Microprocessors and Peripherals

Lab Programs - 2019

Please Note:

- This document contains all lab programs of microprocessors and peripherals along with the output of all software programs.
- Programs are taken from official manual **MP(16CS405) Lab Manual_2018 (**Prepared by : Anisha P Rodrigues, Assistant Professor-II, Computer Science and Engineering, NMAMIT, Nitte). Only changes made in the text formatting.
- This document is unofficial and only for the purpose of reference.

Prepared by:
Shawn Linton Miranda
4NM17CS164

Index

Software Programs:

Hardware Programs:

Software Programs :

Commands used:

EDIT file_name.ASM - To create a file and type assembly code
MASM file_name.ASM; - To assemble the code using masm assembler
LINK file_name.OBJ; - To link object file with code in order to create executable file
file_name.EXE - To run the assembly code

1.Binary search

Search a key element in a list of 'n' 8-bit numbers using the Binary search algorithm.

Program:

```
DATA SEGMENT
    ARR DB 10H,20H,30H,40H,50H,60H
    LEN DB $-ARR
    MID DB ?    ;Variable to hold position of mid element
    KEY DB 50H
    MSG1 DB 10,13,'Key not found.$'
    MSG2 DB 10,13,'Key found at position $'
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE,DS:DATA
START:
    MOV AX,DATA     ;store address of data segment in DS register
    MOV DS,AX
    MOV DL,00H     ;Lower limit in array
    MOV DH,LEN     ;Upper limit in array
    DEC DH
    MOV BX,0000H
    MOV CL,KEY
    UP:
        CMP DL,DH     ;while low<=high
        JG NOTFOUND
        MOV BL,DL
        ADD BL,DH
        SHR BL,01H    ;get mid position
        MOV MID,BL
        CMP CL,ARR[BX]  ;compare key with mid element
        JZ FOUND
        JB FIRSTHALF   ;If key<mid element consider first half of array
        INC MID
        MOV DL,MID    ;Otherwise consider second half
        JMP UP
    FIRSTHALF:
        DEC MID
        MOV DH,MID
        JMP UP
    NOTFOUND:
        LEA DX,MSG1    ;If key not found
        MOV AH,09H
        INT 21H
        JMP EXIT
```

```
        FOUND:
            LEA DX,MSG2
            MOV AH,09H
            INT 21H
            MOV BL,MID
            INC BL
            CALL DISPHEXA      ;Display the position
        EXIT:
            MOV AH,4CH   ; end of program
            INT  21H

        DISPHEXA PROC NEAR     ;Procedure to display 2 digit hexadecimal numbers
            MOV DL,BL
            MOV CL,04H
            SHR DL,CL
            CMP DL,09H
            JBE L1
            ADD DL,07H
          L1:
            ADD DL,30H
            MOV AH,02H
            INT 21H
            MOV DL,BL
            AND DL,0FH
            CMP DL,09H
            JBE L2
            ADD DL,07H
           L2:
            ADD DL,30H
            MOV AH,02H
            INT 21H
            RET
        DISPHEXA ENDP
    CODE ENDS
    END START
```

Output:

When key is 50H

```
C:\>binsrch.exe

Key found at position 05
```

When key is 13H

```
C:\>binsrch.exe

Key not found.
```

2.Read and Display string

Write ALP macros:

i. To read a character from the keyboard in the module (1) (in adifferent file).

ii. To display a character in module(2) (in different file).

iii. Use the above two modules to read a string of characters from the keyboard terminated by the carriage return and print the string on the display in the next line.

