

**Department of Electronics and Communication Engineering**

**V Semester  
(Autonomous)**

**Microprocessors and Microcontrollers Lab manual  
Course Code: 19ECE02**

**(with effect from Academic year 2021-22)**

**(for B.Tech CSE)**

**GMR INSTITUTE OF TECHNOLOGY**

**(An Autonomous Institute Affiliated to JNTUK, Kakinada)**

**(Accredited by NBA, NAAC with grade 'A' & ISO 9001:2008 Certified Institution)**

Approved by AICTE, New Delhi

GMR NAGAR, RAJAM-532 127, A.P.

## **MICROPROCESSORS AND MICROCONTROLLERS**

### **LABORATORY MANUAL**

**Issued to: Dept of ECE**

**Copy No: \_\_\_\_\_ of \_\_\_\_\_**

<b>Prepared/verified by: Staff In charge</b>	<b>Approved by : HOD</b>
<b>Signature</b>	<b>Signature</b>
<b>Date:</b>	<b>Date:</b>

**Department of Electronics and Communication Engineering**

**GMR INSTITUTE OF TECHNOLOGY**

**(An Autonomous Institute Affiliated to JNTUK, Kakinada)**

(Accredited by NBA, NAAC with grade 'A' & ISO 9001:2008 Certified Institution)

Approved by AICTE, New Delhi

GMR NAGAR, RAJAM-532 127, A.P.

## **Microprocessors and Microcontrollers Lab**

**Course Code: 19ECE02, Regulation: AR19**

**B. Tech CSE, 5<sup>th</sup> Semester, A.Y: 2021-22**

### **List of experiments**

<b>Lab Number</b>	<b>Content</b>
<b>Experiments Based on 8086 Microprocessor using MASM IDE</b>	
1	Program for addition and subtraction in assembly language.
2	Program for data conversion in assembly language.
3	Data transfer program with and without using string instruction in assembly language programming.
4	Program to arrange three bytes in ascending and descending order in assembly language.
5	Program to reject negative numbers from a series of bytes in assembly language.
<b>Experiments are Based on 8051 Microcontroller using MCU8051 IDE</b>	
6	Program to perform arithmetic operations.
7	Program to toggle the LED in assembly language.
8	Programming and interfacing of traffic light logic in assembly language.
9	Programming and interfacing of the key pad matrix.
10	Programming and interfacing of seven-segment display.
11	Programming and interfacing of the LCD.

## **Demonstration of MASM Integrated Development Environment**

### **Introduction to MASM Assembler:**

The main advantage of machine language programming is that the memory control is directly in the hands of the programmer enabling him to manage the memory of the system more efficiently. However, there are more disadvantages. The programming, coding and resource management techniques are tedious. As the programmer has to consider all these functions, the chances of human errors are more. To understand the programs one has to have a thorough technical knowledge of the processor architecture and instruction set.

The assembly language programming is simpler as compared to the machine language programming. The instruction mnemonics are directly written in the assembly language programs. The advantage that assembly language has over machine language is that now the address values and the constants can be identified by labels. If the labels are clear then certainly the program will become more understandable, and each time the programmer will not have to remember the different constants and the addresses at which they are stored, throughout the programs.

An assembler is a program that converts an assembly language input file also called as source file to an object file that can further be converted into machine codes or an executable file using a linker. It decides the address of each label and substitutes the values for each of the constants and variables. Two types of assemblers are available, one is MASM (Microsoft Macro Assembler) and TASM (Turbo Assembler). MASM is the popular assemblers used along with a LINK program to structure the codes generated by MASM in the form of an executable file.

### **Programming with an Assembler:**

While writing a program for an assembler, the first step will be to use a text editor and type the program. Before quit the editor program, do not forget to save it. A number of text editors are available in the market, e.g. Norton's Editor [NE], Turbo C [TC], EDLIN, etc. Thus for writing a program in assembly language, one will need editor, MASM assembler, linker and DEBUG utility of DOS.

The main task of any assembler program is to accept the text assembly language program file as an input and prepare an object file. The MASM accepts the file names only with the extension .ASM. Even if a filename without any extension is given as input, it provides an .ASM extension to it.

Example: C> MASM KMB

or

C> MASM KMB.ASM

If the program contains syntax errors, they are displayed using error code number and the corresponding line number at which they appear. Once these syntax errors and warnings are taken care of by the programmer, the assembly process is completed successfully.

