018 00 NOP
C10x0019 00 NOP
C10x001A 00 NOP
C10x001B 00 NOP
C10x001C 00 NOP
C10x001D 00 NOP
C10x001E 00 NOP
C10x001F 00 NOP
C10x0020 00 NOP
      --:

```

P7_LED P6_13.asm P6_11.asm P6_2_led P6_12.asm Disassembly

Output Window

```

* Running with Code Size Limit: 2K
* Load "C:\\Users\\Hp\\OneDrive\\Desktop\\SEM_05"
*** error 65: access violation at C10x001A i f
>
ASM ASSIGN BreakDisable BreakEnable

```

Ready

Hot days ahead 24°C

Address:

Memory #1 Memory #2 Memory #3 Memory #4

Simulation t1: 0.00000050 sec CAP R/W

ENG IN 22:12 08-10-2025

Project Workspace

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x65
r4	0x13
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x65
b	0x65
sp	0x07
sp_max	0x07
dptr	0x0000
PC \$	C0x0000
states	10
sec	0x0000...
paw	0x10

```

C10x0000 7C87 MOV R4,#PCON(0x87)
      2: MOV PSW,#10H
C10x0002 75D010 MOV PSW(0xD0),#0x10
      3: MOV R3,#65H
      4:
C10x0005 7B65 MOV R3,#0x65
      5: MOV A,0x4H
C10x0007 E504 MOV A,0x04
      6: MOV B,13H
C10x0009 8513F0 MOV B(0xF0),#0x13
      7: MOV 04H,B
C10x000C 75D004 MOV 04H,B(0xF0)
      8: MOV 13H,A
C10x000F F513 MOV 0x13,A
      9:
C10x0011 00 NOP
C10x0012 00 NOP
C10x0013 00 NOP
C10x0014 00 NOP
C10x0015 00 NOP
C10x0016 00 NOP
C10x0017 00 NOP
C10x0018 00 NOP
C10x0019 00 NOP
C10x001A 00 NOP
C10x001B 00 NOP
C10x001C 00 NOP
C10x001D 00 NOP
C10x001E 00 NOP
C10x001F 00 NOP
C10x0020 00 NOP
      --:

```

P7_LED P6_13.asm P6_11.asm P6_2_led P6_12.asm Disassembly

Output Window

