Experiment No.1

ARITHMETIC OPERATIONS

AIM : Write and execute an assembly language program to 8086 processor to add, subtract and multiply two 16 bit unsigned numbers.

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. Load the words into registers and perform addition/ subtraction/ multiplication/ division and store the sum/difference/product/quotient-remainder to the result address
4. Terminate the program

PROGRAM:

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
```

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

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**?**

Experiment No.2

UNSIGNED WORD BY BYTE AND DOUBLE WORD BY WORD DIVISION

AIM: Write and execute an assembly language program to 8086 processor to divide word by byte and double word by word unsigned numbers.

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. Load the word of divided into word register and divide with byte
4. Load the byte of divided into byte register and clear the higher order byte of that register and then divide with byte.
5. Load the double word of divided into registers and divide with word
6. Load the word of dividend into lower order register and clear the higher order register and then divide with word
7. Terminate the program

PROGRAM:

```
 1                                      assume cs:code,ds:data
 2
 3   0000                               data segment
 4
 5   0000  0100                         word1 dw 0100h
 6   0002  20                 byte1 db 20h
 7
 8
 9   0003  0000                         dwordh  dw      0000h
10   0005  1000                         dwordl  dw      1000h
11
12   0007  02*(0024)          quotr dw 2        dup('$')
13   000B  02*(0024)          dquot  dw 2 dup('$')
14   000F  02*(0024)          drem dw 2 dup('$')
15
16   0013                               data ends
17
18   0000                               code segment
19
20   0000                               start:
21   0000  B8 0000s                     mov ax,data
22   0003  8E D8                              mov ds,ax
23
```

```
24   0005  A1 0000r          mov ax,word1
25   0008  F6 36 0002r               div byte1
26   000C  A3 0007r          mov quotr,ax
27                           mov quom,dx
28   000F  A1 0005r          mov ax,dwordl
29   0012  8B 16 0003r               mov dx,dwordh
30   0016  F7 36 0000r               div word1
31   001A  A3 000Br          mov dquot,ax
32   001D  89 16 000Fr               mov drem,dx
33
34
35
36   0021  BE 0007r          lea si,quotr
37   0024  BF 000Br          lea di,dquot
38   0027  BB 000Fr          lea bx,drem
39
40   002A  CC                        int 3h
41
42   002B                    code ends
43                           end start
```

RESULT: drem=0000
Quotr=0008
Dquot=0010

VIVA QUESTIONS:

1. What is the syntax of unsigned division instruction?
2. What is the logic for division without using div instruction?
3. What is the implicit register for dividend when the divisor is of type byte and how the result is stored ?
4. What is the implicit register for dividend when the divisor is of word and how the result is stored ?
5. What are maskable interrupts?

Experiment No.3

A. ASCENDING ORDER

AIM : Write and execute an assembly language program to 8086 processor to sort the given array of 16 bit 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 : Write and execute an assembly language program to 8086 processor to sort the given array of 16 bit 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.4

PICK THE MEDIAN

AIM: Program Write and execute an assembly language program to 8086 processor to pick the median from the given array of numbers

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. Load the word of divided into word register and divide with byte
4. Load the byte of divided into byte register and clear the higher order byte of that register and then divide with byte.
5. Load the double word of divided into registers and divide with word
6. Load the word of dividend into lower order register and clear the higher order register and then divide with word
7. 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=0198 , SI=0002 ; D 0002 0003 98 01

VIVA QUESTIONS:

1. What is the counter for loop instruction
2. What are the segment and ends directives
3. What are data definition directives

Experiment No.5

LENGTH OF THE STRING

AIM : Write and execute an assembly language program to 8086 processor to find the length of a given string which terminates with a special character.

EQUIPMENT REQUIRED:

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

ALGORITHM:

1. Define the string of which the length to be calculated in data segment
2. Initialize the data segment register with data segment address
3. Clear registers required
4. Compare the first byte of the string with $, if not equal, increment the register for count and register for accessing the next value.
5. If equal, exit the loop and check the count register.
6. Terminate the program

PROGRAM:

```
 1                          assume cs:code,ds:data
 2
 3  0000                    data segment
 4  0000  61 75 72 6F 72 61       24   n1 db "aurora","$"
 5  0007                    data ends
 6
 7  0000                    code segment
 8  0000                    start:
 9  0000  B8 0000s                  mov ax,data
10  0003  8E D8             mov ds,ax
11  0005  33 C0             xor ax,ax
12  0007  33 D2             xor dx,dx
13  0009  BE 0000r          lea si,n1
14  000C  8A 04             l1:mov al,[si]
15  000E  3C 24             cmp al,"$"
16  0010  74 05             jz    l2
17  0012  FE C3             inc bl
18  0014  46                inc si
19  0015  EB F5             jmp l1
20  0017  B4 09             l2:mov ah,09h
```

```
21   0019  BA 0000r        lea dx,n1
22   001C  CD 21           int 21h
23   001E  CC              int 3h
24   001F                  code ends
25                         end start
```

RESULT:
BL = 06

VIVA QUESTIONS:

1. What are the flags effected for compare instruction
2. Example of unconditional branch instruction

Experiment No.6

A.REVERSE THE STRING READ FROM THE KEYBOARD AND DISPLAY

AIM : Write and execute an assembly language program to 8086 processor to reverse the given 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:

```
 1                                      assume cs:code,ds:data
 2
 3  0000                                data segment
 4
 5  0000  20 65 6E 74 65 72 20+         msg db " enter the string :  ","$"
 6        74 68 65 20 73 74 72+
 7        69 6E 67 20 3A 20 24
 8
 9  0015                                str1 label byte
10  0015  14                            strmax db 20
11  0016  ??                            stract db ?
12  0017  0A*(24)                       strfld db 10 dup('$')
13
14  0021  0A 0D 24                      nline db 10,13,'$'
15
```

```
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
```

```
62
63
64  0031  B4 09                        mov ah,09h
65  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?

B. CHECK WHETHER THE GIVEN STRING IS PALINDROME

AIM : Write and execute an assembly language program to 8086 processor to verify 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","$"
 6        68 65 20 73 74 72 69+
 7        6E 67 24
 8
 9  0011                                str1 label byte
10  0011  14                            strmax db 20
11  0012  ??                            stract db ?
12  0013  0A*(24)                       strfld db 10 dup('$')
```

```
                                        nline db 10,13,'$'

0020  0A*(24)                           rev db 10 dup('$')

002A  74 68 65 20 65 6E 74+             msg1 db "the entered string
                                                is palindrome","$"
      65 72 65 64 20 73 74+
      72 69 6E 67 20 69 73+
      20 70 61 6C 69 6E          64+
      72 6F 6D 65 24
004B  74 68 65 20 65 6E          74+    msg2 db "the entered string
                                                is not palindrome","$"
      65 72 65 64 20 73 74+
      72 69 6E 67 20 69 73+
      20 6E 6F 74 20 70 61+
      6C 69 6E 64 72 6F          6D+
      65 24


0070                                    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 09                                     mov ah,09h
000E  BA 001Dr                                  lea dx,nline
0011  CD 21                                     int 21h

0013  B4 0A                                     mov ah,0ah
0015  BA 0011r                                  lea dx,str1
0018  CD 21                                     int 21h

001A  B4 09                                     mov ah,09h
001C  BA 001Dr                                  lea dx,nline
001F  CD 21                                     int 21h

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.7

VERIFY THE PASSWORD

AIM: Write and execute an assembly language program to 8086 processor to verify the password.