The DOS linking program LINK.EXE links the different object modules of a source program and function library routines to generate an integrated executable code of the source program. The main input to the linker is the .OBJ file that contains the object modules of the source programs.

Example: C> LINK KMB.OBJ

The output of the LINK program is an executable file with the entered filename and .EXE extension. This executable filename can further be entered at the DOS prompt to execute the file.

DEBUG.COM is a DOS utility that facilitates the debugging and trouble-shooting of assembly language programs.

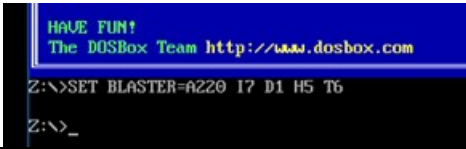

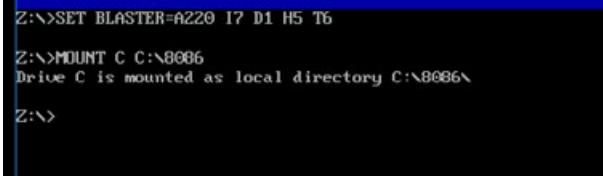
Example: C> DEBUG KMB.EXE

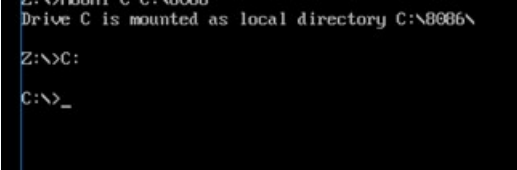
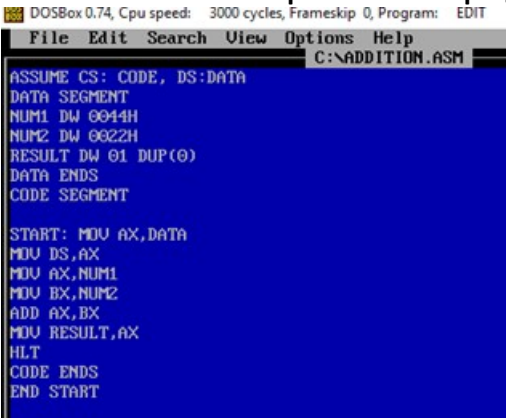
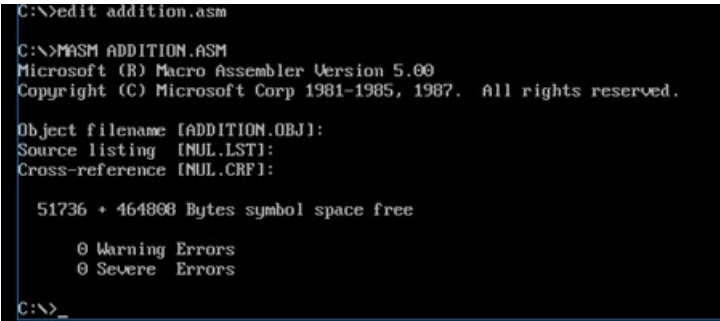
The DEBUG offers a good platform for trouble shooting, executing and observing the results of the assembly language programs.

**Table: DEBUG commands**

COMMAND CHARACTER	Format	Functions
-T	SEG:OFFSET <ENTER>	Trace the program execution by single stepping starting from the address SEG:OFFSET
-R	<ENTER>	Display all registers and flags
-R	<ENTER>	Display specified register contents and modify with the entered new contents.
-D	<ENTER>	Display 128 memory locations of RAM starting from the current display pointer.
-D	SEG:OFFSET1 OFFSET2<ENTER>	Display memory contents in SEG from

		OFFSET1 to OFFSET2.
-E	<ENTER>	Enter Hex data at current display pointer SEG:OFFSET
-E	SEG:OFFSET1 <ENTER>	Enter data at SEG:OFFSET1 byte by byte. The memory pointer is to be increased by space key, data entry is to be completed by pressing the <ENTER> key.
-F	SEG:OFFSET1    OFFSET2    BYTE <ENTER>	Fill the memory area starting from SEG:OFFSET1 to    OFFSET2 by the byte BYTE.
-F	SEG:OFFSET1    OFFSET2    BYTE1, BYTE2, BYTE3<ENTER>	Fill the memory area as abpve with the sequence BYTE1, BYTE2, BYTE3, ETC.
-A	<ENTER>	Assemble from the current CS:IP
-A	SEG:OFFSET <ENTER>	Assemble the entered instruction from SEG:OFFSET
-U	<ENTER>	Unassemble from the current CS:IP
-U	SEG:OFFSET <ENTER>	Unassemble from the address SEG:OFFSET