Program:

F1.MAC (In different file)

```
READCHAR MACRO
    MOV AH,01H
    INT 21H
ENDM
```

F2.MAC (In different file)

```
DISPCHAR MACRO
    MOV AH,02H
    INT  21H
ENDM
```

```
INCLUDE F1.MAC     ;include macro READCHAR
INCLUDE F2.MAC    ;include macro DISPCHAR
```

```
DATA SEGMENT
      S DB 50 DUP(?)  ;Create empty array with 50 locations
      LEN DB ?
      MSG1 DB 10,13,'ENTER A STRING:',10,13,'$'
      MSG2 DB 10,13,'ENTERED STRING IS:',10,13,'$'
DATA ENDS

CODE SEGMENT
       ASSUME CS:CODE,DS:DATA
START:
      MOV AX,DATA
      MOV DS,AX

      LEA DX,MSG1
      MOV AH,09H
      INT 21H
      CALL READSTRING  ;read a string
      LEA DX,MSG2
      MOV AH,09H
      INT 21H
      CALL DISPSTRING   ;display read string
      MOV AH,4CH
      INT 21H

    READSTRING PROC NEAR    ;procedure to read a string
      MOV CL,00
      LEA SI,S
      UP1:CMP CL,50
            JZ L1
            READCHAR      ;invoke macro to read a character
            CMP AL,0DH
            JZ L1
            MOV [SI],AL
            INC SI
            INC CL
            JMP UP1
      L1:  MOV LEN,CL
            RET
     READSTRING ENDP

    DISPSTRING PROC NEAR     ;procedure to display a string
      MOV CL,00
      LEA SI,S
      UP2: CMP CL,LEN
            JZ L1
            MOV DL,[SI]
            DISPCHAR     ;invoke macro to display a character
            INC SI
            INC CL
            JMP UP2
      L2:   RET
     DISPSTRING ENDP
CODE ENDS
END START
```

Output:

```
C:\>str.exe

Enter a string:
Microprocessors and Peripherals

Entered string is:
Microprocessors and Peripherals
```

3.Bubble sort

Sort a given set of 'n' numbers in ascending order using the Bubble Sort algorithm.

Program:

```
DATA SEGMENT
      A DB 61H,38H,05H,91H,82H,03H
      LEN DB $-A
      NL DB 10,13,'$'
DATA ENDS

CODE SEGMENT
      ASSUME CS:CODE,DS:DATA
START:
      MOV AX,DATA
      MOV DS,AX
      DEC LEN
      MOV CL,00H
  UP2:                    ;outer loop
      CMP CL,LEN
      JZ DISPLAY
      MOV BX,0000H
      MOV DL,LEN
      SUB DL,CL
  UP1:                    ;inner loop
      CMP BL,DL
      JZ D1
      MOV AL,A[BX]
      CMP AL,A[BX+1]
      JBE NOSWAP
      MOV AH,A[BX+1]    ;swap if left operand is greater
      MOV A[BX],AH
      MOV A[BX+1],AL
  NOSWAP:
      INC BL
      JMP UP1
  D1:
      INC CL
      JMP UP2
  DISPLAY:
      MOV CL,00H
      LEA SI,A
  UP3:
      CMP CL,LEN
      JA EXIT
      MOV BL,[SI]
      CALL DISPHEXA
      CALL NEWLINE   ;Display next number in a new line
      INC SI
      INC CL
      JMP UP3
  EXIT:
      MOV AH,4CH
      INT 21H
```

```
DISPHEXA PROC NEAR
      PUSH CX
      MOV DL,BL
      MOV CL,04H
      SHR DL,CL
      CMP DL,09H
      JBE L1
      ADD DL,07H
   L1:
      ADD DL,30H
      MOV AH,02H
      INT 21H

      MOV DL,BL
      AND DL,0FH
      CMP DL,09H
      JBE L2
      ADD DL,07H
   L2:
      ADD DL,30H
      MOV AH,02H
      INT 21H
      POP CX
      RET
   DISPHEXA ENDP

   NEWLINE PROC NEAR
      LEA DX,NL
      MOV AH,09H
      INT 21H
      RET
   NEWLINE ENDP
CODE ENDS
END START
```

Output:

```
C:\>bublsort.exe
03
05
38
61
82
```

4.Display ASCII value of alphanumeric character

Read an alphanumeric character and display its equivalent ASCII code at the centre of the screen.

