Experiment No.1

1. PROGRAMS FOR ARITHMETIC OPERATION

AIM : Programs for 16 bit arithmetic operations of 8086 (using various addressing modes)

EQUIPMENT REQUIRED:

1. TASM Software
2. PC with DOS and Debug program

ALGORITHM:

1. Define the values in data segment as per the addressing mode.
2. Initialize the data segment register with data segment address
3. Load the words as per the addressing mode and perform addition/ subtraction/ multiplication/ division and store the sum/ difference/product/quotient-remainder to the result address
4. Terminate the program

PROGRAMS (using register addressing mode):

A. 16 BIT ADDITION

```
 1                          assume cs:code,ds:data
 2
 3  0000                    data segment
 4  0000  1243              n1   dw 1243h
 5  0002  4567              n2   dw 4567h
 6  0004  ????              n3   dw ?
 7  0006                    data ends
 8
 9  0000                    code segment
10
11  0000                   start:
12  0000  B8 0000s          mov ax,data
13  0003  8E D8             mov ds,ax
14
15  0005  A1 0000r          mov ax,n1
16  0008  8B 1E 0002r       mov bx,n2
17  000C  03 C3             add ax,bx
18  000E  A3 0004r          mov n3,ax
19  0011  BE 0004r          lea si,n3
20  0014  CC                int 3
21
22  0015                    code ends
23                          end start
```

B. 16 BIT SUBTRACTION

```
 1                              assume cs:code,ds:data
 2
 3  0000                        data segment
 4  0000  FFFF                  n1   dw 0ffffh
 5  0002  4567                  n2   dw 4567h
 6  0004  ????                  n3   dw ?
 7  0006                        data ends
 8
 9  0000                        code segment
10
11  0000                        start:
12  0000  B8 0000s              mov ax,data
13  0003  8E D8                 mov ds,ax
14
15  0005  A1 0000r              mov ax,n1
16  0008  8B 1E 0002r           mov bx,n2
17  000C  2B C3                 sub ax,bx
18  000E  A3 0004r              mov n3,ax
19  0011  BE 0004r              lea si,n3
20  0014  CC                    int 3
21
22  0015                        code ends
23                              end start
```

C. 16 BIT MULTIPLICATION

```
 1                              assume cs:code,ds:data
 2
 3  0000                        data segment
 4  0000  4444                  n1   dw 4444h
 5  0002  4567                  n2   dw 4567h
 6  0004  ????????              n3   dd ?
 7  0008                        data ends
 8
 9  0000                        code segment
10
11  0000                        start:
12  0000  B8 0000s              mov ax,data
13  0003  8E D8                 mov ds,ax
14
15  0005  A1 0000r              mov ax,n1
16  0008  8B 1E 0002r           mov bx,n2
17  000C  F7 E3                 mul bx
18  000E  BE 0004r              lea si,n3
19  0011  89 04                 mov [si],ax
20  0013  89 54 02              mov [si+2],dx
```

```
21
22  0016  CC              int 3
23
24  0017                  code ends
25                        end start
```

D. WORD BY BYTE DIVISION

```
 1                        assume cs:code,ds:data
 2
 3  0000                  data segment
 4  0000  0444            n1   dw 0444h
 5  0002  45              n2   db 45h
 6  0003  ????            n3   dw ?
 7  0005                  data ends
 8
 9  0000                  code segment
10
11  0000                  start:
12  0000  B8 0000s        mov ax,data
13  0003  8E D8           mov ds,ax
14
15  0005  A1 0000r        mov ax,n1
16  0008  8A 1E 0002r     mov bl,n2
17  000C  F6 F3           div bl
18
19  000E  A3 0003r        mov n3,ax
20  0011  BE 0003r        lea si,n3
21
22  0014  CC              int 3
23
24  0015                  code ends
25                        end start
```

RESULT:

A. 16 BIT ADDITION

AX= 57AA & SI=0004 ; D 0004 0005 AA 57

B. 16 BIT SUBTRACTION

AX= BA98 & SI=0004 ; D 0004 0005 98 BA

C. 16 BIT MULTIPLICATION

AX= CB5C & SI=0004 ; D 0000 0005 44 44 67 45 5C CB

D. WORD BY BYTE DIVISION

AX= 390F & SI=0003 ; D 0000 0004 44 04 45 0F 39

PROGRAMS (using indirect register addressing mode):

E. 16 BIT ADDITION

```
 1                          assume cs:code,ds:data
 2
 3  0000                    data segment
 4  0000  0444              n1   dw 0444h
 5  0002  4545              n2   dw 4545h
 6  0004  ????              n3   dw ?
 7  0006                    data ends
 8
 9  0000                    code segment
10
11  0000                    start:
12  0000  B8 0000s          mov ax,data
13  0003  8E D8             mov ds,ax
14
15  0005  BE 0000r          lea si,n1
16  0008  BF 0002r          lea di,n2
17  000B  8B 04             mov ax,[si]
18  000D  8B 1D             mov bx,[di]
19  000F  03 C3             add ax,bx
20
21  0011  BD 0004r          lea bp,n3
22  0014  89 46 00          mov [bp],ax
23
24
25
26  0017  CC                int 3
27
28  0018                    code ends
29                          end start
```

F. 16 BIT SUBTRACTION

```
 1                          assume cs:code,ds:data
 2
 3   0000                   data segment
 4   0000  AAAA             n1   dw 0aaaah
 5   0002  4545             n2   dw 4545h
 6   0004  ????             n3   dw ?
 7   0006                   data ends
 8
 9   0000                   code segment
10
11   0000                   start:
12   0000  B8 0000s         mov ax,data
13   0003  8E D8            mov ds,ax
14
15   0005  BE 0000r         lea si,n1
16   0008  BF 0002r         lea di,n2
17   000B  8B 04            mov ax,[si]
18   000D  8B 1D            mov bx,[di]
19   000F  2B C3            sub ax,bx
20
21   0011  BD 0004r         lea bp,n3
22   0014  89 46 00         mov [bp],ax
23
24
25
26   0017  CC               int 3
27
28   0018                   code ends
29                          end start
```

RESULT:

E. 16 BIT ADDITION

AX= 4989 , SI=0000 , DI=0002, BP=0004
D 0004 0005 89 49

F. 16 BIT SUBTRACTION

AX= 65 65 , SI=0000 , DI=0002, BP=0004 D 0004 0005 65 65

PROGRAMS (using direct addressing mode):

G. 16 BIT ADDITION

```
 1                          assume cs:code,ds:data
 2
 3  0000                    data segment
 4  0000  5AAA              n1   dw 5aaah
 5  0002  4545              n2   dw 4545h
 6  0004  ????              n3   dw ?
 7  0006                    data ends
 8
 9  0000                    code segment
10
11  0000                    start:
12  0000  B8 0000s          mov ax,data
13  0003  8E D8             mov ds,ax
14
15  0005  A1 0000r          mov ax,n1
16  0008  03 06 0002r       add ax,n2
17  000C  A3 0004r          mov n3,ax
18  000F  BE 0004r          lea si,n3
19
20  0012  CC                int 3
21
22  0013                    code ends
23                          end start
```

H. 16 BIT SUBTRACTION

```
 1                          assume cs:code,ds:data
 2
 3  0000                    data segment
 4  0000  5AAA              n1   dw 5aaah
 5  0002  4545              n2   dw 4545h
 6  0004  ????              n3   dw ?
 7  0006                    data ends
 8
 9  0000                    code segment
10
11  0000                    start:
12  0000  B8 0000s          mov ax,data
13  0003  8E D8             mov ds,ax
14
15  0005  A1 0000r          mov ax,n1
16  0008  2B 06 0002r       sub ax,n2
17  000C  A3 0004r          mov n3,ax
```

```
18  000F  BE 0004r      lea si,n3
19
20  0012  CC            int 3
21
22  0013                code ends
23                      end start
```

RESULT:

G. 16 BIT ADDITION

AX= 9FEF , SI=0004 ; D 0004 0005 EF 9F

H. 16 BIT SUBTRACTION

AX= 6565 , SI=0004 ; D 0004 0005

PROGRAMS (using immediate addressing mode):

I.16 BIT ADDITION

```
 1                      assume cs:code,ds:data
 2
 3  0000                data segment
 4  0000  ????          n3   dw ?
 5  0002                data ends
 6
 7  0000                code segment
 8
 9  0000                start:
10  0000  B8 0000s      mov ax,data
11  0003  8E D8         mov ds,ax
12
13  0005  B8 6666       mov ax,6666h
14  0008  05 7788       add ax,7788h
15  000B  A3 0000r      mov n3,ax
16  000E  BE 0000r      lea si,n3
17
18  0011  CC            int 3
19
20  0012                code ends
21                      end start
```

J. 16 BIT SUBTRACTION

```
 1                          assume cs:code,ds:data
 2
 3  0000                    data segment
 4  0000  ????              n3   dw ?
 5  0002                    data ends
 6
 7  0000                    code segment
 8
 9  0000                    start:
10  0000  B8 0000s          mov ax,data
11  0003  8E D8             mov ds,ax
12
13  0005  B8 ABCD           mov ax,0abcdh
14  0008  2D 7893           sub ax,7893h
15  000B  A3 0000r          mov n3,ax
16  000E  BE 0000r          lea si,n3
17
18  0011  CC                int 3
19
20  0012                    code ends
21                          end start
```

RESULT:

I. 16 BIT ADDITION

AX= DDEE , SI=0000 ; D 0000 0001 EE DD

J. 16 BIT SUBTRACTION

AX= 333A , SI=0000 ; D 0000 0001 3A 33

VIVA QUESTIONS:

1. What is need for initializing the data segment register?
2. What is an interrupt?
3. What are DOS interrupts?
4. What is int 3 ?
5. What are the data definition directives?
6. What are interrupt vectors?
7. What is interrupt vector table?
8. What are bios interrupts?
9. Explain the organization of system memory?
10. What is the syntax of signed multiply instruction?
11. What is the use of END directive**?**

12. What is the syntax of unsigned division instruction?
13. What is the logic for division without using div instruction?
14. What is the implicit register for dividend when the divisor is of type byte and how the result is stored ?
15. What is the implicit register for dividend when the divisor is of word and how the result is stored ?