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.8

A.INSERT A STRING AND DISPLAY

AIM : Write and execute an assembly language program to 8086 processor to insert the string and display

EQUIPMENT REQUIRED:

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

ALGORITHM:

1. Define the strings in data segment
2. Initialize the data segment register with data segment address
3. Initialize the source pointer to the first string
4. Initialize the destination pointer to the location where the total string is to be stored
5. Increment the source pointer and compare the source contents with ' ' (blank space)
6. If not equal, continue the comparison and move the contents from source to the destination
7. If equal, then initialize the another source pointer to the address of the string to be inserted
8. Increment the destination pointer and move the string till the termination character is found
9. Increment the first source pointer as well as destination pointer and move the contents till the termination character is found
10. Display the total string
11. Terminate the program

PROGRAM:

```
1                                   assume cs:code,ds:data
2
3   0000                            data segment
4
5   0000  67 6F 6F 64 20 69    6E+      msg db "good india","$"
6   64 69 61 24
7
8   000B  0A 0D 24                      nline db 10,13,'$'
9
10
11  000E  20 6D 6F 72 6E 69    6E+      msg2 db " morning","$"
12  67 24
```

```
0017  28*(24)          msg1 db 40 dup('$')

003F                   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  BE 0000r                 lea si,msg
000F  BF 0017r                 lea di,msg1

0012  8A 04              l1:   mov al,[si]
0014  3C 20                    cmp al,' '
0016  74 06                    je l2

0018  88 05                    mov [di],al
001A  46                       inc si
001B  47                       inc di
001C  EB F4                    jmp l1

001E  BB 000Er                 l2:     lea bx,msg2

0021  8A 07              l3:   mov al,[bx]
0023  3C 24                    cmp al,'$'
0025  74 06                    je l4
0027  88 05                    mov [di],al
0029  43                               inc bx
002A  47                       inc di
002B  EB F4                    jmp l3

002D                           l4:
002D  8A 04                      mov al,[si]
002F  3C 24                      cmp al,'$'
0031  74 06                      je l5
0033  88 05                      mov [di],al
0035  46                                 inc si
0036  47                                 inc di
0037  EB F4                      jmp l4

0039  B4 09                    l5:mov ah,09h
003B  BA 000Br                           lea dx,nline
003E  CD 21                      int 21h
```

```
63
64   0040  B4 09                mov ah,09h
65   0042  BA 0017r             lea dx,msg1
66   0045  CD 21                int 21h
67
68   0047  B4 4C                mov ah,4ch
69   0049  CD 21                int 21h
70   004B                       code ends
71                              end start
```

RESULT: good morning india

VIVA QUESTIONS:

1. What are the instructions used to set and reset the direction flag is a palindrome?
2. What is the use of index registers?
3. What is the base register?
4. What are unconditional branch instructions, specify some of them?
5. What are ptr and label directives?

B. DELETE A STRING AND DISPLAY

AIM : Write and execute an assembly language program to 8086 processor to delete a string from the given string and display

EQUIPMENT REQUIRED:

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

ALGORITHM:

1. Define the string in data segment
2. Initialize the data segment register with data segment address
3. Initialize the source pointer to the first string and initialize the destination pointer to new address
4. Increment the source pointer and compare the source contents with blank space
5. If not equal, continue the comparison and move the contents from source to destination
6. If equal, increment the source pointer
7. Now again compare the source contents with blank space
8. If not equal, continue the comparison and do not move the string.
9. If equal, then increment the source pointer and destination pointer and move the string from source to destination till termination character is found
10. Terminate the program