Program:

```
F3.MAC   (In different file)
CLRSCR MACRO
      MOV AH,00H  ;function number to clear screen
      MOV AL,02H    ;Video display mode 25x80
      INT 10H      ;Interrupt number
ENDM
```

```
F4.MAC   (In different file)
SETCURSOR MACRO ROW,COL
      MOV DL,COL
      MOV DH,ROW
      MOV AL,02H
      MOV BH,00H  ;current screen
      MOV AH,02H  ;function number
      INT 10H
ENDM
```

```
INCLUDE F3.MAC
INCLUDE F4.MAC
DATA SEGMENT
      MSG1 DB 10,13,'Enter a character : $'
      N DB ?
DATA ENDS
```

```
 CODE SEGMENT
      ASSUME CS:CODE,DS:DATA
START:
      MOV AX,DATA
      MOV DS,AX

      LEA DX,MSG1
      MOV AH,09H
      INT 21H
      MOV AH,01H
      INT 21H
      MOV N,AL
      CLRSCR       ;Invoke macro to clear the screen
      SETCURSOR 12,40       ; Sets the cursor at position 12,40
      MOV BL,N
      CALL DISPHEXA
      MOV AH,01H
      INT 21H
      MOV AH,4CH
      INT 21H

  DISPHEXA PROC NEAR
      MOV DL,BL
      MOV CL,04H
      SHR DL,CL
      CMP DL,09H
      JBE L1
      ADD DL,07H
  L1:ADD DL,30H
      MOV AH,02H
      INT 21H
      MOV DL,BL
      AND DL,0FH
      CMP DL,09H
      JBE L2
      ADD DL,07H
  L2:ADD DL,30H
      MOV AH,02H
      INT 21H
      RET
  DISPHEXA ENDP
CODE ENDS
END START
```

Output:

5.Palindrome

Reverse a given string and check whether it is a palindrome or not.

Program:

```
DATA SEGMENT
      STR1 DB 20 DUP(?)
      STR2 DB 20 DUP(?)
      N DB ?
      M1 DB 10,13,'Enter the string : $'
```

```
        M2 DB 10,13,'String is a palindrome. $'
        M3 DB 10,13,'String is not a palindrome. $'
DATA ENDS

CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
START:
        MOV AX,DATA
        MOV DS,AX

        LEA SI,STR1
        LEA DI,STR2
        LEA DX,M1
        MOV AH,09H
        INT 21H
        CALL READSTRING
        MOV N,CL
        MOV CL,00H
        DEC SI
    UP1:            ;Reverse the string
        CMP CL,N
        JZ CHECK
        MOV AL,[SI]
        MOV [DI],AL
        DEC SI
        INC CL
        INC DI
        JMP UP1
    CHECK:
        LEA SI,STR1
        LEA DI,STR2
        MOV CL,00H
    UP2:            ;Compare each characters of original and reversed string
        CMP CL,N
        JZ PAL
        MOV AL,[SI]
        CMP AL,[DI]
        JNZ NOTPAL   ;If any character is not matching then display not palindrome
        INC SI
        INC DI
        INC CL
        JMP UP2
    NOTPAL:
        LEA DX,M3
        MOV AH,09H
        INT 21H
        JMP EXIT
    PAL:
        LEA DX,M2
        MOV AH,09H
        INT 21H
    EXIT:
        MOV AH,4CH
        INT 21H
```

```
    READSTRING PROC NEAR

        MOV CL,00
     UP:
        MOV AH,01H
        INT 21H
        CMP AL,0DH
        JZ L1
        MOV [SI],AL
        INC SI
        INC CL
        JMP UP
     L1:
        RET
   READSTRING ENDP
  CODE ENDS
  END START
```

Output:

```
C:\>palindrome.exe

Enter the string : MALAYALAM

String is a palindrome.
```

```
C:\>palindrome.exe

Enter the string : Madam

String is not a palindrome.
```

```
C:\>palindrome.exe

Enter the string : cat

String is not a palindrome.
```

6.Comparison of two strings

Read two strings, store them in locations STR1 and STR2. Check whether they are equal or not and display appropriated messages. Also display the length of the stored strings.