Experiment No.2

2. PROGRAM FOR SORTING AN ARRAY FOR 8086

A. ASCENDING ORDER

AIM : Program to sort the numbers in ascending order

EQUIPMENT REQUIRED:

1. TASM Software
2. PC with DOS and Debug program

ALGORITHM:

1. Define the values in data segment
2. Initialize the data segment register with data segment address
3. Clear the various registers
4. Initialize outer counter for arranging the given numbers
5. Initialize inner counter for performing comparisons
6. Compare the first two values, if carry is generated then continue for next values
7. Otherwise, exchange both values and continue for next values
8. Continue from step 5 till the count is zero.
9. Terminate the program

PROGRAM:

```
    1                                       assume cs:code,ds:data
 2  0000                                    data segment
 3
 4  0000  0198 0135      0234 0098          n1  dw 198h,135h,234h,098h
 5  0008  0A*(0000)                         res dw 10 dup(0)
 6        =0003                             count equ 3
 7
 8  001C                                    data ends
 9  0000                  code segment
10  0000                  start:
11  0000  B8 0000s        mov ax,data
12  0003  8E D8           mov ds,ax
13
14  0005  33 C0           xor ax,ax
15  0007  33 D2           xor dx,dx
16  0009  33 C9           xor cx,cx
17
18  000B  BA 0003         mov dx,count
19  000E  B9 0003         x1:mov cx,count
20  0011  BE 0000r        lea si,n1
```

```
21  0014  8B 04          mov ax,[si]
22  0016  3B 44 02       x:cmp ax,[si+2]
23  0019  72 05          jc   l1
24  001B  87 44 02       xchg ax,[si+2]
25  001E  89 04          mov [si],ax
26  0020  46             l1:inc si
27  0021  46             inc si
28  0022  8B 04          mov ax,[si]
29  0024  E2 F0          loop x
30  0026  4A             dec dx
31  0027  75 E5          jnz x1
32
33  0029  BE 0000r       lea si,n1
34
35  002C  CC             int 3h
36  002D                 code ends
37                       end start
```

RESULT:
AX=0234 , SI=0000
D 0000 0007 98 00 35 01 98 01 34 02

B. DESCENDING ORDER

AIM : Program to sort the numbers in descending order

EQUIPMENT REQUIRED:

1. TASM Software
2. PC with DOS and Debug program

ALGORITHM:

1. Define the values in data segment
2. Initialize the data segment register with data segment address
3. Clear the various registers
4. Initialize outer counter for arranging the given numbers
5. Initialize inner counter for performing comparisons
6. Compare the first two values, if no carry is generated then continue for next values
7. Otherwise, exchange both values and continue for next values
8. Continue from step 5 till the count is zero.
9. Terminate the program

PROGRAM:

```
 1                                          assume cs:code,ds:data
 2  0000                                    data segment
 3
 4  0000  0198 0135      0234 0098         n1 dw 198h,135h,234h,098h
 5  0008  0A*(0000)                         res dw 10 dup(0)
 6        =0003                             count equ 3
 7
 8  001C                                    data ends
 9  0000                  code segment
10  0000                  start:
11  0000  B8 0000s        mov ax,data
12  0003  8E D8           mov ds,ax
13
14  0005  33 C0           xor ax,ax
15  0007  33 D2           xor dx,dx
16  0009  33 C9           xor cx,cx
17
18  000B  BA 0003         mov dx,count
19  000E  B9 0003         x1:mov cx,count
20  0011  BE 0000r        lea si,n1
21  0014  8B 04           mov ax,[si]
22  0016  3B 44 02        x:cmp ax,[si+2]
23  0019  73 05           jnc l1
24  001B  87 44 02        xchg ax,[si+2]
25  001E  89 04           mov [si],ax
26  0020  46              l1:inc si
27  0021  46              inc si
28  0022  8B 04           mov ax,[si]
29  0024  E2 F0           loop x
30  0026  4A              dec dx
31  0027  75 E5           jnz x1
32
33  0029  BE 0000r        lea si,n1
34
35  002C  CC              int 3h
36  002D                  code ends
37                        end start
```

RESULT:

AX=0098 , SI=0000

D 0000 0007 34 02 98 01 35 01 98 00

VIVA QUESTIONS:

1. Give the concept of Jump with return and jump with non return.
2. What are the flags that are effected by compare statement?
3. What is the Significance of inserting label in programming.
4. What is the Significance of int 3h.
5. What is the purpose of offset?

Experiment No.3

3. PROGRAM FOR SEARCHING FOR A NUMBER OR CHARACTER IN A STRING FOR 8086

AIM : Program for searching for a number or character in string for 8086

EQUIPMENT REQUIRED:

1. TASM Software
2. PC with DOS and Debug program

ALGORITHM:

1. Define the values in data segment
2. Initialize the data segment register with data segment address
3. Clear all the various registers
4. Initialize the counter for number of comparisons
5. Compare the input with the numbers in an array one at a time
6. If zero flag is set, display the message 'number found'
7. If zero flag is not set even after all the comparisons i.e., till the count is zero then display the message 'number not found'
8. Terminate the program

PROGRAM:

```
 1                                      assume cs:code,ds:data
 2
 3  0000                                data segment
 4  0000  12CD 3BCD    34CD             n1      dw 12cdh,3bcdh,34cdh
 5  0006  04CD                          n2      dw 04cdh
 6  0008  70 61 73 73 77 6F       72+   msg1 db "number found
                                                $"
 7        64 20 66 6F 75 6E 64+
 8        20 24
 9  0018  70 61 73 73 77 6F       72+   msg2 db "number not
                                                found $"
10        64 20 6E 6F 74 20 66+
11        6F 75 6E 64 20 24
12        =0003                         count equ 3
13  002C                                data ends
14
15  0000                  code segment
16  0000                  start:
17  0000  B8 0000s        mov ax,data
18  0003  8E D8           mov ds,ax
19  0005  33 C0           xor ax,ax
```

```
20   0007  33 D2          xor dx,dx
21   0009  33 C9          xor cx,cx
22   000B  B9 0003        mov cx,count
23   000E  BE 0000r       lea si,n1
24   0011  A1 0006r       mov ax,n2
25   0014  3B 04          l1:cmp ax,[si]
26   0016  74 0E          jz   l2
27   0018  46             inc si
28   0019  46             inc si
29   001A  E2 F8          loop l1
30   001C  B4 09          mov ah,09h
31   001E  BA 0018r       lea dx,msg2
32   0021  CD 21          int 21h
33   0023  EB 08 90       jmp l3
34   0026  B4 09          l2:mov ah,09h
35   0028  BA 0008r       lea dx,msg1
36   002B  CD 21          int 21h
37   002D  CC             l3:int 3h
38   002E                 code ends
39                        end start
```

RESULT:

Input of 04cdh, the message password found is displayed.

Input of 2340h, the message password not found is displayed.

Experiment No.4

4. PROGRAM FOR STRING MANIPULATION FOR 8086

A. MOVE THE STRING

AIM : Program to move the string from source location to destination location

EQUIPMENT REQUIRED:

1. TASM Software
2. PC with DOS and Debug program

ALGORITHM:

1. Define the string to be displayed in data segment
2. Initialize the data segment register with data segment address
3. Initialize the source pointer to the starting address of defined string
4. Initialize the destination pointer to the location where the string is to be stored
5. Move the contents from the source to the destination till the '$' (termination character) is found
6. Display the string from the destination location
7. Terminate the program

PROGRAM:

```
                                        assume cs:code,ds:data

0000                                    data segment

0000  20 68 65 6C 6C 6F      20+        msg db " hello everyone","$"
      65 76 65 72 79 6F 6E+
      65 24

0010  0A 0D 24                          nline db 10,13,'$'

0013  28*(24)                           msg1 db 40 dup('$')

003B                                    data ends



0000                                    code segment

0000  B8 0000s                          start:mov ax,data
0003  8E D8                             mov ds,ax
```

```
21
22  0005  B4 09              mov ah,09h
23  0007  BA 0000r           lea dx,msg
24  000A  CD 21              int 21h
25
26  000C  BE 0000r           lea si,msg
27  000F  BF 0013r           lea di,msg1
28
29  0012  8A 04              l1: mov al,[si]
30  0014  3C 24              cmp al,'$'
31  0016  74 06              je l2
32  0018  88 05              mov [di],al
33  001A  46                 inc si
34  001B  47                 inc di
35  001C  EB F4              jmp l1
36
37  001E  B4 09              l2:mov ah,09h
38  0020  BA 0010r           lea dx,nline
39  0023  CD 21              int 21h
40
41  0025  B4 09              mov ah,09h
42  0027  BA 0013r           lea dx,msg1
43  002A  CD 21              int 21h
44
45  002C  B4 4C              mov ah,4ch
46  002E  CD 21              int 21h
47
48  0030                     code ends
49                           end start
```

RESULT: hello everyone

VIVA QUESTIONS:

1. Specify branch instructions.
2. What are conditional branch instructions?
3. What is the syntax of compare instruction?
4. What are the flags affected in compare instruction?
5. What is the total memory addressing capability of 8086 processor?

B.REVERSE THE STRING READ FROM THE KEYBOARD AND DISPLAY

AIM : Program to reverse the string read from the keyboard and display

EQUIPMENT REQUIRED:

1. TASM Software
2. PC with DOS and Debug program

ALGORITHM:

1. Define the prompt messages to be displayed in data segment
2. Initialize the data segment register with data segment address
3. Use 0ah function to read the string from the keyboard
4. Initialize the source pointer to the end of the read string
5. Initialize the destination pointer to the location where the reversed string is to be stored
6. Initialize the counter to the actual length of the entered string
7. Copy the contents from the source to the destination till the counter is zero
8. Display the reversed string
9. Terminate the program

PROGRAM:

```
                                assume cs:code,ds:data

0000                            data segment

0000  20 65 6E 74 65 72 20+     msg db " enter the string :  ","$"
      74 68 65 20 73 74 72+
      69 6E 67 20 3A 20 24

0015                            str1 label byte
0015  14                        strmax db 20
0016  ??                        stract db ?
0017  0A*(24)                   strfld db 10 dup('$')

0021  0A 0D 24                  nline db 10,13,'$'

0024  0A*(24)                   rev db 10 dup('$')

002E  20 74 68 65 20 72 65+     msg1 db " the reversed string   is  : " , "$"
      76 65 72 73 65 64 20+
```

```
                73 74 72 69 6E 67 20+
                69 73 20 20 3A 20 24


004A                            data ends


0000                            code segment

0000  B8 0000s                  start:  mov ax,data
0003  8E D8                             mov ds,ax

0005  B4 09                             mov ah,09h
0007  BA 0000r                          lea dx,msg
000A  CD 21                             int 21h

000C  B4 0A                             mov ah,0ah
000E  BA 0015r                          lea dx,str1
0011  CD 21                             int 21h

0013  B4 09                             mov ah,09h
0015  BA 0021r                          lea dx,nline
0018  CD 21                             int 21h

001A  BE 0017r                          lea si,strfld
001D  BF 0024r                          lea di,rev

0020  33 C9                             xor cx,cx
0022  8A 0E 0016r                       mov cl,stract

0026  03 F1                             add si,cx
0028  4E                                dec si


0029  8A 04                     top:    mov al,[si]
002B  88 05                             mov [di],al
002D  47                                inc di
002E  4E                                dec si
002F  E2 F8                             loop    top


0031  B4 09                             mov ah,09h
0033  BA 0021r                          lea dx,nline
```

```
66  0036  CD 21                        int 21h
67
68  0038  B4 09                       mov ah,09h
69  003A  BA 002Er                    lea dx,msg1
70  003D  CD 21                       int 21h
71
72
73  003F  B4 09                       mov ah,09h
74  0041  BA 0024r                    lea dx,rev
75  0044  CD 21                       int 21h
76
77  0046  B4 4C              exit:    mov ah,4ch
78  0048  CD 21                       int 21h
79
80  004A                              code ends
81
82                                    end start
```

RESULT: enter the string hello
the reversed string is olleh

VIVA QUESTIONS:

1. Why to create a newline?
2. Specify string instructions.
3. What is the syntax of move string instruction?
4. What is the use of extra segment in 8086 processor?
5. What is use of direction flag?