```

* Running with Code Size Limit: 2K
* Load "C:\\Users\\Hp\\OneDrive\\Desktop\\SEM_05"
*** error 65: access violation at C10x001A i f
>
ASM ASSIGN BreakDisable BreakEnable

```

Ready

Hot days ahead 24°C

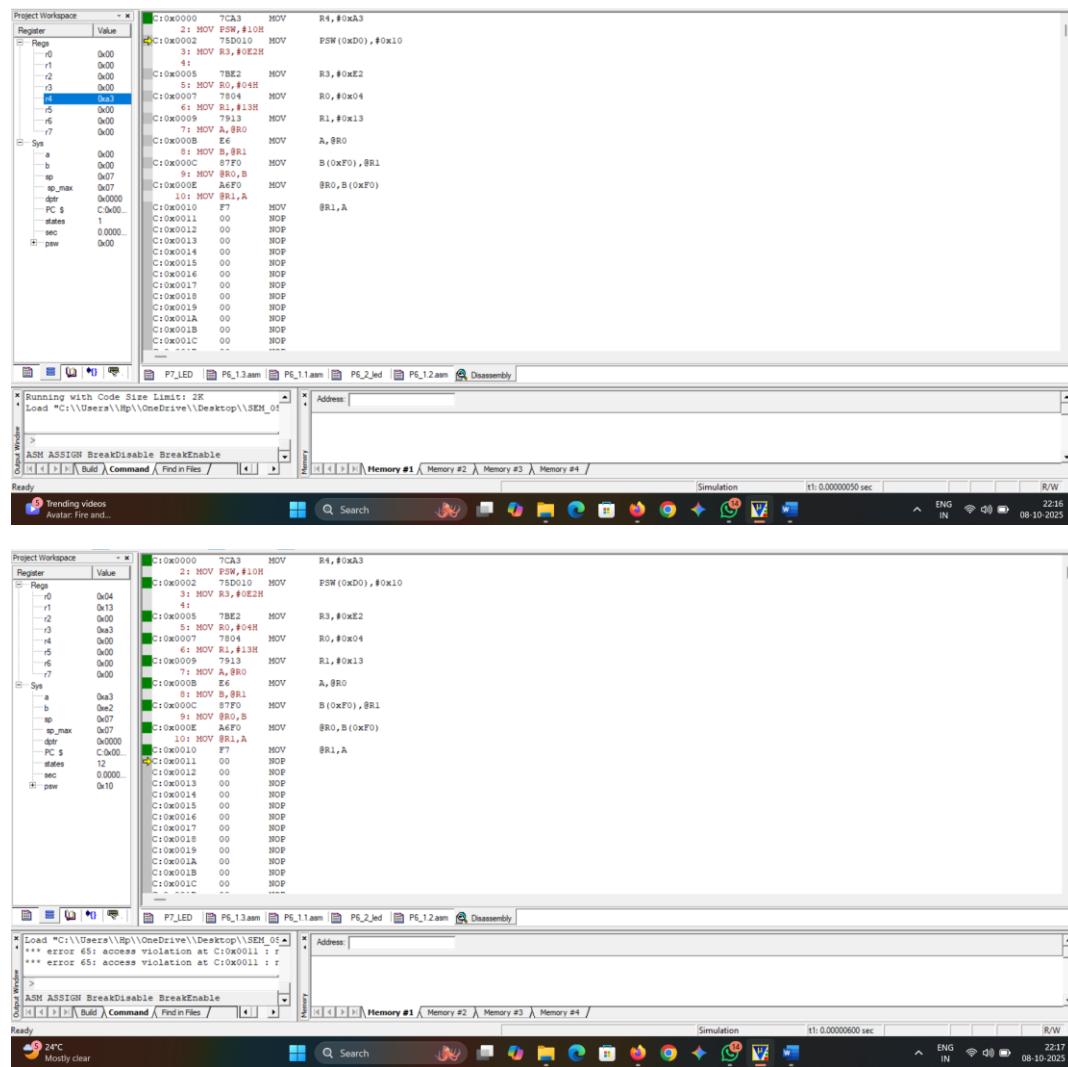
Address:

Memory #1 Memory #2 Memory #3 Memory #4

Simulation t1: 0.00000050 sec CAP R/W

ENG IN 22:13 08-10-2025

C. Using Indirect Addressing Mode:



2. Learning to create hex file, loading it into hardware and observe the output on real hardware. Observing the same in the Proteus software.

- (a) Write a sample program (using ‘c’ language) to switch ON the LEDs connected on port P1 one by one (without using bit level instructions) with some delay. After the successful execution in Keil software, create the hex file of it, load the hex file into 8051 IC using Proteus circuit simulator, do the necessary hardware connections and observe the output on it.

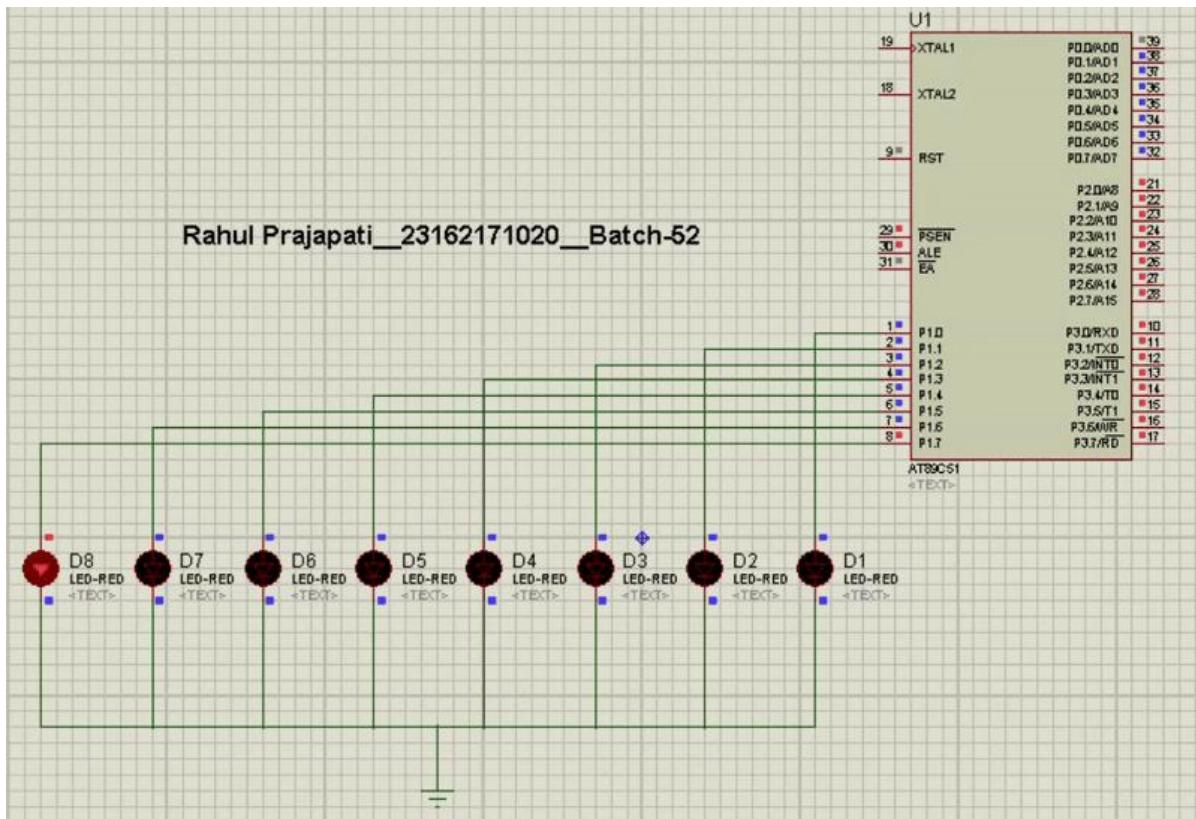
```