Program:

```
DATA SEGMENT
    STR1 DB 20 DUP(?)
    STR2 DB 20 DUP(?)
    N1 DB ?
    N2 DB ?
    M1 DB 10,13,'Strings are equal.$'
    M2 DB 10,13,'Strings are not equal.$'
    M3 DB 10,13,'Enter string-1 : $'
    M4 DB 10,13,'Enter string-2 : $'
    M5 DB 10,13,'Length of string-1 : $'
    M6 DB 10,13,'Length of string-2 : $'
    M7
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE,DS:DATA
START:
    MOV AX,DATA
    MOV DS,AX

    LEA DX,M3
    MOV AH,09H
    INT 21H
    LEA SI,STR1
    CALL READSTRING  ;Read string-1
    MOV N1,CL
    LEA DX,M4
    MOV AH,09H
    INT 21H
    LEA SI,STR2
    CALL READSTRING  ;Read string-2
```

```
    MOV N2,CL

    CMP N1,CL   ;If length of string is not same then ther are not equal
    JNZ NOTEQ
    LEA SI,STR1
    LEA DI,STR2
    MOV CL,00
UP1:            ;Compare correspoding characters of two strings
    CMP CL,N1
    JZ STREQ
    MOV AL,[SI]
    CMP AL,[DI]
    JNZ NOTEQ
    INC SI
    INC DI
    INC CL
    JMP UP1
NOTEQ:
    LEA DX,M2
    MOV AH,09H
    INT 21H
    JMP DISPLEN
STREQ:
    LEA DX,M1
    MOV AH,09H
    INT 21H
DISPLEN:
    LEA DX,M5
    MOV AH,09H
    INT 21H
    MOV BL,N1
    CALL DISPHEXA
    LEA DX,M6
    MOV AH,09H
    INT 21H
    MOV BL,N2
    CALL DISPHEXA
    MOV AH,4CH
    INT 21H

READSTRING PROC NEAR
    MOV CL,00H
UP:
    MOV AH,01H
    INT 21H
    CMP AL,0DH
    JZ L1
    MOV [SI],AL
    INC SI
    INC CL
    JMP UP
L1:
    RET
READSTRING ENDP
```

```
    DISPHEXA PROC NEAR
        MOV DL,BL
        MOV CL,04H
        SHR DL,CL
        CMP DL,09H
        JBE L2
        ADD DL,07H
    L2:
        ADD DL,30H
        MOV AH,02H
        INT 21H
        MOV DL,BL
        AND DL,0FH
        CMP DL,09H
        JBE L3
        ADD DL,07H
    L3:
        ADD DL,30H
        MOV AH,02H
        INT 21H
        RET
    DISPHEXA ENDP
CODE ENDS
END START
```

Output:

```
C:\>strcmp.exe

Enter string-1 : Micro

Enter string-2 : Mic

Strings are not equal.
Length of string-1 : 05
Length of string-2 : 03
```

```
C:\>strcmp.exe

Enter string-1 : ABCD

Enter string-2 : ABCD

Strings are equal.
Length of string-1 : 04
Length of string-2 : 04
```

7.Read and display name

Read your name from the keyboard and display it at a specified location on the screen in front of the message "What is your name?" You must clear the entire screen before display.