C. CHECK WHETHER THE GIVEN STRING IS PALINDROME

AIM : Program to check whether the given string is palindrome or not

EQUIPMENT REQUIRED:

1. TASM Software
2. PC with DOS and Debug program

ALGORITHM:

1. Define the prompt messages to be displayed in data segment
2. Initialize the data segment register with data segment address
3. Use 0ah function to read the string from the keyboard
4. Initialize the source pointer to the end of the read string
5. Initialize the destination pointer to the location where the reversed string is to be stored

6. Initialize the counter to the actual length of the entered string
7. Copy the contents from the source to the destination till the counter is zero
8. Display the reversed string
9. Initialize the counter to the actual length of the entered string again
10. Initialize the source pointer to the starting address of the read string
11. Initialize the destination pointer to the starting address of the reversed string
12. Compare the contents character- wise if found equal display the string is palindrome, else if any character is not equal then display string is not palindrome
13. Terminate the program

PROGRAM:

```
 1                                      assume cs:code,ds:data
 2
 3  0000                                data segment
 4
 5  0000  65 6E 74 65 72 20 74+         msg db "enter the string","$"
 6        68 65 20 73 74 72 69+
 7        6E 67 24
 8
 9  0011                                str1 label byte
10  0011  14                            strmax db 20
11  0012  ??                            stract db ?
12  0013  0A*(24)                       strfld db 10 dup('$')
13
14  001D  0A 0D 24                      nline db 10,13,'$'
15
16  0020  0A*(24)                       rev db 10 dup('$')
17
18  002A  74 68 65 20 65 6E 74+         msg1 db "the entered string
                                                is palindrome","$"
19        65 72 65 64 20 73 74+
20        72 69 6E 67 20 69 73+
21        20 70 61 6C 69 6E       64+
22        72 6F 6D 65 24
23  004B  74 68 65 20 65 6E       74+   msg2 db "the entered string
                                                is not palindrome","$"
24        65 72 65 64 20 73 74+
25        72 69 6E 67 20 69 73+
26        20 6E 6F 74 20 70 61+
27        6C 69 6E 64 72 6F       6D+
28        65 24
29
30
31  0070                                data ends
```

```
32
33
34
35  0000                        code segment
36
37  0000  B8 0000s              start:  mov ax,data
38  0003  8E D8                         mov ds,ax
39
40  0005  B4 09                         mov ah,09h
41  0007  BA 0000r                      lea dx,msg
42  000A  CD 21                         int 21h
43
44  000C  B4 09                         mov ah,09h
45  000E  BA 001Dr                      lea dx,nline
46  0011  CD 21                         int 21h
47
48  0013  B4 0A                         mov ah,0ah
49  0015  BA 0011r                      lea dx,str1
50  0018  CD 21                         int 21h
51
52  001A  B4 09                         mov ah,09h
53  001C  BA 001Dr                      lea dx,nline
54  001F  CD 21                         int 21h
55
56  0021  B4 09                         mov ah,09h
57  0023  BA 0013r                      lea dx,strfld
58  0026  CD 21                         int 21h
59
60  0028  BE 0013r                      lea si,strfld
61  002B  BF 0020r                      lea di,rev
62
63  002E  33 C9                         xor cx,cx
64  0030  8A 0E 0012r                   mov cl,stract
65
66  0034  03 F1                         add si,cx
67  0036  4E                            dec si
68
69  0037  8A 04                 top:    mov al,[si]
70  0039  88 05                         mov [di],al
71  003B  47                            inc di
72  003C  4E                            dec si
73  003D  E2 F8                         loop top
74
75
76  003F  B4 09                         mov ah,09h
77  0041  BA 001Dr                      lea dx,nline
```

```
 78  0044  CD 21                    int 21h
 79
 80
 81  0046  B4 09                    mov ah,09h
 82  0048  BA 0020r                 lea dx,rev
 83  004B  CD 21                    int 21h
 84
 85  004D  BE 0013r                 lea si,strfld
 86  0050  BF 0020r                 lea di,rev
 87
 88  0053  33 C9                    xor cx,cx
 89  0055  8A 0E 0012r              mov cl,stract
 90
 91  0059  8A 04              top1: mov al,[si]
 92  005B  3A 05                    cmp al,[di]
 93  005D  75 15                    jne down
 94  005F  46                       inc si
 95  0060  47                       inc di
 96  0061  E2 F6                    loop    top1
 97
 98  0063  B4 09                    mov ah,09h
 99  0065  BA 001Dr                 lea dx,nline
100  0068  CD 21                    int 21h
101
102
103  006A  B4 09                    mov ah,09h
104  006C  BA 002Ar                 lea dx,msg1
105  006F  CD 21                    int 21h
106
107  0071  EB 0F 90                 jmp exit
108
109  0074  B4 09              down: mov ah,09h
110  0076  BA 001Dr                 lea dx,nline
111  0079  CD 21                    int 21h
112
113  007B  B4 09                    mov ah,09h
114  007D  BA 004Br                 lea dx,msg2
115  0080  CD 21                    int 21h
116
117  0082  B4 4C              exit: mov ah,4ch
118  0084  CD 21                    int 21h
119
120  0086                     code ends
121
122                           end start
```

RESULT: enter the string hello
the reversed string is olleh
the entered string is not palindrome
(or)
enter the string liril
the reversed string is liril
the entered string is palindrome

VIVA QUESTIONS:

1. What is a palindrome?
2. Specify string instructions.
3. What is the syntax of compare string instruction?
4. What is the use of data segment in 8086 processor?
5. What is use of index registers?

Experiment No. 5

5. PROGRAM FOR INTERFACING ADC AND DAC TO 8086

A. A TO D CONVERTER

AIM: Write an ALP to convert the analog signal into its equivalent digital form.

EQUIPMENT REQUIRED:

1. 8086 kit
2. A to D converter interfacing card
3. Flat ribbon cable bus
4. Power supply to 8086 kit
5. Jumper.

HARDWARE CONNECTIONS REQUIRED:

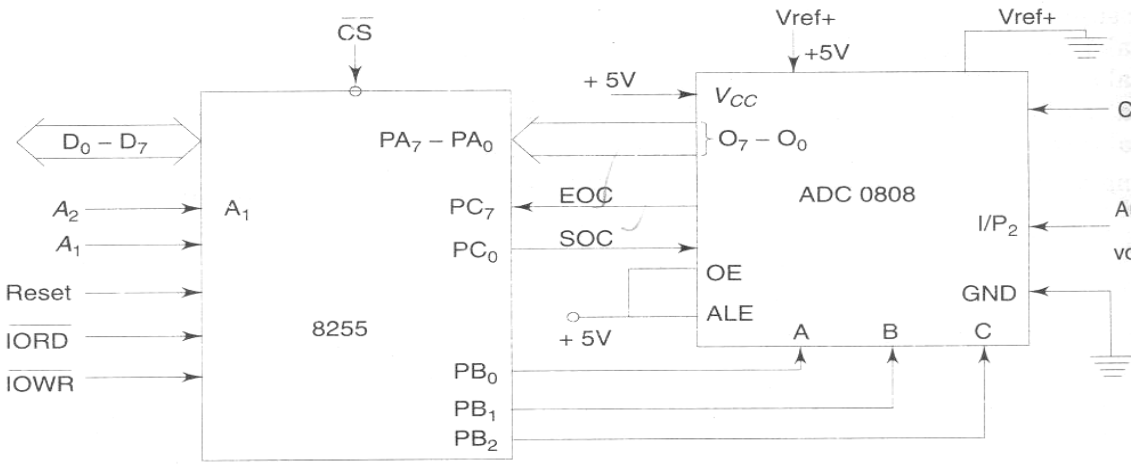
a) connect J_2 to provide 8 channels of ADC which are selected by address supplied by port B (J_3) and latched by Pc_4 bit.

b) This port B is read port of ADC while Pc_1 (lower port C) is input while $Pc_{4,5,6}$ (upper port C) is output commands.

c) To experiment use on board potentiometer as voltage source by shorting 7J1 & 8J1.

INTERFACING CIRCUIT:

ADC 0808 WITH 8086 THROUGH 8255:

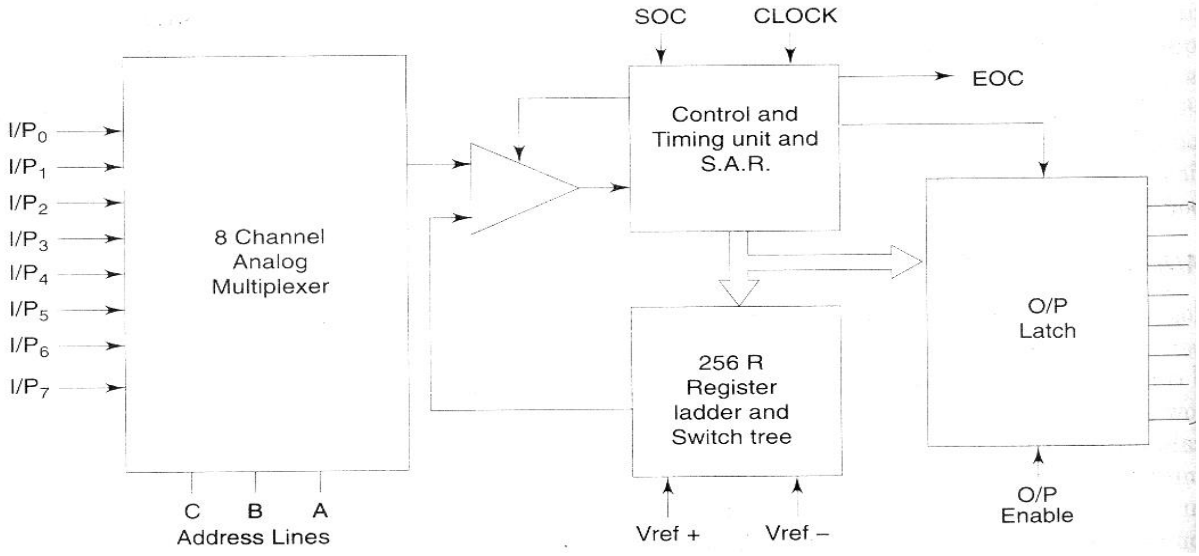


DESCRIPTION ABOUT IC's:

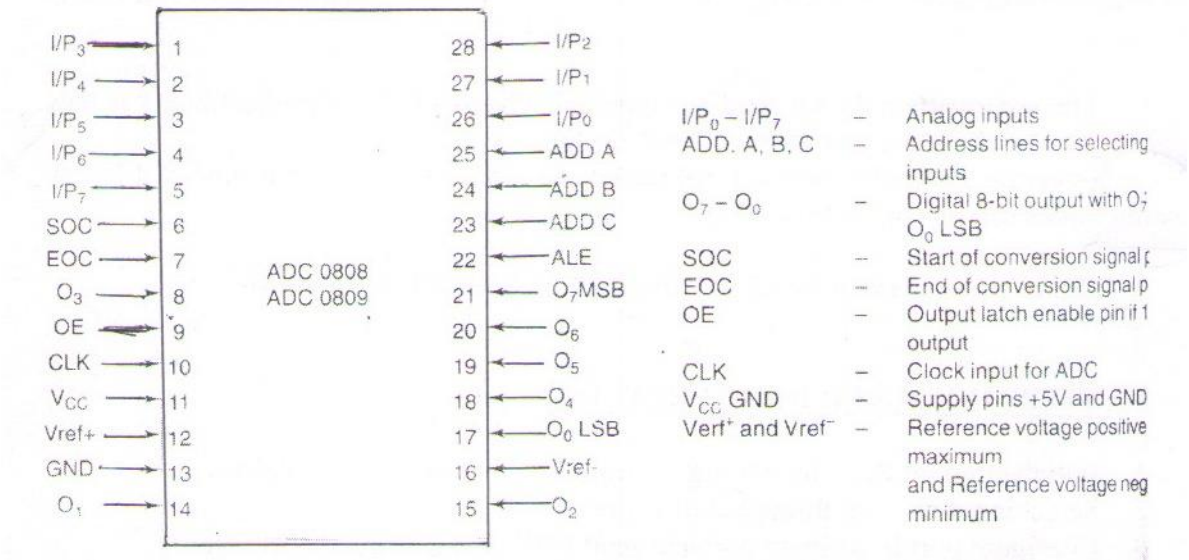
ADC 0808:

* The ADC chips 0808 are 8 bit CMOS, successive approximation converters.

Block diagram of ADC



Pin diagram of ADC 0808/0809



$I/P_0 – I/P_7$ – Analog inputs
(ADD, A,B,C) ADDRESS lines A,B,C – Address lines for selecting analog inputs.
$O_7 – O_0$ – Digital 8 bit output with 0_7 MSB and O_0 LSB
SOC – Start of conversion signal pin
EOC – End of conversion signal pin
OE – Out put latch enable pin if 1 enable output.