01 #include <reg51.h>
02 void delay(){
03     int i,j;
04     for (i=0;i<200;i++) { for (j=0;j<100;j++) {} }
05 }
06 void main(){
07     while(1){
08         P1 = 0x01;
09         delay();
10         P1 = 0x02;
11         delay();
12         P1 = 0x04;
13         delay();
14         P1 = 0x08;
15         delay();
16         P1 = 0x10;
17         delay();
18         P1 = 0x20;
19         delay();
20         P1 = 0x40;
21         delay();
22         P1 = 0x80;
23         delay();
24         P1 = 0xFF;
25         delay();
26     }
27 }

```

Build target 'Target 1'
linking...
Program Size: data=9.0 xdata=0 code=100
"practical" - 0 Error(s), 0 Warning(s).

→ SIMULATION: To see animation click on image .



CONCLUSION:

In this practical, we learned different addressing modes of the 8051 microcontroller by writing assembly programs to swap data between registers using register, direct, and indirect addressing. We also understood the process of creating a HEX file from C code in Keil, loading it into Proteus, and simulating real hardware behavior. This helped us practically understand 8051 programming, data handling, and hardware interfacing concepts.

Practical-6

1. Swap the contents of R4 of register bank 0 and R3 of register Bank 2 using all possible addressing modes.

→ A. Using Register Addressing Mode:

MOV R4, #26H ; load 26H into R4 of bank0

MOV PSW, #10H ; select reg. bank 2

MOV R3, #5BH ; load 5BH into R3 of bank 2

MOV PSW, #00H ; select bank 0

MOV A, R4 ; MOV R4(Bank0) to accumulator

MOV PSW, #10H ; switch to bank 2

MOV B, A ; store acc. content 26H in B reg.

MOV A, R3 ; move R3 (Bank 2) to accumulator.

MOV PSW, #00H ; switch to Bank 0

MOV R4, A ; store acc. content 5BH into R4(Bank0)

MOV PSW, #10H ; switch to register bank 2.

MOV R3, B ; store B(26H) into R3 (bank 2)

END ; end of program,

→ B. Using Direct Addressing mode:

```
MOV R4, #87H ; load 87H into R4 of bank 0  
MOV PSW, #10H ; switch to bank 2.  
MOV R3, #65H ; load 65H into R3 of bank 2.  
MOV A, OUTH ; move data from OUTH to Accumulator  
MOV B, 13H ; move data from 13H to register B.  
MOV OUT, B ; store content of B into address OUTH  
MOV 13H, A ; store content of A into address 13H  
END ; End of program.
```

→ C. Using indirect Addressing mode:

```
MOV R4, #0A3H ; Load 0A3H into RH (Bank 0)  
MOV PSW, #10H ; Select register bank 2  
MOV R3, #0E2H ; Load 0E2H into R3 (bank 2).  
MOV R0, #OUTH ; R0 points to OUTH  
MOV R1, #13H ; R1 points to 13H  
MOV A, @R0 ; move data from OUTH to A  
MOV B, @R1 ; move data from 13H to B  
MOV @R0, B ; store B into OUTH  
MOV @R1, A ; store A into 13H  
END ; End of program.
```