Program:

```
INCLUDE F3.MAC
INCLUDE F4.MAC
DATA SEGMENT
    MSG1 DB 'Enter your name : $'
    MSG2 DB 'What is your name?$'
    STR DB 30 DUP(?)
    N DB ?
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE,DS:DATA
START:
    MOV AX,DATA
    MOV DS,AX

    LEA DX,MSG1
    MOV AH,09H
    INT 21H
    LEA SI,STR
    CALL READSTRING
    MOV N,CL
    CLRSCR
    MOV AL,02H
```

F3.MAC (In different file)

```
CLRSCR MACRO
    MOV AH,00H  ;function number to clear screen
    MOV AL,02H  ;Video display mode 25x80
    INT 10H     ;Interrupt number
ENDM
```

F4.MAC (In different file)

```
SETCURSOR MACRO ROW,COL
    MOV DL,COL
    MOV DH,ROW
    MOV AL,02H
    MOV BH,00H  ;current screen
    MOV AH,02H  ;function number
    INT 10H
ENDM
```

```
        SETCURSOR 10,30
        LEA DX,MSG2
        MOV AH,09H
        INT 21H
        LEA SI,STR
        MOV CL,N
        CALL DISPSTRING
        MOV AH,01H
        INT 21H
        MOV AH,4CH
        INT 21H

    READSTRING PROC NEAR
        MOV CL,00H
    UP1:
        CMP CL,30
        JZ L1
        MOV AH,01H
        INT 21H
        CMP AL,0DH
        JZ L1
        MOV [SI],AL
        INC SI
        INC CL
        JMP UP1
    L1:
        RET
    READSTRING ENDP

    DISPSTRING PROC NEAR
    UP2:
        CMP CL,00H
        JZ L2
        MOV DL,[SI]
        MOV AH,02H
        INT 21H
        INC SI
        DEC CL
        JMP UP2
    L2:
        RET
    DISPSTRING ENDP
CODE ENDS
END START
```

Output:

8.Factorial

Compute the factorial of a positive integer 'n' using recursive procedure.

Program:

```
DATA SEGMENT
    N DB 06H
    FACT DW ?
    MSG2 DB 10,13,'FACTORIAL(06) = $'
DATA ENDS
```

```
CODE SEGMENT
      ASSUME CS:CODE,DS:DATA
START:
      MOV AX,DATA
      MOV DS,AX

      MOV AX,0001   ;Initialize AX with multiplicative identity
      MOV BL,N
      MOV BH,00H
      CALL FACTORIAL ;Call factorial function
      MOV FACT,AX

      MOV BL,AH
      LEA DX,MSG2
      MOV AH,09H
      INT 21H
      CALL DISPHEXA ;Display MSB 2 digits
      MOV BX,FACT
      CALL DISPHEXA  ;Display LSB 2 digits
      MOV AH,4CH
      INT 21H

   FACTORIAL PROC NEAR
      CMP BX,01H
      JZ L
      PUSH BX
      DEC BX ;Push value of BX to stack
      CALL FACTORIAL ;Recursive call
      POP BX
      MUL BX
   L:
      RET
  FACTORIAL ENDP

  DISPHEXA PROC NEAR
      PUSH CX
      MOV DL,BL
      MOV CL,04H
      SHR DL,CL
      CMP DL,09H
      JBE L1
      ADD DL,07H
   L1:
      ADD DL,30H
      MOV AH,02H
      INT 21H
      MOV DL,BL
      AND DL,0FH
      CMP DL,09H
      JBE L2
      ADD DL,07H
   L2:
      ADD DL,30H
```

```
        MOV AH,02H
        INT 21H
        POP CX
        RET
    DISPHEXA ENDP
CODE ENDS
END START
```

Output:

```
C:\>fact.exe

FACTORIAL(06) = 02D0
```

9.Fibonacci numbers

Generate the first 'n' Fibonacci numbers.