CIK – clock input for ADC.
V_{cc}, GND – supply pins +5V and GND.
V_{ref}^+ and V_{ref} – Reference Voltage positive +5V max and reference voltage negative OV minimum.

Electrical specifications of ADC 0808/0809 are given below:

Min. SOC pulse width	100ns
Min. ALE pulse width	100ns
Clock Frequency	10 to 1280 KHz
Conversion time	100ns at 640 KHz
Resolution	8 bit
Error	+/- 1 LSB
V_{ref}+	Not more than +5V
V_{ref}-	Not less than GND
$+V_{cc}$ supply	+5V DC

These converters do not need any external zero (or) full scale adjustments as they are already taken care of by internal circuits. These converters internally have a 3:8 analog multiplexer so that at a time 8 different analog inputs can be connected to chips.

Out of these one is selected by using address lines A,B,C as shown.

STEPS INVOLVED IN INTERFACING:

1. Initialization of 8255 by writing control word into the control register
2. Select input channel through port B lines
3. Configure port B as input port and send SOC signal through port C line
4. Read the status of EOC through port C line
5. If EOC is active, read the digital data through port B.

PROGRAM:

```
MOV AL, 81
MOV DX, 8807   ; Configuring ports as output ports except port C
OUT DX, AL

MOV AL, 00
MOV DX, 8803  ; Sending channel addr on port B
OUT DX, AL

MOV AL, 08
MOV DX, 8807 ; Generate ALE signal on PC3
OUT DX, AL

MOV AL, 09
MOV DX, 8803
OUT DX, AL ; Configure port B as input port

MOV AL, 0C ; Generate start of conversion pulse on PC6
OUT DX, AL
MOV AL, 0D
OUT DX, AL
MOV AL, 0C
OUT DX, AL

MOV DX, 8805
Above: IN AL,DX ; Read End of conversion on PC1
AND AL,02
JZ Above

MOV AL, 0B
MOV DX, 8807; Set O/P enable signal high
OUT DX, AL

MOV AL, 8803; Read the status from AL register
IN AL, DX

INT 3  ; TERMINATE
```

RESULT: When potentiometer was in minimum position the digital output is 00 and when maximum output at AL is FF.

VIVA QUESTIONS:

1. Explain the difference between microcomputer, microprocessor and microcontroller.
2. What are the various types of ADCs
3. Which is the fastest type of ADC
4. Specify the specifications of ADCs
5. What is conversion time
6. Which ADC is having high resolution
7. Name some applications of ADCs
8. What is meant by settling time
9. MC 1208 ADC is how many bit converter
10. Explain the purpose of DMA controller when flash type ADC is used.

B. D TO A CONVERTER

AIM: Write an ALP to convert Digital data into its equivalent Analog Form.

a. Generate a Triangular waveform
b. Generate a Square waveform

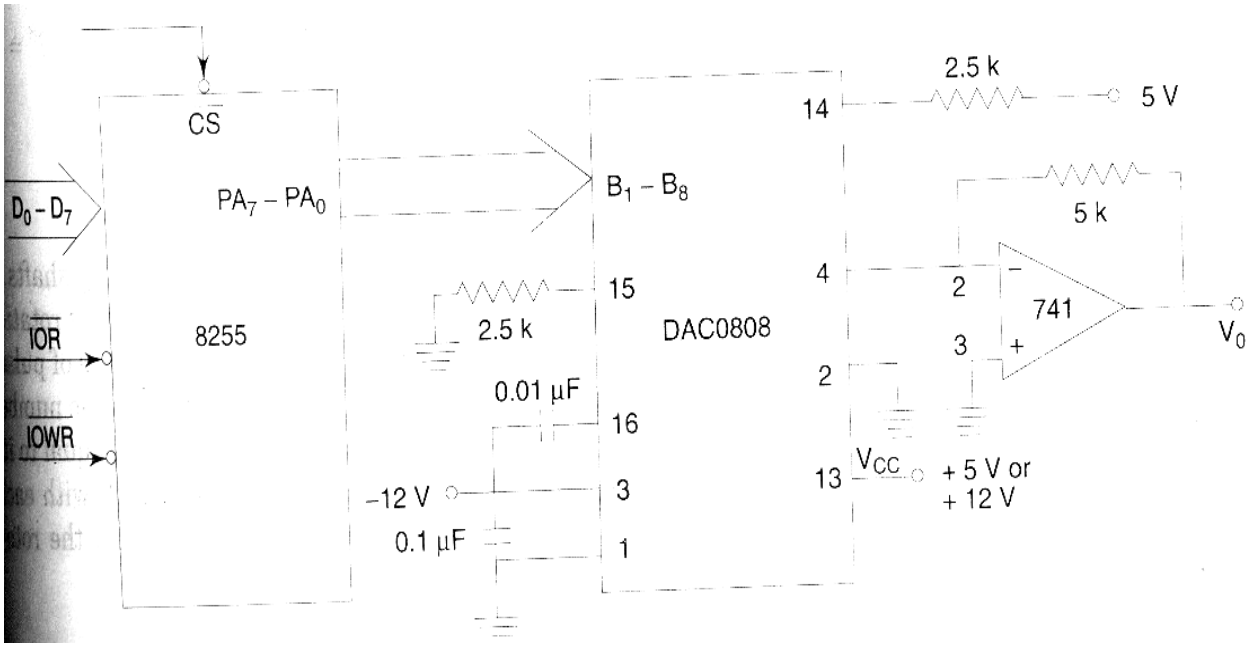
EQUIPMENT REQUIRED:

1. 8086 kit
2. D to A converter interfacing card
3. Flat ribbon cable-buses
4. Power supply to 8086 kit
5. CRO.

HARDWARE CONNECTIONS REQUIRED:

1. To the port A lines of 8255 the DAC 0808 is connected
2. The DAC output is connected to the CRO

INTERFACING CIRCUIT:



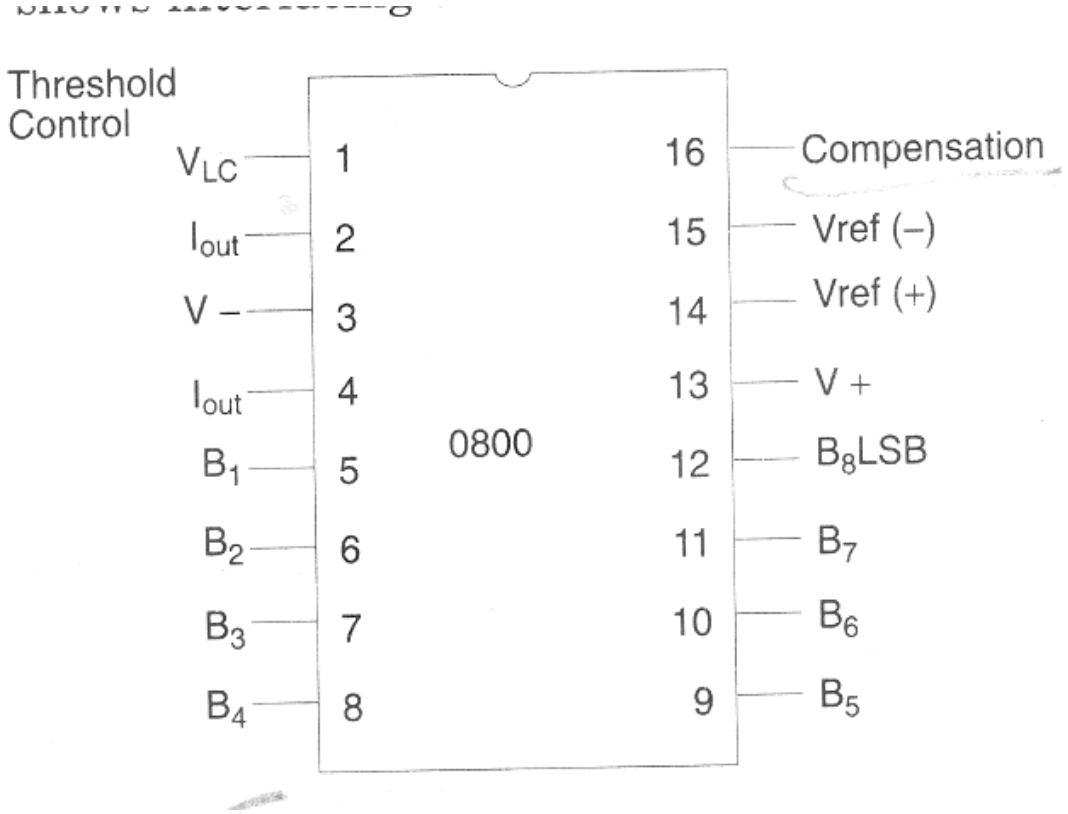
DESCRIPTION ABOUT IC'S:

DAC 0800:

The DAC 0800 is a monolithic 8 bit DAC manufactured by National semi-conductor. It has setting time around 100ms and can operate on a range of power supply voltages, i.e. from 4.5V to +18V.

Usually supply V+ is 5V (or) +12V. The V-pin can be kept at a minimum of -12V.

FIGURE:



B_1-B_8 – Digital inputs
V_{ref}(-), V_{ref}(+) – Reference Voltages
I_{out} – Analog Output signal

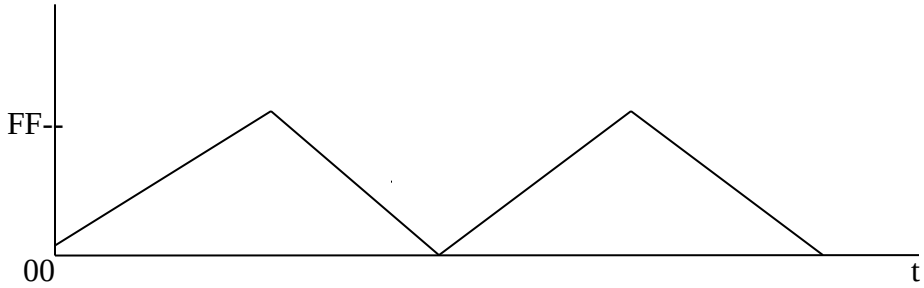
STEPS INVOLVED IN INTERFACING:

1. Initialization of 8255 by writing the control word into the control register
2. Load the accumulator with required data
3. Send the data out in the required way to generate the waveform through port A of 8255
4. Observe the desired output waveform in the CRO.

A.PROGRAM TO GENERATE TRIANGULAR WAVEFORM

```
        MOV AL, 80
        MOV DX, 8807  ; Initialize the 8255 with control word
        OUT DX, AL

        MOV DX, 8801 ; Initialize the DX Register with port A address
Above:MOV AL, 00
Loop1:OUT DX,AL ; Initialize the accumulator with 00

        INC AL ; Increment the accumulator content
        JNZ Loop1 ; Jump for no zero to Label specified

        MOV AL,FF ; Initialize the accumulator with FF
Loop2: OUT DX,AL ; Write the content of accumulator to port A
        DEC AL  ; Decrement the content of accumulator
        JNZ Loop2 ; Jump for no zero to label specified

        JMP Above ; Jump to the label specified
```

EXPECTED GRAPH

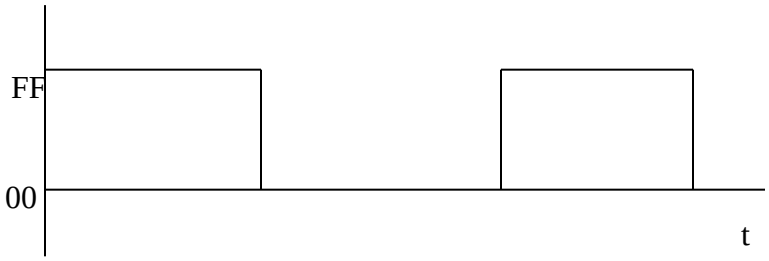


B.PROGRAM TO GENERATE SQUARE WAVEFORM

```
        MOV AL,80
        MOV DX,8807        ; Initialize the 8255 with control word
        OUT DX,AL

        MOV DX,8801        ; Initialize the DX Register with port A
Above:MOV AL,FF            ; Move FF into Accumulator
        MOV BL,10          ; Move the unit value into BL Register

Loop1: OUT DX,AL           ; Write content of accumulator to port A
```

```
        DEC BL      ; Decrement the content of BL Register
        JNZ Loop1   ; Jump for BL not zero to specified label

        MOV AL,00   ; Move 00 into the accumulator
        MOV BL,10   ; Move 10 into the BL Register

Loop2: OUT DX,AL    ; Write content of accumulator into port A

        DEC BL      ; Decrement the content of BL Register
        JNZ Loop2   ; Jump for BL not zero to specified label

        JMP Above   ; Jump to the specified location
```

EXPECTED GRAPH



RESULT:

Observed the generated triangular waveform & square waveform in CRO.