Step	MICROPROCESSOR LAB EXECUTION PROCEDURE
1	<p>Open "DOS BOX"</p> 
2	<p>Z:\&gt; mount c c: \8086 </p> 

3	<p>Z:\&gt; c:    ➡</p> 
4	<p>c:\&gt; edit name_of_file.asm    ➡</p> <p><b>“A asm file editor will open and write program in editor”</b></p>  <p>After writing the program: In the editor window, Select File → Save In the editor window, Select File → Exit</p>
5	<p>c:\&gt;masm name_of_file.asm    ➡</p> <p><b>“An object file will created with extension .obj”</b></p> 
	<p>c:\&gt;link name_of_file.obj    ➡</p>

**"An executable file will created with extension .exe"**

```
0 Severe Errors

C:\>LINK ADDITION.OBJ

Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.

Run File [ADDITION.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment

C:\>_
```

c:\debug name\_of\_file.exe ←

Type "t"

Single step execution of program will takes place

```
C:\>DEBUG ADDITION.EXE
-t
AX=076A BX=0000 CX=0022 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076B IP=0003 NU UP EI PL NZ NA PO NC
076B:0003 8ED8          MOV     DS,AX
-t
AX=076A BX=0000 CX=0022 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0005 NU UP EI PL NZ NA PO NC
076B:0005 A10000       MOV     AX,[0000]          DS:0000=0044
-t
```



## Experiment No-1

### Arithmetic Operations

- a) **AIM:** Write an ALP of 8086 microprocessor for addition of two 16-bit numbers. Use assembler directives to write the program.

#### APPARATURS:

1. PC with windows
2. MASM software

#### PROGRAM:

ASSUME CS:CODE, DS:DATA

DATA SEGMENT

OPR1 DW 3423H

OPR2 DW 6789H

RESULT DW 02H DUP(?)

DATA ENDS

CODE SEGMENT

START: MOV AX,DATA

MOV DS,AX

MOV AX,OPR1

MOV BX,OPR2

XOR CX,CX

ADD AX,BX

ADC CX,0000H

MOV [RESULT],AX

MOV [RESULT+2],CX

HLT

CODE ENDS

END START

**RESULTS:**

Before Execution of Program	
Memory Location	DATA (HEX)

After Execution of Program	
Memory Location	DATA (HEX)

**CONCLUSION:**

- b) AIM:** Write an ALP of 8086 microprocessor for subtraction of two 16-bit numbers. Use assembler directives to write the program.

**APPARATURS:**

1. PC with windows
2. MASM software

**PROGRAM:**

ASSUME CS:CODE, DS:DATA

DATA SEGMENT

OPR1 DW 3423H

OPR2 DW 6789H

RESULT DW 02H DUP(?)

DATA ENDS

CODE SEGMENT

START: MOV AX,DATA

MOV DS,AX

MOV AX,OPR1

MOV BX,OPR2

XOR CX,CX

SUB AX,BX

SBB CX,0000H

MOV [RESULT],AX

MOV [RESULT+2],CX

HLT

CODE ENDS

END START

**RESULTS:**

Before Execution of Program	
Memory Location	DATA (HEX)

After Execution of Program	
Memory Location	DATA (HEX)

**CONCLUSION:**

## Experiment No-2

### Data Conversion

**AIM:** Write an ALP of 8086 microprocessor to convert a packed BCD to ASCII number

#### APPARATURS:

1. PC with windows
2. MASM software

#### PROGRAM:

BCD	ASCII Equivalent
0	30H
1	31H
2	32H
3	33H
4	34H
5	35H
6	36H
7	37H
8	38H
9	39H

ASSUME CS:CODE, DS:DATA

DATA SEGMENT

N1 DB 45H

RESULT DW 01H DUP(?)

DATA ENDS

CODE SEGMENT

```
START: MOV AX,DATA
        MOV DS,AX
        MOV AL,N1
        MOV AH,AL
        AND AL,0FH
        AND AH,0F0H
        MOV CL,04H
        ROR AH,CL
        OR AX,3030H
        MOV RESULT,AX
        HLT
CODE ENDS
        END START
```

**RESULTS:**

Before Execution of Program	
Memory Location	DATA (HEX)

After Execution of Program	
Memory Location	DATA (HEX)

**CONCLUSION:**

### Experiment No-3

#### Data Transfer

- a) **AIM:** Write an ALP to transfer five bytes from one memory location to another without using string instruction. Use assembler directives to write the program.