Program:

```
DATA SEGMENT
        F1 DB 00H      ;First fibonacci number is 00
        F2 DB 01H      ;Second fibonacci number is 01
        F3 DB ?
        N DB 10
        MSG DB 10,13,'The fibonacci sequence is:',10,13,'$'
DATA ENDS

CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
START:
        MOV AX,DATA
        MOV DS,AX

        SPACE MACRO   ;Prints a single blank space
                MOV DL,' '
                MOV AH,02H
                INT 21H
        ENDM

        LEA DX,MSG
        MOV AH,09H
        INT 21H

        MOV CL,00
        MOV BL,F1
        CALL DISPHEXA  ;Display 00
        INC CL
        SPACE    ;print a blank space

        MOV BL,F2
        CALL DISPHEXA  ;Display 01
        INC CL
        SPACE

      UP:
        CMP CL,N
        JZ EXIT
        MOV AL,F1
        ADD AL,F2   ;Add first and second number to get third number
        MOV F3,AL
```

```
        MOV BL,AL
        CALL DISPHEXA
        SPACE
        MOV AL,F2
        MOV F1,AL   ;Replace 1st with second
        MOV AL,F3
        MOV F2,AL  ;Replace 2nd with third
        INC CL
        JMP UP
    EXIT:
        MOV AH,4CH
        INT 21H

    DISPHEXA PROC NEAR
        PUSH CX
        MOV DL,BL
        MOV CL,04H
        SHR DL,CL
        CMP DL,09H
        JBE L1
        ADD DL,07H
    L1:
        ADD DL,30H
        MOV AH,02H
        INT 21H

        MOV DL,BL
        AND DL,0FH
        CMP DL,09H
        JBE L2
        ADD DL,07H
    L2:
        ADD DL,30H
        MOV AH,02H
        INT 21H
        POP CX
        RET
     DISPHEXA ENDP
  CODE ENDS
  END START
```

Output:

```
C:\>fibb.exe

The fibonacci sequence is:
00 01 01 02 03 05 08 0D 15 22
```

10.Display current system time

Read the current time from the system and display it in the standard format on the screen.

Program:

```
  DATA SEGMENT
        HR DB ?
        MIN DB ?
        SEC DB ?
        MSEC DB ?
        MSG DB 'CURRENT SYSTEM TIME  -  $'
  DATA ENDS
```

```
CODE SEGMENT
    ASSUME CS:CODE,DS:DATA
START:
    MOV AX,DATA
    MOV DS,AX

    COLON MACRO     ;Prints a colon on the screen
        MOV DX,':'
        MOV AH,02H
        INT 21H
    ENDM

    MOV AH,2CH  ;Function to read current system time
    INT 21H
    MOV HR,CH    ;Hours stored in CH
    MOV MIN,CL   ;minutes in CL
    MOV SEC,DH   ;seconds in DH
    MOV MSEC,DL  ;Milliseconds in DL

    LEA DX,MSG
    MOV AH,09H
    INT 21H

    MOV AL,HR
    AAM         ;Convert packed to unpacked BCD
    MOV BX,AX
    CALL DISPUNPACKEDBCD
    COLON

    MOV AL,MIN
    AAM
    MOV BX,AX
    CALL DISPUNPACKEDBCD
    COLON

    MOV AL,SEC
    AAM
    MOV BX,AX
    CALL DISPUNPACKEDBCD
    COLON

    MOV AL,MSEC
    AAM
    MOV BX,AX
    CALL DISPUNPACKEDBCD

    MOV AH,4CH
    INT 21H

  DISPUNPACKEDBCD PROC NEAR   ;Displays unpacked BCD numbers
    MOV DL,BH
    ADD DL,30H
    MOV AH,02H
    INT 21H
```

```
        MOV DL,BL
        ADD DL,30H
        MOV AH,02H
        INT 21H
        RET
    DISPUNPACKEDBCD ENDP
  CODE ENDS
  END START
```