VIVA QUESTIONS:

1. Explain the difference between the near call and the far call
2. Explain the generation technique to convert Digital signal into Analog form.
3. Compare R-2R ladder method to Weighted resister method
1. Explain about the specifications of DAC
2. How the DAC are interfaced with processor
3. What is meant by resolution of DAC
4. What is meant by settling time
5. Define linearity, accuracy for DAC
6. Give some applications of DAC
7. What is the importance of Op-amp in DAC

Experiment No. 6

6. INTERFACING TO 8086 AND PROGRAMMING TO CONTROL STEPPER MOTOR

AIM: Program to interface the stepper motor to 8086 microprocessor

EQUIPMENT REQUIRED:

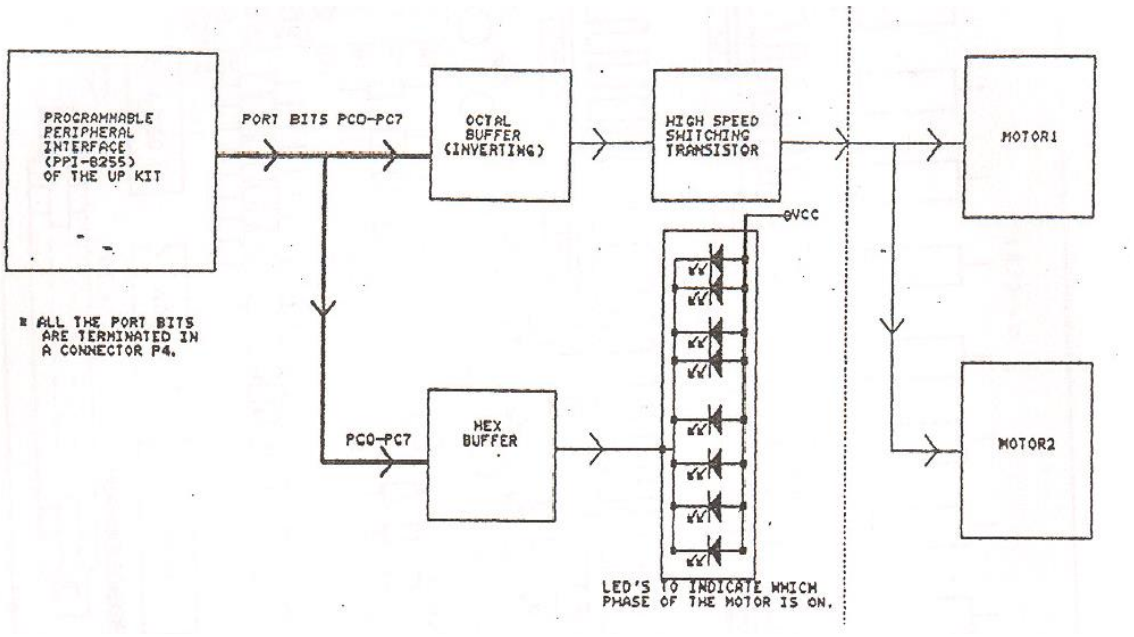
1. 8086 microprocessor trainer kit
2. Stepper motor interfacing module
3. Flat ribbon cable bus
4. Keyboard
5. Power chord

HARDWARE CONNECTIONS REQUIRED:

1. Connect P3 on 86M to the connector C1 on the interfacing using a 26 core flat cable.
2. Motor is a Z phase, 6 wire motor.
3. Power connections:
   A 4-Way power mate female is provided with wires.

White/blue/orange	-	+5V
Black & Red	-	COM (GND)
Green	-	+12V or 6V

INTERFACING CIRCUIT:



DESCRIPTION ABOUT STEPPER MOTOR INTERFACE:

Stepper motors are very useful electro –mechanical transducers for position control. They are used in a number of industrial control applications.

The dual stepper motor interface designed to be used with ALS-SDA-86 can simultaneously control two stepper motors. It can be used to control single phase, two phase , three phase and four phase indigenous and imported stepper motors.

The interface is designed with high speed switching Darlington transistors with MAX 1A, 80V rating with appropriate heat sinks. These switches are driven through open collector TTL buffers which interface the lines to the motor circuits. The logic power and motor power are supplied directly to the interface .This allows the use of motors with different input voltage ratings with same drive circuit .The interface is also provided with current limiting circuits which protects the power devices against overload or accidental voltages. The LEDs on the interface indicate the phases which are energized for easy demonstration. There are suppression circuits which clamp the transient voltages to safe limits during switching of phases. By suitable switch sequences the motor can be used in three modes of operation:

a. One phase ON (medium torque)
b. Two phase ON (high torque)
c. Half stepping (low torque)

The interface can be connected to the 8255 ports in 8086 trainer kit using 26 core flat cable. In order to generate logic sequences conveniently using the Bit-Set/Reset facility of port C in 8255, the interface uses port C signals to drive the switches.

Switching logic:

The stepping action is caused by sequential switching of supply to the two phases of the motor.

4 Step input sequence			
Phase 1		Phase 2	
G	B	O	R
1	0	1	1 = 3 Hex
1	0	0	1 = 9
1	1	0	0 = C
0	1	1	0 = 6

Four step input sequence gives 1.8 degree (full) step function.

8 Step input sequence			
Phase 1		Phase 2	
G	B	O	R
0	0	1	1 = 3 Hex
0	0	0	1 = 1
1	0	0	1 = 9
1	0	0	0 = 8
1	1	0	0 = C

0	1	0	0 = 4
0	1	1	0 = 6
0	0	1	0 = 2

Eight step input sequence gives 0.9 degree (half) step function.
To change the directions follow the sequence from bottom to top. The step rate (speed of rotation) is governed by the frequency of switching.

ONE PHASE ON SCHEME:
At a time only one of the phases is switched ON as given below.

STEPS INVOLVED IN INTERFACING: I

1. Initialization of 8255 by writing the control word into the control register
2. Load the accumulator with byte to switch on phase A (first step sequence)
3. Send the data out through port C of 8255 for producing first step in stepper motor.
4. Introduce a delay between each steps
5. Load the accumulator with byte to switch on phase B (second step sequence)
6. Send the data out through port C of 8255 for producing second step in stepper motor.
7. Introduce a delay between each steps
8. Load the accumulator with byte to switch on phase C (third step sequence)
9. Send the data out through port C of 8255 for producing third step in stepper motor.
10. Introduce a delay
11. Load the accumulator with byte to switch on phase D (fourth step sequence)
12. Send the data out through port C of 8255 for producing fourth step in stepper motor.
13. Introduce a delay
14. Continue from step 2 to rotate the stepper motor in clock-wise direction.

PROGRAM 1 :

```
2000  B0 80          MOV AL,80        ; Initialize 8255
2002  BA C6 FF       MOV DX,FFC6
2005  EE             OUT DX,AL

2006  B0 EE    START:MOV AL,EE        ; Byte to switch on A phase
2008  BA C4 FF       MOV DX,FFC4
200B  EE             OUT DX,AL

200C  E8 F1 00       CALL DELAY       ; Wait

200F  B0 DD          MOV AL,DD        ; Byte to switch on B phase
2011  BA C4 FF       MOV DX,FFC4
```

```
2014   EE              OUT DX,AL

2015   E8 EB 00        CALL DELAY      ; Wait

2018   B0 BB           MOV AL,BB       ; Byte to switch on C phase
201A   BA C4 FF        MOV DX, FFC4
201D   EE              OUT DX,AL

201E   E8 DF 00        CALL DELAY      ; Wait

2021   B0 77           MOV AL,77       ; Byte to switch on D phase
2023   BA C4 FF        MOV DX,FFC4
2026   EE              OUT DX, AL

2027   E8 D6 00        CALL DELAY      ; Wait

202A   EB DA           JMP START       ; Go to start
202C   CC              INT 3

2100   B9 FF FF        DELAY:MOV CX,0FFFF  ; Initialize counter(Hex value)
2103   90              L1:   NOP           ; No operation

2104   90              NOP
2105   90              NOP
2106   49              DEC CX
2107   75 FA           JNZ L1
2109   C3              RET
```

STEPS INVOLVED IN INTERFACING: II

1. Initialization of 8255 by writing the control word into the control register
2. Load the accumulator with byte to switch on phase A (first step sequence)
3. Send the data out through port C of 8255 for producing first step in stepper motor.
4. Introduce a delay
5. Rotate the contents of accumulator to left by 1 bit in order to switch to the next phase
6. Introduce a delay
7. Continue from step 5 to rotate the stepper motor in clock-wise direction.

PROGRAM 2 :

```
2000   B0 80           MOV AL,80       ; Initialize 8255
2002   BA C6 FF        MOV DX,FFC6
2005   EE              OUT DX,AL

2006   B0 EE     START:MOV AL,EE       ; Byte to switch on A phase
```

```
2008   BA C4 FF        MOV DX,FFC4
200B   EE              OUT DX,AL

200C   E8 F1 00        CALL DELAY        ; Wait

200F   D0 C0           ROL AL,1          ; Rotate phases
2011   EB F8           JMP L1
2013   CC              INT 3

2100   B9 FF FF        DELAY:MOV CX,0FFFF  ; Initialize counter(Hex value)
2103   90              L1:    NOP          ; No operation

2104   90              NOP
2105   90              NOP
2106   49              DEC CX
2107   75 FA           JNZ L1
2109   C3              RET
```

RESULTS:

Observed the stepper motor rotation

VIVA QUESTIONS:

1.What is the principle of working of stepper motor.
2.What are the applications of stepper motor
3.In what way the stepper motor is different from other motors.
4.Specify the changes in the logic to obtain the step of 180 degrees
5.Name the technique used to obtain the steps of smaller size
6.What are optical shaft encoders
7. What are the components used in the interfacing of stepper to processor
8. What are A/D converters?
9. What are D/A converters?
10. Give the specifications that are of concern in selecting steppers for given application

Experiment No.7

7. PROGRAM USING ARITHMETIC, LOGICAL, AND BIT MANIPULATION INSTRUCTIONS OF 8051.

AIM: Program using arithmetic, logical and bit manipulation instructions of 8051.