#### APPARATURS:

1. PC with windows
2. MASM software

#### PROGRAM:

ASSUME CS:CODE, DS:DATA

DATA SEGMENT

NUMBERS DB 34H, 40H, 55H, 0A4H, 5FH

ORG 0200H

TRANSFER DB 05H DUP(?)

DATA ENDS

CODE SEGMENT

START: MOV AX, DATA

MOV DS, AX

MOV SI, OFFSET NUMBERS

MOV DI, OFFSET TRANSFER

MOV CL, 05H

AGAIN: MOV AL, [SI]

MOV [DI], AL

INC SI

INC DI



DEC CL

JNZ AGAIN

HLT

CODE ENDS

END START

**RESULTS:**

Before Execution of Program	
Memory Location	DATA (HEX)

After Execution of Program	
Memory Location	DATA (HEX)

**CONCLUSION:**

- b) AIM:** Write an ALP to transfer five bytes from one memory location to another location using string instruction. Use assembler directives to write the program.

**APPARATURS:**

1. PC with windows
2. MASM software

**PROGRAM:**

ASSUME CS:CODE, DS:DATA

DATA SEGMENT

NUMBERS DB 34H, 40H, 55H, 0A4H, 5FH

TRANSFER DB 05H DUP(?)

DATA ENDS

CODE SEGMENT

START: MOV AX, DATA

MOV DS, AX

MOV ES, AX

MOV SI, OFFSET NUMBERS

MOV DI, OFFSET TRANSFER

MOV CX, 0005H

REP MOVSB

HLT

CODE ENDS

END START

**RESULTS:**

Before Execution of Program	
Memory Location	DATA (HEX)

After Execution of Program	
Memory Location	DATA (HEX)

**CONCLUSION:**

## **Experiment No-4**

### **Sorting Operations**

**a) AIM:** Write an Assembly language program to sort a given data in ascending order.

#### **APPARATURS:**

1. PC with windows
2. MASM software

#### **PROGRAM:**

ASSUME DS:DATA, CS:CODE

DATA SEGMENT

LIST DB 56H,78H,34H,78H,26H,18H

DATA ENDS

CODE SEGMENT

START: MOV AX,DATA

MOV DS,AX

MOV CL,05H

L1: MOV DL,CL

MOV SI,OFFSET LIST

L2: MOV AL,[SI]

CMP AL,[SI+1]

JC NEXT

XCHG AL,[SI+1]

XCHG AL,[SI]

NEXT: INC SI

DEC DL

JNZ L2

DEC CL

JNZ L1

HLT

CODE ENDS

END START

**RESULTS:**

Before Execution of Program	
Memory Location	DATA (HEX)

After Execution of Program	
Memory Location	DATA (HEX)

--	--

**CONCLUSION:**

**b) AIM:** Write an Assembly language program to sort a given data in descending order.

**APPARATURS:**

1. PC with windows
2. MASM software

**PROGRAM:**

```
ASSUME DS:DATA, CS:CODE
```

```
DATA SEGMENT
```

```
LIST DB 56H,78H,34H,78H,26H,18H
```

```
DATA ENDS
```

```
CODE SEGMENT
```

```
START: MOV AX,DATA
```

```
MOV DS,AX
```

```
MOV CL,05H
```

```
L1:  MOV DL,CL
      MOV SI,OFFSET LIST

L2:  MOV AL,[SI]
      CMP AL,[SI+1]
      JNC NEXT
      XCHG AL,[SI+1]
      XCHG AL,[SI]

NEXT: INC SI
      DEC DL
      JNZ L2
      DEC CL
      JNZ L1
      HLT

CODE ENDS

END START
```

**RESULTS:**

Before Execution of Program	
Memory Location	DATA (HEX)

--	--

After Execution of Program	
Memory Location	DATA (HEX)

**CONCLUSION:**



### Experiment No-5

#### Reject negative numbers from a series of bytes

**AIM:** Write an ALP of 8086 microprocessor to reject negative numbers from a series of five bytes and store the positive numbers in memory

#### APPARATURS:

1. PC with windows
2. MASM software

#### PROGRAM:

ASSUME CS:CODE, DS:DATA

DATA SEGMENT

NUMBERS DB 34H, 40H, 85H, 0A3h, 70H

POSITIVE DB 05H DUP(?)