PROGRAM:

```
 1                                  assume cs:code,ds:data
 2
 3   0000                           data segment
 4
 5   0000  67 6F 6F 64 20 6D        6F+     msg db "good morning    india","$"
 6         72 6E 69 6E 67 20  69+
 7         6E 64 69 61 24
 8
 9   0013  0A 0D 24                 nline db 10,13,'$'
10
11   0016  28*(24)                  msg1 db 40 dup('$')
12
13   003E                                data ends
14
15
16
17   0000                                code segment
18
```

```
19   0000  B8 0000s          start:mov ax,data
20   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 0016r                lea di,msg1
28
29   0012  8A 04                   l1:    mov al,[si]
30   0014  3C 20                   cmp al,' '
31   0016  74 06                   je l2
32
33   0018  88 05                   mov [di],al
34   001A  46                      inc si
35   001B  47                      inc di
36   001C  EB F4                   jmp l1
37
38   001E  46                   l2: inc si
39   001F  8A 04                   mov al,[si]
40
41   0021  3C 20                      cmp al,' '
42   0023  74 02                      je l3
43   0025  EB F7                      jmp l2
44
45   0027                          l3:
46   0027  8A 04                      mov al,[si]
47   0029  3C 24                      cmp al,'$'
48   002B  74 06                   je l4
49   002D  88 05                      mov [di],al
50   002F  46                         inc si
51   0030  47                         inc di
52   0031  EB F4                      jmp l3
53
54   0033  B4 09                   l4:    mov ah,09h
55   0035  BA 0013r                lea dx,nline
56   0038  CD 21                      int 21h
57
58   003A  B4 09                      mov ah,09h
59   003C  BA 0016r                   lea dx,msg1
60   003F  CD 21                      int 21h
61
62   0041  B4 4C                      mov ah,4ch
63   0043  CD 21                      int 21h
64   0045                          code ends
65                                 end start
```

RESULT: good india

VIVA QUESTIONS:

1. Mention the different index registers that are used as offset for all 4 segment registers.
2. What is the use of segmentation?
3. What are the memory banks of 8086?
4. What are the lines used for selecting those banks?
5. What are interrupts?

Experiment No. 10

LED INTERFACE

AIM: Write and execute an assembly language program to 8086 processor to call a delay subroutine and display the character on the LED display;

EQUIPMENT REQUIRED:

1. 8086 microprocessor trainer kit
2. 8255 interfacing card with LEDs and switches
3. Flat ribbon cable bus
4. Keyboard
5. Power chord

HARDWARE CONNECTIONS REQUIRED:

PORT A is connected LEDs.

PROCEDURE:

1. Initialize 8255 by writing the control word into it.
2. load the accumulator and send out through port A to make the LED glow
3. Introduce the delay
4. continue from start

PROGRAM:

```
MOV AL,80
MOV DX,3006        ;Initialize 8255
OUT DX,AL


MOV AL,55
MOV  DX,3002       ; Send the data to glow LEDs
OUT DX,AL

MOV  CX,FF         ; delay
L1 :LOOP L1

JMP START          ; Continue from start

INT  3  ; Terminate
```

RESULT:

The LEDs are glowing according to the status of switches

VIVA QUESTIONS:

1. What are the types of LED display
2. What is the advantage using LEDs
3. What is the disadvantage using LEDs
4. What are the type of switches.

Experiment No. 10

STEPPER MOTOR

AIM: Program to interface the stepper motor to 8086 microprocessor and operate it in clockwise and anti-clockwise by choosing variable step-size

EQUIPMENT REQUIRED:

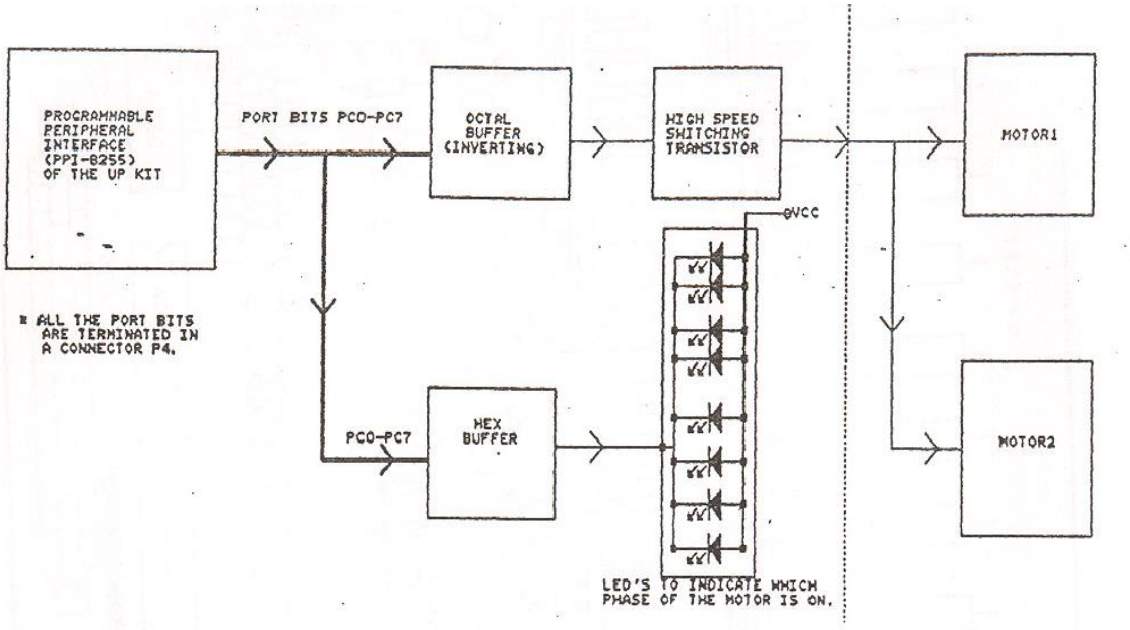
1. 8086 microprocessor trainer kit
2. Stepper motor interfacing module
3. Flat ribbon cable bus
4. Keyboard
5. Power chord

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:

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:

```
        MOV AL,80          ; Initialize 8255
        MOV DX,FFC6
        OUT DX,AL

START:MOV AL,EE            ; Byte to switch on A phase
        MOV DX,FFC4
        OUT DX,AL

        CALL DELAY         ; Wait

        MOV AL,DD          ; Byte to switch on B phase
        MOV DX,FFC4
```

```
        OUT DX,AL

        CALL DELAY      ; Wait

        MOV AL,BB       ; Byte to switch on C phase
        MOV DX, FFC4
        OUT DX,AL

        CALL DELAY      ; Wait

        MOV AL,77       ; Byte to switch on D phase
        MOV DX,FFC4
        OUT DX, AL

        CALL DELAY      ; Wait

        JMP START       ; Go to start

DELAY:MOV CX,0FFFF      ; Initialize counter (Hex value)
L1:     NOP             ; No operation

        NOP
        NOP
        DEC CX
        JNZ L1
        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. 11

A TO D CONVERTER

AIM: Interface an 8 bit ADC to 8086 and generate digital output and store it in memory for the given waveform input.

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:

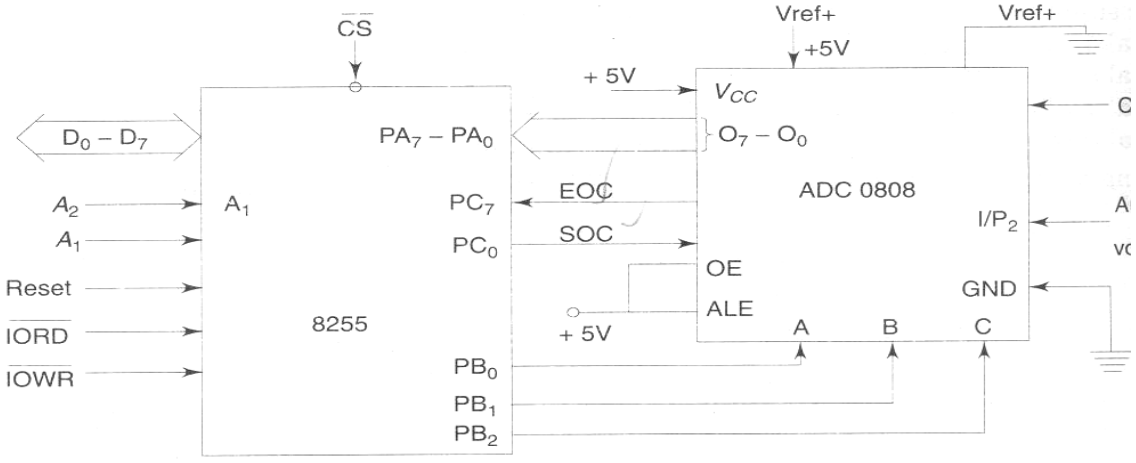
a) connect J_2 to provide 8 channels of ADC which are selected by address supplied by port B (J_3) and latched by Pc_4 bit.

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:

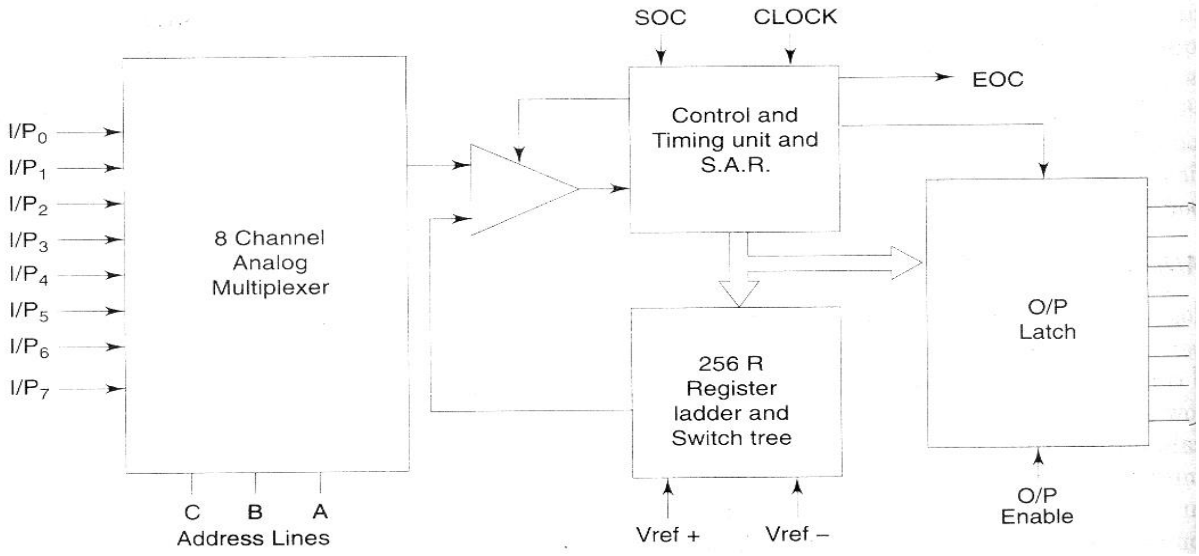


DESCRIPTION ABOUT IC's:

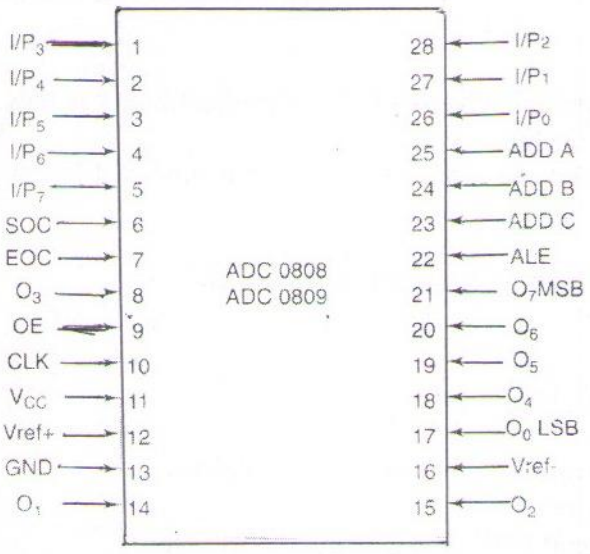
ADC 0808:

* 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 for selecting inputs
$O_7 - O_0$	–	Digital 8-bit output with O_7 O_0 LSB
SOC	–	Start of conversion signal p
EOC	–	End of conversion signal p
OE	–	Output latch enable pin if 1 output
CLK	–	Clock input for ADC