EQUIPMENT REQUIRED:

1. 8051 Trainer kit
2. RS232 cable
3. Max 232 IC,
4. Keil µ Vision 3 software

PROCEDURE:

1. Write the assembly language program for this task in the keil compiler
2. Load the two values into two registers
3. Perform the required operation using corresponding operator
4. End the program

PROGRAM:

A. ARITHMETIC OPERATIONS

ADDITION:

```
C:0X0000    7405      MOV A,#0X05
C:0X0002    74F005    MOV B(0XF0),#0X05
C:0X0005    25F0      ADD A,B(0XF0)
                      END
```

SUBSTRACTION:

```
C:0X0000    7405      MOV A,#0X05
C:0X0002    75F002    MOV B,B(0XF0),#0X02
C:0X0005    95F0      ADD A,B(0XF0)
                      END
```

MULTIPLICATION:

```
C:0X0000    7402      MOV A,#0X05
C:0X0002    74F003    MOV B(0XF0),#0X03
C:0X0005    AA        MUL AB
                       END
```

DIVISION:

```
C:0X0000    7409      MOV A,#0X09
C:0X0002    75F003    MOV B(0XF0),#0X03
C:0X0005    84        DIV AB
                      END
```

RESULT:
ADDITION: A=05
SUBTRACTION:A=03
MULTIPLICATION:A=0F
DIVISION:A=03

B.LOGICAL OPERATIONS
OR LOGIC:

```
C:0X0000    7403      MOV A,#0X03
C:0X0002    74F002    MOV B(0XF0),#0X02
C:0X0005      45F0    ORL A,B(0XF0)
                      END
```

AND LOGIC:

```
C:0X0000    7404      MOV A,#0X04
C:0X0002    75F002    MOV B(0XF0),#0X02
C:0X0005      55F0    ANL A,B(0XF0)
                      END
```

XOR LOGIC:

```
C:0X0000    7404      MOV A,#0X04
C:0X0002    75F002    MOV B(0XF0),#0X02
C:0X0005      65F0    XRL A,B(0XF0)
                      END
```

RESULT:
OR LOGIC: A=03
AND LOGIC: A=00
XOR LOGIC: A=06

BIT MANIPULATION OPERATORS
ROTATE RIGHT:

```
C:0X0000    740F      MOV A,#0X0F
C:0X0002      03      RR A
                      END
```

ROTATE RIGHT WITH CARRY:

```
C:0X0000    7405      MOV A,#0X05
C:0X0002      13      RRC A
                      END
```

ROTATE LEFT:

```
C:0X0000    7405      MOV A,#0X05
C:0X0002      23      RL A
                      END
```

ROTATE LEFT WITH CARRY:

```
C:0X0000    740C      MOV A,#0X0C
C:0X0002      33      RLC A
                      END
```

SWAP:

```
C:0X0000    7408      MOV A,#0X08
```

C:0X0002	14	SWAP A
		END

RESULT:
ROTATE RIGHT:A=07
ROTATE RIGHT WITH CARRY:A=02
ROTATE LEFT:A=0A
ROTATE LEFT WITH CARRY:A=18
SWAP:A=80

VIVA QUESTIONS:
1.What are the types of instructions in 8051
2. Discuss the various bit manipulation instructions
3. Discuss the various arithmetic and logical instructions

Experiment No.8

8. PROGRAM AND VERIFY TIMER/COUNTER IN 8051

A. PROGRAMMING TIMER

AIM: To create the square wave of 50% duty cycle on the p1.5 bit. Timer 0 is used to generate
the time delay.

EQUIPMENT REQUIRED:

1. 8051 Trainer kit
2. RS232 cable
3. Max 232 IC
4. Keil µ Vision 3 software

PROCEDURE:

1. Write the assembly language program for this task in the keil compiler.
2. Load the delay values into timer 0
3. Complement the port line and start timer
4. If timer is complete, complement the port line and again start the timer by reloading values
5. This is repeated continuously

PROGRAM:

```
                MOV     TMOD,#01          ;Timer   0, mode   1(16-bit mode)
HERE:           MOV     TL0,#0F2H         ;TL0   =F2H,    the Low byte
                MOV     TH0,#0FFH         ;TH0   =FFH,    the High byte
                CPL     P1.5              ; Toggle    P1.5
                A CALL DELAY
                SJMP HERE                  ; load TH, TL again

;…….. delay using Timer 0
DELAY:
                SET B   TR0               ; Start Timer 0
AGAIN:          JNB     TF0, AGAIN        ; Monitor Timer 0 Flag Until
                                          ; It rolls over
                CLR     TR0               ; Stop Timer 0
                CLR     TF0               ; Clear Timer 0 Flag
                RET
```

RESULT: As the timer 0 rolls over ,the port line p1.5 status toggles continuously.

B. PROGRAMMING COUNTER

AIM: Assuming that clock pulses are fed into pin T1,Write a program for counter 1 in mode 2 to
count the pulses and display the state of the TL1 count on P2.

EQUIPMENT REQUIRED:

1. 8051 Trainer kit
2. RS232 cable
3. Max 232 IC
4. Keil µ Vision 3 software.

PROCEDURE:

1. Write the assembly language program for this task in the keil compiler.
2. Choose the counter mode for operation
3. Run the counter and it starts counting the pulses from T1 pin
4. If counter is stopped, the count value is displayed
5. This is repeated continuously

PROGRAM:

```
        MOV   TMOD, #01100000B    ; counter 1, mode 2, C/T=1
                                   ; External pulses
        MOV   TH1 , #0            ;clear TH1
        SETB   P3.5               ; make T1 input
AGAIN : SETB   TR1                 ; start the counter
BACK :  MOV    A,TL1               ; get copy of count TL1
        MOV    P2,A                ; display it on port2
        JNB    TF1, BACK           ; keep doing it if TF=0
        CLR    TR1                 ; stop the counter 1
        CLR    TF1                 ; make TF=0
        SJMP   AGAIN               ; keep doing it
```

RESULT:

The counter counts and displays the count value on port 2.

VIVA QUESTIONS:

1. What are the addresses of the two timers
2. What is the purpose of timers
3. What are the modes in which they work
4. What are the SFRs used to define modes for counters
5. What is the purpose of counters
6. What are the modes in which they work
7. What is the significance of gate bit in TMOD register.

Experiment No.9

9. PROGRAM AND VERIFY INTERRUPT HANDLING IN 8051

A. USING TIMER INTERRUPTS

AIM: Write a program to generate a square wave of 50 Hz frequency on pin P1.2. Use an interrupt for timer 0. Assume the XTAL =11.0592 MHz.

EQUIPMENT REQUIRED:

1. 8051 Trainer kit
2. RS232 cable
3. Max 232 IC
4. Keil µ Vision 3 software

PROCEDURE:

1. Write the assembly language program for this task in the keil compiler.
2. Load the delay values into timer 0
3. Complement the port line and start timer
4. If timer is complete, the corresponding timer interrupt is generated, complement the port line and again start the timer by reloading values
5. This is repeated continuously

PROGRAM:

```
        ORG 0
        LJMP MAIN
        ORG 000BH          ; ISR for timer0
        CPL  P1.2          ; complement p1.2
        MOV TL0,#00        ;reload timer values
        MOV TH0,#0DCH
        RETI               ; return from interrupt
        ORG 30H             ; main program for initialization
MAIN:   MOV TMOD,#01H      ;timer 0, mode 1
        MOV  TL0,#00H
        MOV TH0,#0D2H
        MOV IE,#82H        ;enable timer 0 interrrupt
        SETB TR0           ; start timer
HERE:   SJMP HERE          ; stay here until interrupted
        END
```

RESULT:
The functionality of timer interrupts is verified.

B. PROGRAMMING EXTERNAL INTERRUPTS

AIM: Assume that the INT1 pin is connected to a switch that is normally high. Whenever it goes low , it should turn on a LED. The LED is connected to p1.3 and is normally off. When it is turned on it should stay on for a fraction of second. As long as the switch is pressed low,the LED should stay on.

EQUIPMENT REQUIRED:

1. 8051 Trainer kit
2. RS232 cable
3. Max 232 IC
4. Keil µ Vision 3 software

PROCEDURE:

1. Write the assembly language program for this task in the keil compiler.
2. Load the delay values into timer 0
3. Complement the port line and start timer
4. If an interrupt is generated, complement the port line and again start the timer by reloading values
5. This is repeated continuously

PROGRAM:

```
                ORG    000H
                LJMP   MAIN            ;bypass interrupt vector table
;--ISR for hardware interrupt INT1 to turn on the LED
                ORG    0013H           ;INT1  ISR
                SETB   P1.3            ; turn on LED
                MOV    R3, #255        ; load counter
BACK:           DJNZ   R3, BACK        ; keep LED on for a while
                CLR    P1.3            ; turn off the LED
                RET1                   ; return from ISR
;-- MAIN Program for initialization
                ORG 30H
MAIN:           MOV    IE,#1000100B    ; enable external INT1
HERE:           SJMP   HERE            ; stay here until interrupted
                END
```

RESULT:
The functionality of external interrups is verified.

C. PROGRAMMING SERIAL INTERRUPTS

AIM: Write a program in which the 8051 reads data from P1 and writes it toP2 continuously
while giving a copy of it to the serial COM port to be transferred serially. Assume that
XTAL=11.0592 MHz. Set the baud rate at 9600.

EQUIPMENT REQUIRED:

1. 8051 Trainer kit
2. RS232 cable
3. Max 232 IC
4. Keil µ Vision 3 software

PROCEDURE:

1. Write the assembly language program for this task in the keil compiler.
2. Load the delay values into timer 0
3. Complement the port line and start timer
4. If timer is complete, complement the port line and again start the timer by reloading values
5. This is repeated continuously

PROGRAM:

```
        ORG   0
        LJMP  MAIN
        ORG   23H
        LJMP   SERIAL              ; Jump to serial interrupt ISR
        ORG    30H
MAIN:   MOV    P1, #0FFH           ; Make P1 an input port
        MOV    TMOD, #20H          ; timer 1, mode 2(auto-reload)
        MOV    TH1, #0FDH          ; 9600 baud rate
        MOV     SCON, #50H         ; 8-bit, 1 stop, REN enabled
        MOV      IE, #10010000B    ; enable serial interrupt
        SETB      TR1              ; Start timer 1
BACK:   MOV       A,P1             ; read data from port 1
        MOV      SBUF, A           ; give a copy to SBUF
        MOV       P2, A            ; send it to p2
        SJMP      BACK             ; Stay in loop indefinitely
```

```
;
;---------------------------- Serial port   ISR

                ORG       100H
SERIAL:     JB          T,TRANS                ; jump if T1 is high
                MOV       A, SBUF                ; otherwise due to receive
                CLR        RI                     ; clear RI  since CPU does not
                RETI                              ; return from ISR
```

RESULT:
The functionality of serial i.e., transmit interrupt and receive interrupt is verified.

VIVA QUESTIONS:

1. What is the interrupt structure of 8051
2. What are the SFRs used with interrupts of 8051
3. What are the vector addresses of these interrupts.
4. What is the priority of interrupts

Experiment No.10

10. PROGRAM FOR UART OPERATION IN 8051

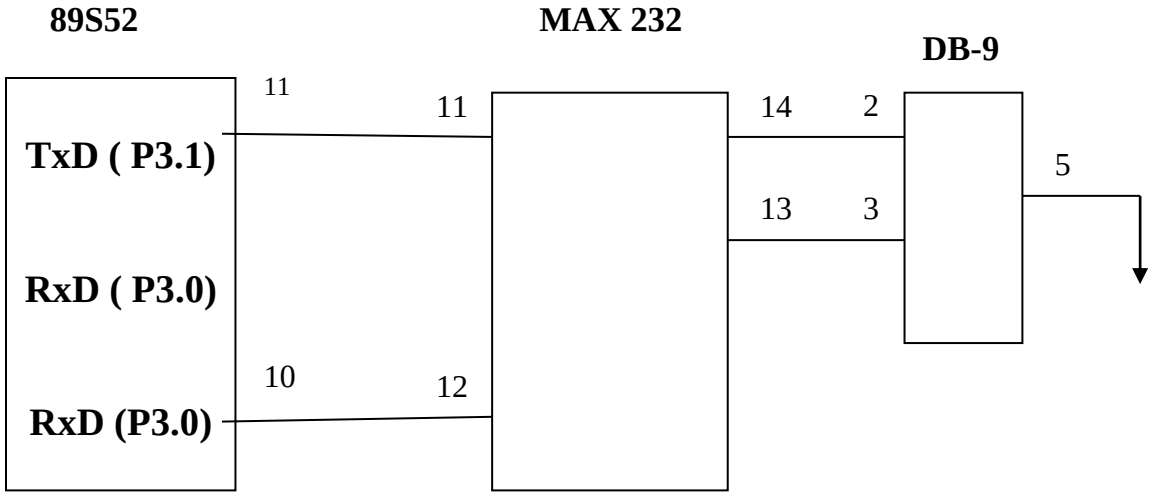
A.SERIAL TRANSMISSION

AIM: To send the data serially out from microcontroller to pc.