DATA ENDS

CODE SEGMENT

START: MOV AX, DATA

MOV DS, AX

MOV SI, OFFSET NUMBERS

MOV DI, OFFSET TRANSFER

MOV CX,0005H

REPEAT: MOV AL,[SI]

ROL AL,01H

JC REJECT

MOV [DI],AL

INC DI

REJECT:     INC SI  
  
              LOOP REPEAT  
  
CODE ENDS  
  
              END START

**RESULTS:**

Before Execution of Program	
Memory Location	DATA (HEX)

After Execution of Program	
Memory Location	DATA (HEX)

**CONCLUSION:**

**Experiment No-6**  
**Arithmetic Operations**

**a) AIM:** Write an ALP of 8051 microcontroller for addition of two 8-bit numbers

**APPARATURS:**

1. PC with windows
2. MCU8051

**PROGRAM:**

```
ORG 0000h

MOV R0,#30H

MOV R1,#40H

MOV A,@R0

INC R0

ADD A,@R0

MOV @R1,A

HERE: SJMP HERE

END
```

**RESULTS:**

Before Execution of Program	
Memory Location	DATA (HEX)

--	--

After Execution of Program	
Memory Location	DATA (HEX)

**CONCLUSION:**

**b) AIM:** Write an ALP of 8051 microcontroller for subtraction of two 8-bit numbers

**APPARATURS:**

1. PC with windows
2. MCU8051

**PROGRAM:**

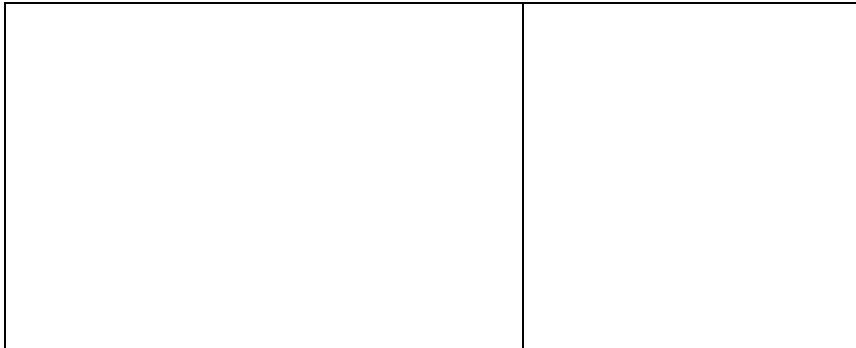
ORG 0000h

```
MOV R0,#30H  
MOV R1,#40H  
MOV A,@R0  
INC R0  
SUBB A,@R0  
MOV @R1,A  
HERE: SJMP HERE  
END
```

**RESULTS:**

Before Execution of Program	
Memory Location	DATA (HEX)

After Execution of Program	
Memory Location	DATA (HEX)



**CONCLUSION:**

c) **AIM:** Write an ALP of 8051 microcontroller for multiplication of two 8-bit numbers

**APPARATURS:**

1. PC with windows
2. MCU8051

**PROGRAM:**

```
ORG 0000h

MOV R0,#30H

MOV A,@R0

INC R0

MOV B,@R0

MUL AB

HERE: SJMP HERE

END
```

**RESULTS:**

Before Execution of Program	
Memory Location	DATA (HEX)

After Execution of Program	
Memory Location	DATA (HEX)

**CONCLUSION:**

**d) AIM:** Write an ALP of 8051 microcontroller for division of two 8-bit numbers

**APPARATURS:**

1. PC with windows
2. MCU8051

**PROGRAM:**

ORG 0000h

MOV R0,#30H

MOV A,@R0

INC R0

MOV B,@R0

DIV AB

HERE: SJMP HERE

END

**RESULTS:**

Before Execution of Program	
Memory Location	DATA (HEX)

After Execution of Program	
Memory Location	DATA (HEX)



--	--

**CONCLUSION:**

## **Experiment No-7**

### **Toggle LED**

**AIM:** Write an ALP of 8051 microcontroller to toggle a LED connected at P1.0 pin.

#### **APPARATURS:**

1. PC with windows
2. MCU8051

#### **PROGRAM:**

ORG 0000H

REPEAT:

SETB P1.0

LCALL DELAY

CLR P1.0

LCALL DELAY

SJMP REPEAT

ORG 0030H

DELAY: MOV R0,#03H

B2: MOV R1,#0F0H

B1: DJNZ R1, B1

DJNZ R0, B2

RET

END

#### **CONCLUSION:**

## Experiment No-8

### Traffic Light Interfacing

**AIM:** Write a ALP of 8051 microcontroller to interface traffic light logic

**APPARATURS:**

1. PC with windows
2. MCU8051

**Program**

The traffic light logic can be observed at P0.0 pin, P0.1 pin and P0.2 pin. Let Red LED is connected at P0.0, Yellow LED is connected at P0.1 and Green LED is connected at P0.2. Red and LED is ON for 15 seconds and Yellow LED is ON for 5 seconds.