Output:

```
C:\>currtime.exe
CURRENT SYSTEM TIME  -  16:22:39:16
```

11.Decimal up counter 00-99

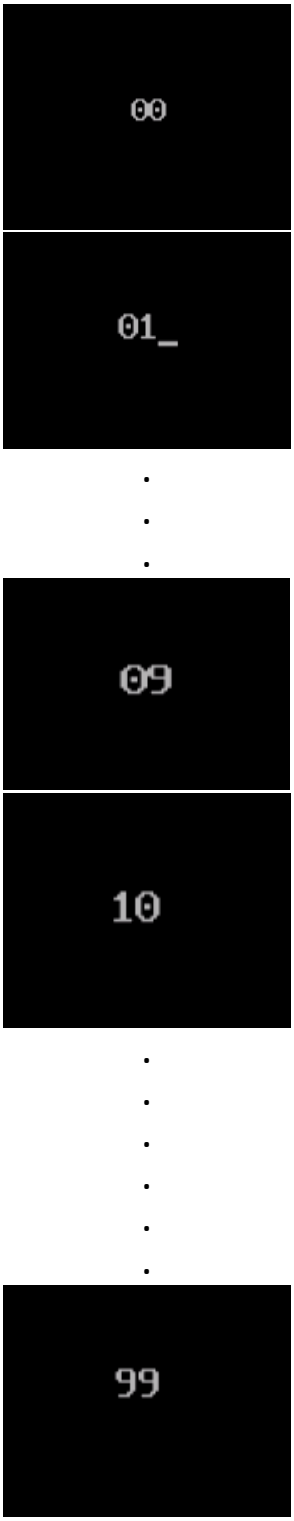
Program to simulate a Decimal Up-counter to display 00-99.

Program:

```
  CLRSCR MACRO
        MOV AH,00H
        MOV AL,02H
        INT 10H
  ENDM

  CODE SEGMENT
        ASSUME CS:CODE
  START:
        CLRSCR
        MOV AL,00H
     UP:            ;Display 0-98
        CMP AL,99H
        JZ DISP99
        CALL CENTER
        MOV BL,AL
        CALL DISPLAY
        CALL DELAY
        ADD AL,01H
        DAA     ;decimal adjust accumulator after addition
        JMP UP
     DISP99:        ;display 99
        CALL CENTER
        MOV BL,AL
        CALL DISPLAY
        CALL DELAY
        MOV AH,01H
        INT 21H
        MOV AH,4CH
        INT 21H

    CENTER PROC
        PUSH AX
        MOV DL,39
        MOV DH,12
        MOV BH,00H
        MOV AH,02H
        INT 10H
        POP AX
```

```
        RET
    CENTER ENDP

    DISPLAY PROC
        PUSH AX
        MOV DL,BL
        MOV CL,04H
        SHR DL,CL
        CMP DL,09H
        JBE DOWN1
        ADD DL,07H
      DOWN1:
        ADD DL,30H
        MOV AH,02H
        INT 21H
        MOV DL,BL
        AND DL,0FH
        CMP DL,09H
        JBE DOWN2
        ADD DL,07H
      DOWN2:
        ADD DL,30H
        MOV AH,02H
        INT 21H
        POP AX
        RET
    DISPLAY ENDP

    DELAY PROC      ;delaying process by executing
        PUSH AX       ;instruction having no effect
        PUSH BX
        PUSH CX
        MOV CX,2FFFH
      L1:MOV BX,5FFFH
      L2:DEC BX
        JNZ L2
        LOOP L1
        POP CX
        POP BX
        POP AX
        RET
    DELAY ENDP
  CODE ENDS
  END START
```

Output:



12.Moving cursor to a specified location

Read a pair of input co-ordinates in BCD and move the cursor to the specified location on the screen.