EQUIPMENT REQUIRED:

1. 8051 Trainer kit
2. RS232 cable
3. Max 232 IC,
4. Keil µ Vision 3 software

HARDWARE CONNECTIONS REQUIRED:



PROCEDURE:

1. Write the assembly language program for this task in the keil compiler.
2. Compile it and dump the hex file as per the procedure given above.
3. Place this Microcontroller in the trainer kit and fix it properly.
4. Power up the trainer kit from the mains supply.
5. In order to establish communication between PC and Microcontroller, Serial cable RS232 and Voltage converter MAX232 IC are used. This section is available on the system board.
6. For serial communication, port 3(pins 10 and 11 of Microcontroller) are used as reception and transmission pins respectively.
7. The pins 10 and 11 of Microcontroller are connected to the pins 12 and 11 of MAX232 IC respectively.
8. Connect one end of the serial cable to the processor and the other end to the DB9 pin provided on the system board.
9. Check the connections before switching on the mains.

10. An LED is provided in the power supply section of the trainer kit to indicate if the board is powered or not.
11. After the board is powered up , open the Hyperterminal window in the PC.
12. According to the program written in the Microcontroller, the characters entered in the Hyperterminal window in the PC can be seen on the LCD display i.e., the Microcontroller receives the data from PC.

PROGRAM:

```
        ORG 00H
        MOV TMOD,#20H   ;TO SELECT TIMER 1 IN MODE 2(AUTO RELOAD)
        MOV SCON,#50H   ; TO SELECT SERIAL COMMUNICATION IN MODE 1
                        ; AND RECEIVER ENABLE
        MOV TH1,#0FDH   ; TO SET THE BAUD RATE TO 9600
        MOV DPTR,#MSG
BACK:   CLR A
        MOVC A,@A+DPTR  ; TO LOAD THE MESSAGE INTO ACCUMULATOR
                        ; CHARACTER BY CHARACTER
        JZ NEXT
        ACALL TRANSMIT  ; CALL SUBROUTINE FOR TRANSMISSION
        INC DPTR
        SJMP BACK
NEXT:   SJMP NEXT       ; TERMINATION OF THE PROGRAM

                /*TRANSFERRING DATA SERIALLY*/

TRANSMIT: SETB TR1      ; TO START THE TIMER
        MOV SBUF,A      ; LOAD DATA INTO SBUF
        JNB TI,$        ; WAIT FOR THE TRANSMISSION TO BE
                          COMPLETED
        CLR TI          ; CLEAR TI FOR NEXT TRANSMISSION
        RET

MSG:    DB 13,10,"WELCOME TO ALL",13,10,0

        END
```

RESULT:

The data bytes is transmitted serially out and is displayed on hyper-terminal of PC.

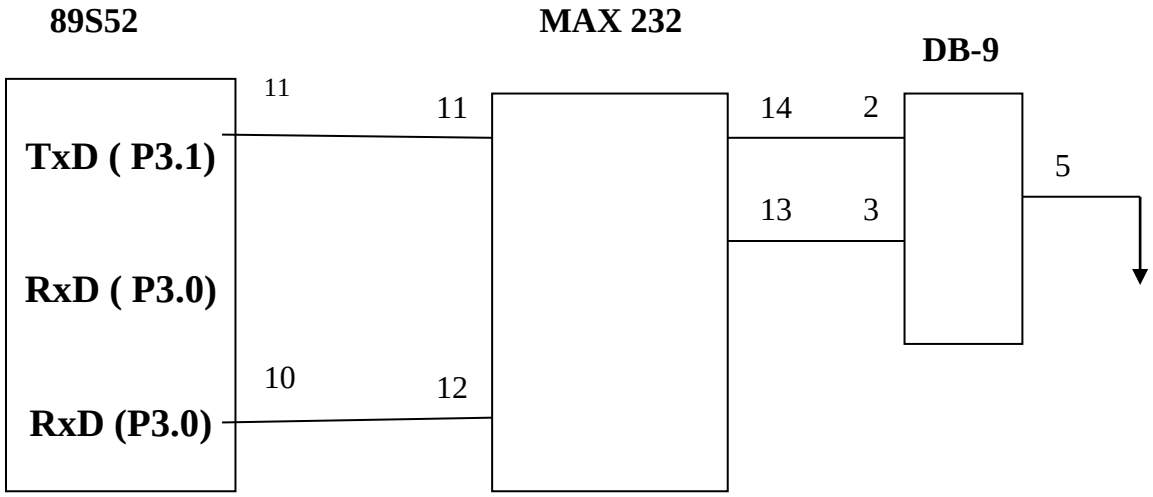
B.SERIAL RECEPTION

AIM: To receive the data serially from pc to microcontroller

EQUIPMENT REQUIRED:

1. 8051 Trainer kit
2. RS232 cable
3. Max 232 IC,
4. Keil µ Vision 3 software

HARDWARE CONNECTIONS REQUIRED:



LCD CONTROL PINS- RS-PORT P2.0
RW-P2.1
EN- P2.2

LCD DATA PINS- PORT P1

SERIAL COMMUNICATION- PORT P3.0, P3.1

PROCEDURE:

1. Write the assembly language program for this task in the keil compiler.
2. Compile it and dump the hex file as per the procedure given above.
3. Place this Microcontroller in the trainer kit and fix it properly.
4. Power up the trainer kit from the mains supply.
5. In order to establish communication between PC and Microcontroller, Serial cable RS232 and Voltage converter MAX232 IC are used. This section is available on the system board.
6. For serial communication, port 3(pins 10 and 11 of Microcontroller) are used as reception and transmission pins respectively.
7. The pins 10 and 11 of Microcontroller are connected to the pins 12 and 11 of MAX232 IC respectively.

8. Connect one end of the serial cable to the processor and the other end to the DB9 pin provided on the system board.
9. Check the connections before switching on the mains.
10. An LED is provided in the power supply section of the trainer kit to indicate if the board is powered or not.
11. According to the program written in the Microcontroller, the message can be seen in the LCD display of the microcontroller kit.

PROGRAM:

```
                RS11 EQU P2.0        ;ASSIGNING NAMES TO THE PORT
PINS AS
                                     ; MENTIONED
                RW11 EQU P2.1
                EN EQU P2.2

                ORG 00H
                MOV DPTR,#CMD   ;INITIALIZATION OF LCD COMMANDS
      BACK1:    CLR A
                MOVC A,@A+DPTR
                JZ NEXT
                ACALL COMMAND
                INC DPTR
                SJMP BACK1

      NEXT:     MOV TMOD,#20H   ;TO SELECT TIMER 1 IN MODE 2
(AUTO
                                ; RELOAD)
                MOV SCON,#50H   ; TO SELECT SERIAL
COMMUNICATION IN
                                 ; MODE 1
                                 ; AND RECEIVER ENABLE
                MOV TH1,#0FDH   ; TO SET THE BAUD RATE TO 9600
      BACK:     SETB TR1        ; TO START THE TIMER

                JNB RI,$        ;WAIT FOR THE CHARACTER TO BE
                                ; LOADED INTO SBUF
                CLR RI          ; CLEAR RI FOR THE NEXT
RECEPTION
                MOV A,SBUF      ; TO MOVE THE RECEIVED
CHARACTER
                                ; INTO A
```

```
                ACALL DATA1     ; CALL SUBROUTINE FOR DISPLAY
ON THE
                                ;LCD
                JMP BACK
      COMMAND:MOV P1,A
                CLR RS11
                CLR RW11
                SETB EN
                ACALL DELAY
                CLR EN
                RET

      DATA1:    MOV P1,A
                SETB RS11
                CLR RW11
                SETB EN
                ACALL DELAY
                CLR EN
                RET

      DELAY:    MOV R6,#255
      HERE:     MOV R7,#255
                DJNZ R7,$
                DJNZ R6,HERE
                RET

      CMD:      DB 38H,0EH,01H,06H,80H,0
                END
```

RESULT:

The message received is displayed on the LCD screen of microcontroller kit.

VIVA QUESTIONS:

1. What are the SFRs used in UART mode of 8051
2. What are the modes in which serial communication can be done
3. How the baudrate are calculated in different serial communication modes of 8051

Experiment No.11

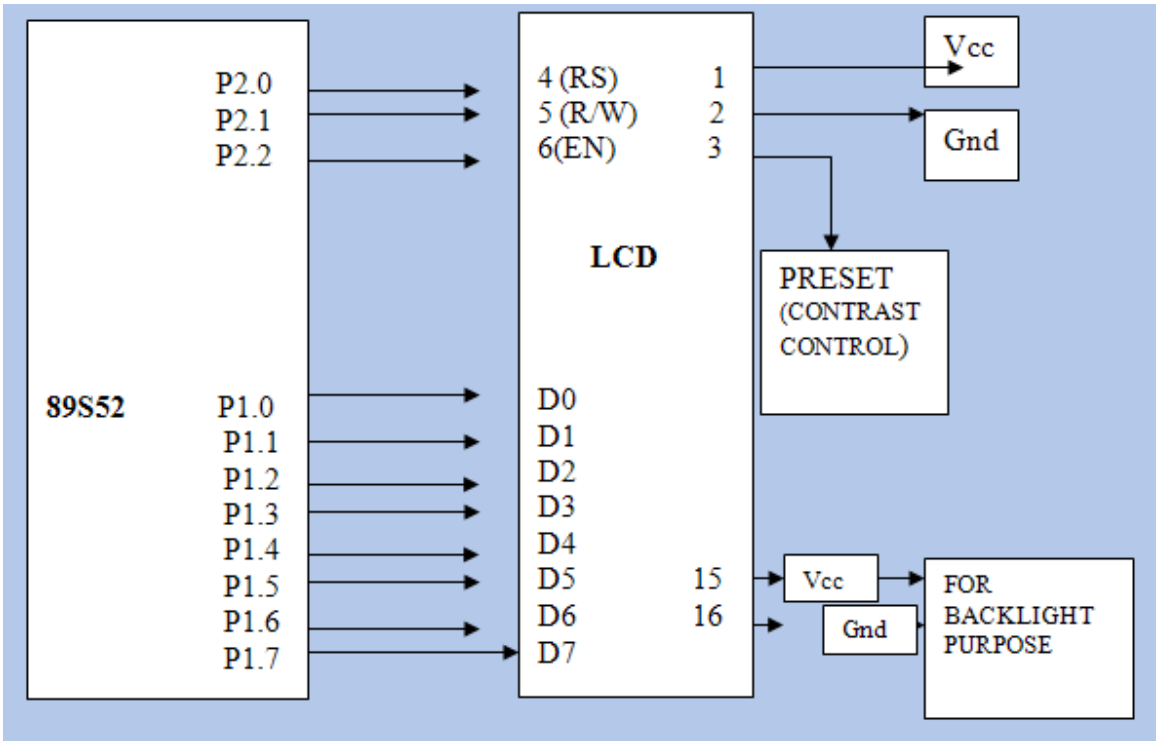
11. PROGRAM FOR INTERFACING LCD TO 8051

AIM: To display message on the LCD.