ORG 0000H

REPEAT:

SETB P0.0 ; RED LED = ON

CLR P0.1 ; YELLOW LED = OFF

CLR P0.2 ; GREEN LED = OFF

LCALL DELAY10SEC

SETB P0.1 ; YELLOW LED = ON

LCALL DELAY5SEC

CLR P0.0 ; RED LED = OFF

CLR P0.1 ; YELLOW LED = OFF

SETB P0.2 ; GREEN LED = ON

LCALL DELAY10SEC

SETB P0.1 ; YELLOW LED = ON

LCALL DELAY5SEC

SJMP REPEAT

ORG 0030H

DELAY10SEC:MOV TMOD,#10H

MOV R0,#200D

L1: MOV TH1,#4BH

MOV TL1,#0FDH

SETB TR1

B1: JNB TF1,B1

CLR TR1

CLR TF1

DJNZ R0,L1

RET

ORG 0090H

DELAY5SEC:MOV TMOD,#10H

MOV R0,#100D

L2: MOV TH1,#4BH

MOV TL1,#0FDH

SETB TR1

B2: JNB TF1,B2

CLR TR1

CLR TF1

DJNZ R0,L2

RET

END

**CONCLUSION:**

## Experiment No-9

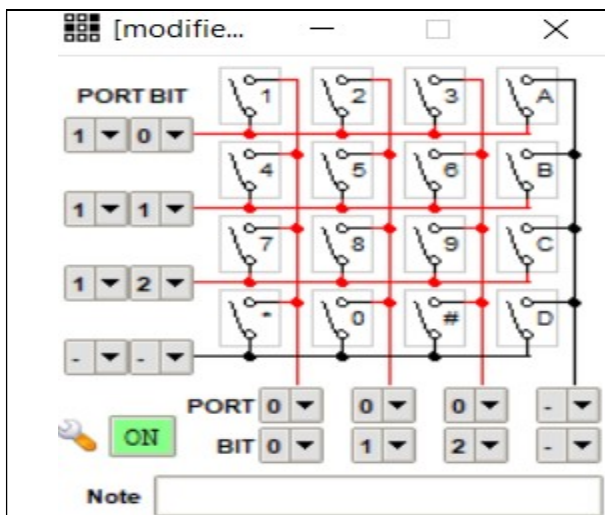
### Key-pad Matrix Interfacing

**AIM:** Interface 8051 microcontroller with a key pad matrix.

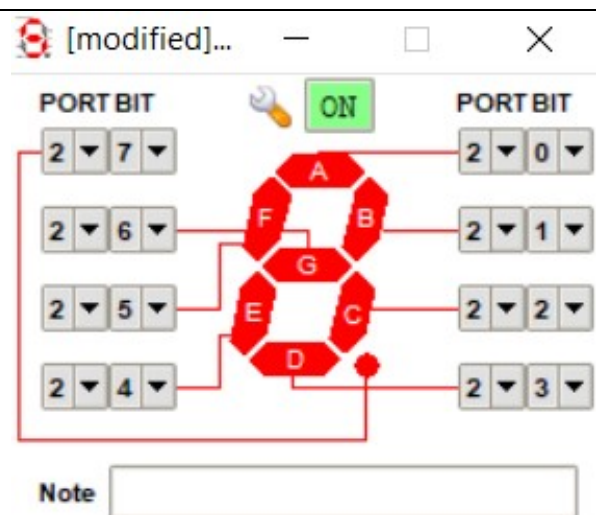
#### APPARATURS:

1. PC with windows
2. MCU8051

#### INTERFACING DIAGRAM:



**Fig. 1: Interfacing of 3x3 key matrix with 8051**



**Fig. 2: Interfacing of CC 7-Segment Display with 8051**

**Instruction for Execution: Execute in single step execution mode by pressing F7 key every time.**

**PROGRAM:**

```
ORG 0000H

REP:LCALL DELAY

MOV P2,#00H ; 7-segment display off

MOV P1,#00H

SCAN:

JNB P0.0,COL_1 ;if key in col.1 is pressed then go to label COL_1
JNB P0.1,COL_2 ;if key in col.2 is pressed then go to label COL_2
JNB P0.2,COL_3 ;if key in col.3 is pressed then go to label COL_3
SJMP SCAN    ; if no key is pressed then go to label SCAN


COL_1:

MOV P1,#11111110B

JNB P0.0,ONE

MOV P1,#11111101B

JNB P0.0,FOUR

MOV P1,#11111011B

JNB P0.0,SEVEN

COL_2:

MOV P1,#11111110B

JNB P0.1,TWO

MOV P1,#11111101B
```

```
JNB P0.1,FIVE  
MOV P1,#11111011B  
JNB P0.1,EIGHT  
COL_3:  
MOV P1,#11111110B  
JNB P0.2,THREE  
MOV P1,#11111101B  
JNB P0.2,SIX  
MOV P1,#11111011B  
JNB P0.2,NINE  
ONE:MOV P2, #06H ;Control word to display 1  
SJMP REP  
TWO:MOV P2,#5BH ;Control word to display 2  
SJMP REP  
THREE:MOV P2,#4FH ;Control word to display 3  
SJMP REP  
FOUR:MOV P2,#66H ;Control word to display 4  
SJMP REP  
FIVE:MOV P2,#6DH ;Control word to display 5  
SJMP REP  
SIX:MOV P2,#7DH ;Control word to display 6  
SJMP REP  
SEVEN:MOV P2,#07H ;Control word to display 7  
SJMP REP
```



EIGHT:MOV P2,#7FH ;Control word to display 8

SJMP REP

NINE:MOV P2,#67H ;Control word to display 9

SJMP REP

ORG 0100h

DELAY:MOV TMOD,#20H

MOV TH1,#00H

MOV TL1,#00H

SETB TR1

B1:JNB TF1,B1

CLR TF1

RET

END

**CONCLUSION:**

## Experiment No-10

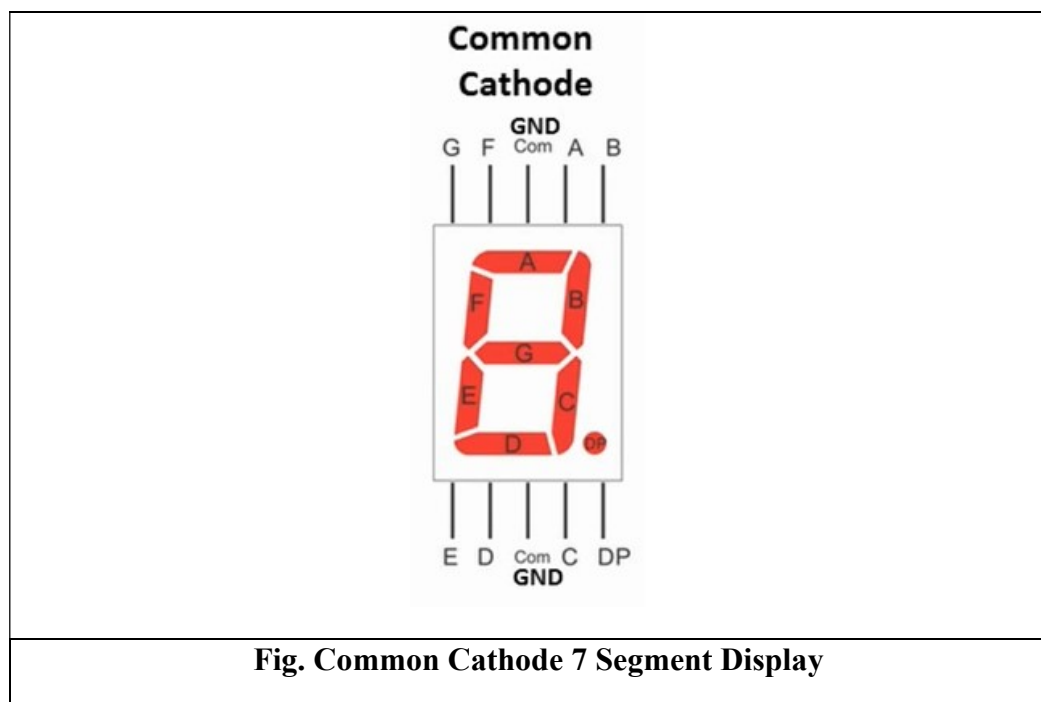
### Seven Segment Display Interfacing

**AIM:** Interface 8051 microcontroller with a common cathode Seven segment display and develop a ALP to display numbers from 0 to 9