V_{CC} GND	–	Supply pins +5V and GND
$Verf^+$ and $Vref^-$	–	Reference voltage positive maximum and Reference voltage neg minimum

$I/P_0 – I/P_7$ – 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.

Experiment No. 12

D TO A CONVERTER

AIM: Interface an DAC to 8086 and generate a Triangular waveform and Square waveform with different periods.

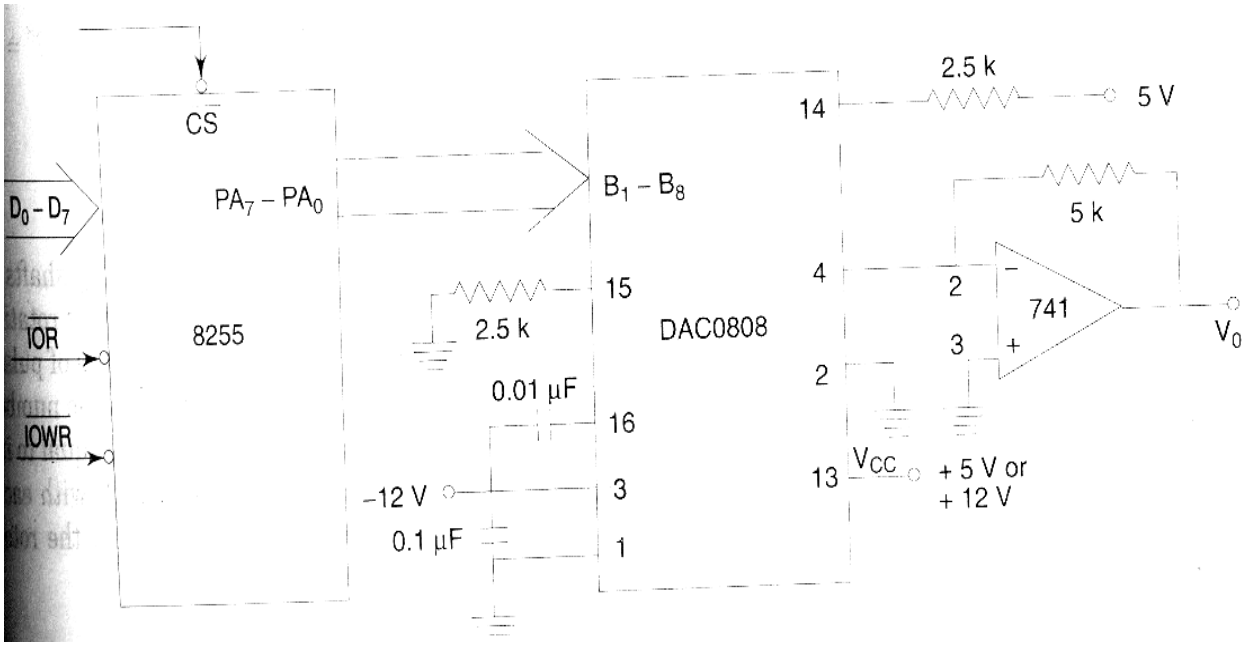
EQUIPMENT REQUIRED:

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:

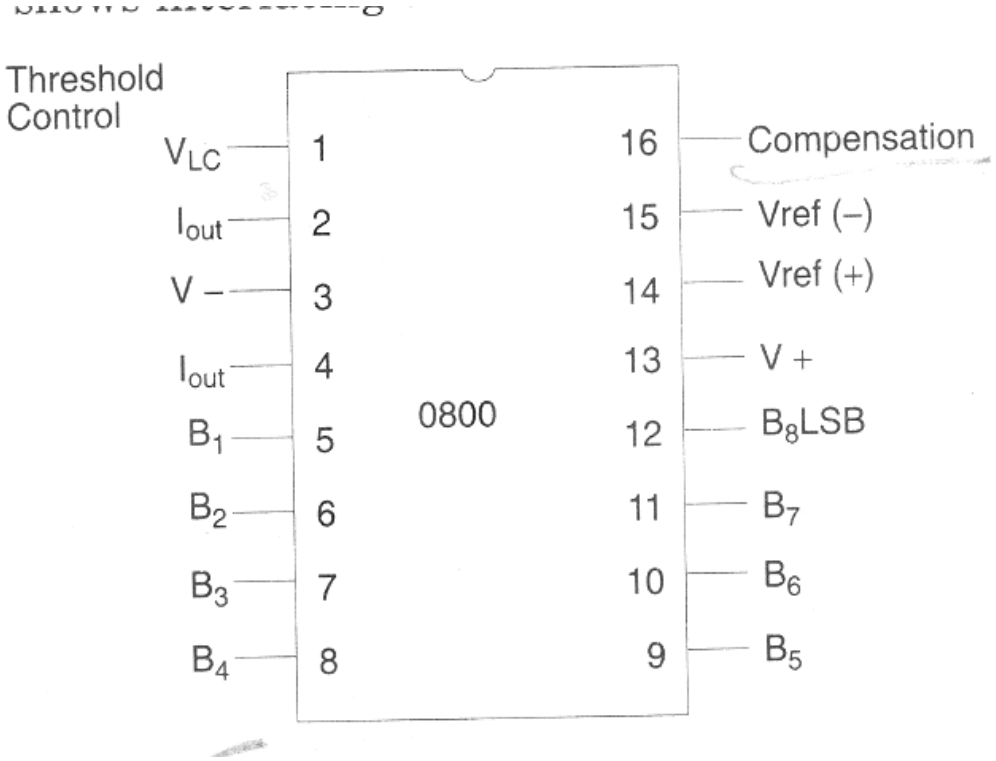


DESCRIPTION ABOUT IC'S:

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

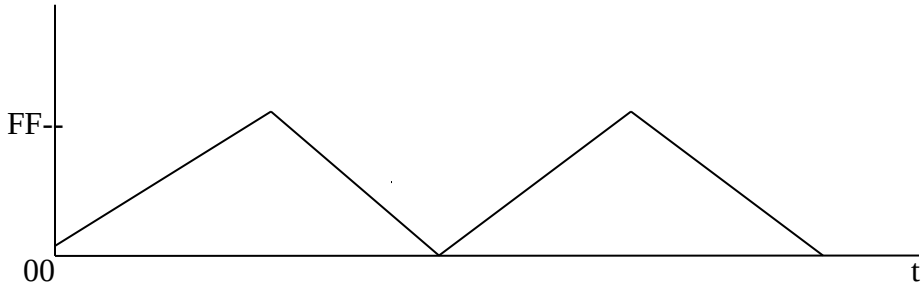
STEPS INVOLVED IN INTERFACING:

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
```

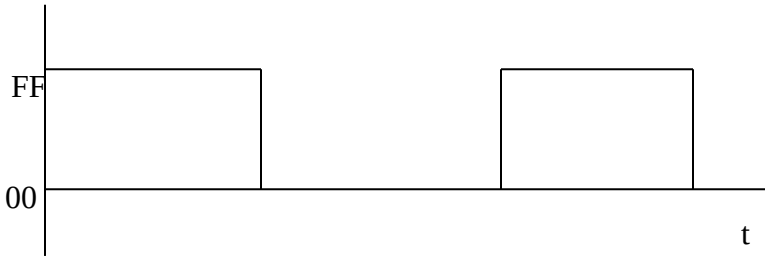
EXPECTED GRAPH



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