EQUIPMENT REQUIRED:

1. 8051 Trainer kit
2. RS232 cable
3. Max 232 IC
4. Keil µ Vision 3 software

HARDWARE CONNECTIONS REQUIRED:



LCD CONTROL PINS- RS-P3.5
RW-P3.6
EN-P3.7

LCD DATA PINS- PORT 2(P2)

PROCEDURE:

1. Write the assembly language program for this task in the keil compiler.

2. In order to display any message on the LCD, first the LCD should be initialized. This is done by writing some commands in the programs. LCD contains both data and commands.
3. 3 Pins for commands and 8 pins for data of LCD are connected to the microcontroller port pins in the program.
4. Compile it and dump the hex file as per the procedure given above.
5. Place this microcontroller in the trainer kit and fix it properly.
6. Power up the trainer kit from the mains supply.
7. Now connect the Port 1(pins 1 to 8 of Microcontroller) to the data pins of LCD provided on the system board using 8-pin connector.
8. Similarly connect the Port 2(pins 21, 22, 23 of microcontroller) to the command pins of LCD provided on the system board using 3-pin connector.
9. Check the connections before switching on the mains.
10. An LED is provided in the power supply section of the trainer kit to indicate if the board is powered or not.
11. After the board is powered up, the message will be displayed on the LCD.
12. According to the program written in the microcontroller, the message can be seen on the LCD provided on the system board.

PROGRAM:

```
        RS11 EQU P2.0      ;ASSIGNING NAMES TO THE PORT PINS
        RW1 EQU P2.1
        EN EQU P2.2
        ORG 00H
        MOV DPTR,#COMM
STAY:   CLR A
        MOVC A,@A+DPTR
        JZ MESG
```

```
                ACALL COMMAND       ;       CALL        COMMAND
SUBROUTINE
                INC DPTR
                SJMP STAY
    MESG:       MOV DPTR, #MSG
                ACALL DISPLAY
                ACALL DELAY
                MOV A, #0C2H
                ACALL COMMAND
                MOV DPTR,#MSG1
                ACALL DISPLAY
                ACALL DELAY
   HERE1:       SJMP HERE1          ; STAY HERE
   DISPLAY:     CLR A
                MOVC A,@A+DPTR
                JZ XX
                ACALL DATA1         ; CALL DATA SUBROUTINE
                INC DPTR
                SJMP DISPLAY
    XX:         RET
   COMMAND:                         ; SEND COMMAND TO LCD
                MOV P1, A           ; SEND COMMAND TO PORT
                CLR RS11            ; RS=0 FOR COMMAND
                CLR RW1             ; RW=O FRO WRITE OPERATION
                SETB EN             ; EN=1 FOR HIGH PULSE
                ACALL DELAY         ; GIVE LCD SOME TIME
                CLR EN              ;  EN=0  FOR  HIGH  TO  LOW
PULSE
                RET
  DATA 1 :                  ; SEND DATA TO LCD
                MOV P1, A   ; SEND DATA TO PORT
```

```
                SETB RS11           ; RS=1 FOR DATA
                CLR RW1             ; RW=0 FOR WRITE OPERATION
                SETB EN             ; EN=1 FOR HIGH PULSE
                ACALL DELAY         ; GIVE LCD SOME TIME
                CLR EN              ; EN=0 FOR HIGH TO LOW PULSE
                RET
DELAY:          MOV R5, #255
HERE:           MOV R6, #255
                DJNZ R6, $
                DJNZ R5, HERE
                RET
COMM:           DB 38H, 0CH, 01H, 06H, 84H, 0     ; COMMANDS
MSG:            DB "WINEYARD", 0                   ; DATA
MSG1:           DB "TECHNOLOGIES", 0               ; DATA
                END
```

RESULT:

The message displayed on the LCD screen is observed.

VIVA QUESTIONS:

1. What LCD stands for and what is it principle
2. What is the advantage of LCD over LED
3. What are the applications of LCD
4. What is command mode and data mode of LCD
5. What are the various commands used in LCD.

Experiment No.12

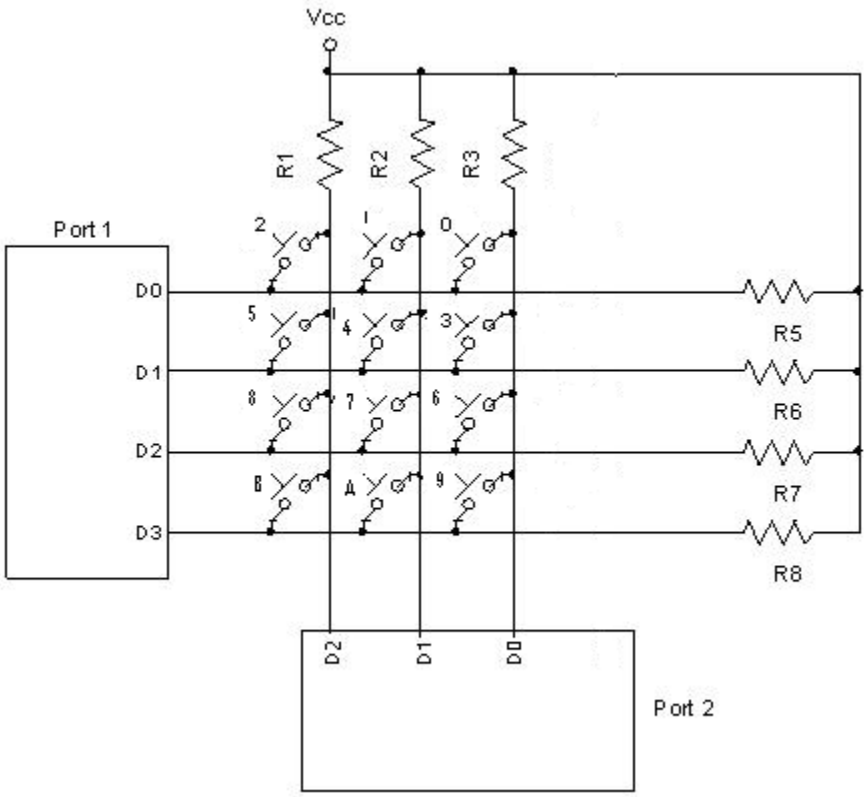
12. INTERFACING MATRIX / KEY BOARD TO 8051

AIM: To display message on the LCD.

EQUIPMENT REQUIRED:

1. 8051 Trainer kit
2. RS232 cable
3. Max 232 IC
4. Keil µ Vision 3 software

HARDWARE CONNECTIONS REQUIRED:



LED7 --- LED0 - P0.7 ------ P0.0

KEYPAD ROWS - P1.0, P1.1, P1.2, P1.3
KEYPAD COLUMNS - P2.0, P2.1, P2.2, P2.3

PROCEDURE:

1. Write the assembly language program for this task in the keil compiler.
2. Compile it and dump the hex file as per the procedure given above.
3. Place this Microcontroller in the trainer kit and fix it properly.
4. Power up the trainer kit from the mains supply.
5. In order to establish communication between PC and Microcontroller, Serial cable RS232 and Voltage converter MAX232 IC are used. This section is available on the system board.
6. For serial communication, port 3(pins 10 and 11 of Microcontroller) are used as reception and transmission pins respectively.
7. The pins 10 and 11 of Microcontroller are connected to the pins 12 and 11 of MAX232 IC respectively.
8. Connect one end of the serial cable to the processor and the other end to the DB9 pin provided on the system board.
9. Check the connections before switching on the mains.
10. An LED is provided in the power supply section of the trainer kit to indicate if the board is powered or not.
11. After the board is powered up , open the Hyperterminal window in the PC.
12. According to the program written in the Microcontroller, the characters entered in the Hyperterminal window in the PC can be seen on the LCD display i.e., the Microcontroller receives the data from PC.

PROGRAM:

```
                                    ; KEYCOUNT EQU 20H.0
        ORG 00H
        MOV P2, # 0FFH              ; MAKE P2 AS INPUT PORTS
K1:     MOV P1, #0                  ; GROUND ALL ROWS AT ONCE
        MOV A, P2    ; READ ALL COLUMNS ENSURE ALL KEYS OPEN
        ANL A, #00001111B; MASK UNUSED BITS
        CJNE A, #00001111B, K1      ; CHECK TILL ALL KEYS RELEASED
K2:     ACALL DELAY                 ; CAL 20MS DELAY
        MOV A, P2                   ; SEE IF ANY KEY IS PRESSED
         ANL A, #00001111B          ; MASK UNUSED BITS
          CJNE A, #00001111B, OVER  ; KEY PRESSED, AWAIT CLOSURE
          SJMP K2                   ; CHECK IF KEY PRESSED

OVER:   ACALL DELAY          ; WAIT 20 MS DEBOUNCE TIME
        MOV A, P2            ; CHECK KEY CLOSURE
```

```
                ANL A, #00001111B       ; MASK UNUSED BITS
                CJNE A, #00001111B, OVER1; KEY PRESSED FIND ROW
                 SJMP K2

OVER1:          MOV P1, #11111110B      ; GROUND ROW 0
                MOV A, P2               ; READ ALL COLUMNS
                ANL A, #00001111B       ; MASK UNUSED BITS
                CJNE A, #00001111B, ROW_0; KEY ROW 0 FIND COLUMN
                MOV P1, #11111101B      ; GROUND ROW 1
                MOV A, P2               ; READ ALL COLUMNS
                ANL A, #00001111B       ; MASK UNUSED BITS
                CJNE A, #00001111B, ROW_1; KEY ROW 1 FIND COLUMN
                 MOV P1, #11111011B     ; GROUND ROW 2
                MOV A, P2               ; READ ALL COLUMNS
                ANL A, #00001111B       ; MASK UNUSED BITS
                CJNE A, #00001111B, ROW_2; KEY ROW 2 FIND COLUMN
                MOV P1, #11110111B      ; GROUND ROW 3
                MOV A, P2               ; READ ALL COLUMNS
                ANL A,#00001111B        ; MASK UNUSED BITS
                CJNE A,#00001111B,ROW_3 ; KEY ROW 3 FIND COLUMN
                LJMP K2                 ; IF NONE FALSE INPUT, REPEAT

          ROW_0:MOV DPTR, # KCODE0  ; SET DPTR=START OF ROW 0
                 SJMP FIND          ; FIND COLUMN KEY BELONGS TO
          ROW_1:MOV DPTR, # KCODE 1 ; SET DPTR=START OF ROW 1
                 SJMP FIND          ; FIND COLUMN KEY BELONGS TO

          ROW_2:MOV DPTR, # KCODE 2  ; SET DPTR=START OF ROW 2
                 SJMP FIND          ; FIND COLUMN KEY BELONGS TO

          ROW_3:MOV DPTR, # KCODE 3; SET DPTR=START OF ROW 3
                 SJMP FIND          ; FIND COLUMN KEY BELONGS TO

           FIND: RRC A              ; SEE ANY CY BIT IS LOW
                 JNC MATCH          ; IF ZERO GET THE ASCII CODE
                 INC DPTR           ; POINT TO NEXT COLUMN ADDRESS
                 SJMP FIND          ; KEEP SEARCHING


              MATCH:CLR A           ; SET A=0(MATCH IS FOUND)
                     MOVC A,@A+DPTR ; GET ASCII CODE FROM THE TABLE

                    MOV P0, A       ; DISPLAY PRESSED KEY
                    LJMP K1         ; ASCII LOOK-UP TABLE FOR EACH ROW
```

```
DELAY:      MOV R5, #255
HERE:       MOV R6, #255
            DJNZ R6, $
            DJNZ R5, HERE
            RET

            ORG 300H
KCODE0:     DB   '0','1','2','3'      ; ROW 0
KCODE1:     DB   '4','5','6','7'      ; ROW 1
KCODE2:     DB   '8','9','A','B'      ; ROW 2
KCODE3:     DB   'C','D','E','F'      ; ROW 3
            END
```

RESULT: The Key press is identified and display

VIVA QUESTIONS:

1. What are the various type of matrix keyboards
2. What is the logic level that is taken for the no key press.
3. What is the lookup table and its advantages;