#### APPARATURS:

1. PC with windows
2. MCU8051

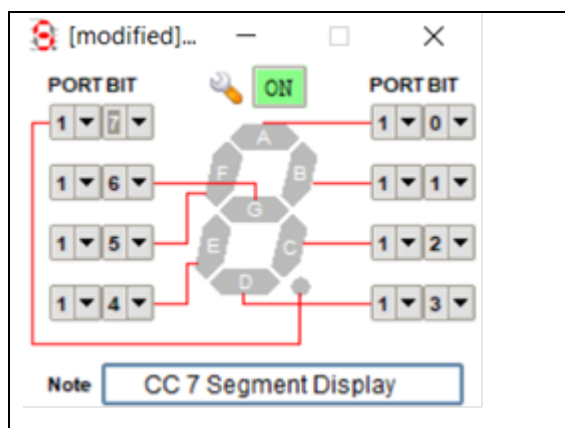
**Develop the control words to display numbers from 0 to 9 in a common cathode Seven segment display**



**Common Cathode Connection and Control word:**

7	6	5	4	3	2	1	0	Pins of Port 1	
Segments of Display									
Dp	g	f	e	d	c	b	a	Number to display	Control word in HEX for display
0	0	1	1	1	1	1	1	0	3FH
0	0	0	0	0	1	1	0	1	06H
0	1	0	1	1	0	1	1	2	5BH
0	1	0	0	1	1	1	1	3	4FH
0	1	1	0	0	1	1	0	4	66H
0	1	1	0	1	1	0	1	5	6DH
0	1	1	1	1	1	0	1	6	7DH
0	0	0	0	0	1	1	1	7	07H
0	1	1	1	1	1	1	1	8	7FH
0	1	1	0	0	1	1	1	9	67H

**INTERFACING DIAGRAM:**



**Fig. 1: CC 7 Segment Interfacing with Port 1**

**PROGRAM:**

```
ORG 0000h

REPEAT: MOV P1,#3FH ; DISPLAY 0

LCALL DELAY

MOV P1,#06H ; DISPLAY 1

LCALL DELAY

MOV P1,#5BH ; DISPLAY 2

LCALL DELAY

MOV P1,#4FH ; DISPLAY 3

LCALL DELAY

MOV P1,#66H ; DISPLAY 4

LCALL DELAY

MOV P1,#6DH ; DISPLAY 5

LCALL DELAY

MOV P1,#7DH ; DISPLAY 6

LCALL DELAY

MOV P1,#07H ; DISPLAY 7

LCALL DELAY

MOV P1,#7FH ; DISPLAY 8

LCALL DELAY
```

```
MOV P1,#67H ; DISPLAY 9
LCALL DELAY
HERE: SJMP REPEAT
DELAY:    MOV TMOD,#10H
MOV R0,#10H
LOOP1:MOV TH1,#3CH
        MOV TL1,#0B0H
        SETB TR1
BACK1:JNB TF1,BACK1
        CLR TR1
        CLR TF1
        DJNZ R0,LOOP1
        RET
        END
```

**CONCLUSION:**

### Experiment No-11

**AIM:** Interfacing of LCD with 8051 microcontroller and display the message.

**APPARATURS:**

1. PC with windows
2. MCU8051

**PROGRAM:**

ORG 0000H

LCALL COMANDWR1

LCALL DATAWR1

HERE:SJMP HERE

ORG 0020H

COMANDWR1: MOV DPTR,#COMMANDWORD1

REP1: CLR A

MOVC A,@A+DPTR

JZ LAST1

MOV P3,A

CLR P2.0

CLR P2.1

SETB P2.2

LCALL DELAY

CLR P2.2

INC DPTR

SJMP REP1

LAST1:RET

ORG 0050H

DATAWR1: MOV DPTR,#DATAWORD1

REP2: CLR A

MOVC A,@A+DPTR

JZ LAST2

MOV P3,A

SETB P2.0

CLR P2.1

SETB P2.2

LCALL DELAY

CLR P2.2

INC DPTR

SJMP REP2

LAST2:RET

ORG 00E0H

COMMANDWORD1:DB 38H,0FH,80H,00H

ORG 00F0H

DATAWORD1:DB 'GMRIT RAJAM',00H

ORG 0130H

DELAY:MOV R1,#255D

B1:DJNZ R1,B1

RET

END

**CONCLUSION:**