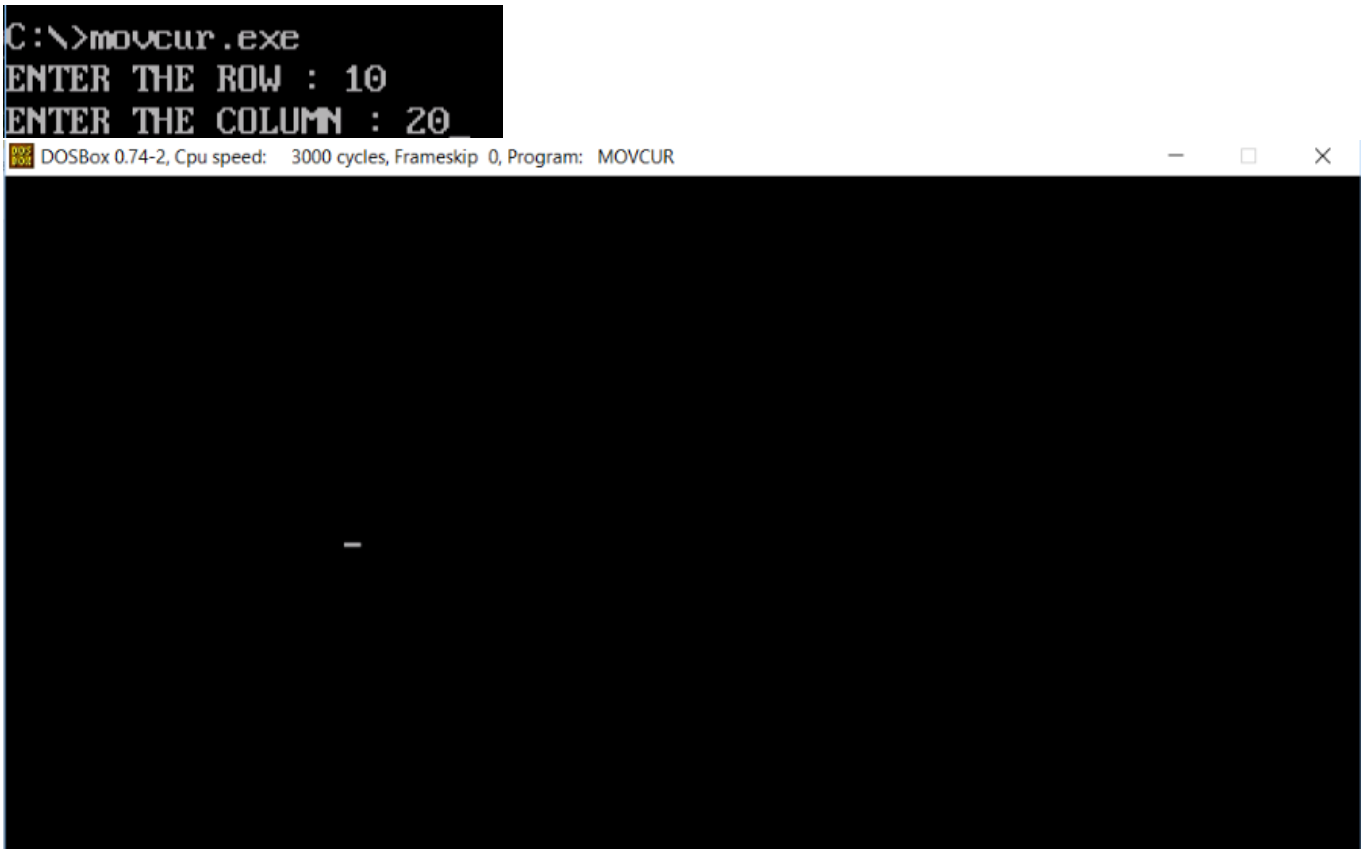
Program:

```
  CLRSCR MACRO
        MOV AH,00
        MOV AL,02
        INT 10H
  ENDM
```

```
SETCURSOR MACRO R,C
        MOV DH,R
        MOV DL,C
        MOV AL,02H
        MOV BH,00H
        MOV AH,02H
        INT 10H
ENDM

DATA SEGMENT
        BCD_R DB ?
        BCD_C DB ?
        BIN_R DB ?
        BIN_C DB ?
        MSG1 DB 'ENTER THE ROW : $'
        MSG2 DB 10,13,'ENTER THE COLUMN : $'
DATA ENDS

CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
START:
        MOV AX,DATA
        MOV DS,AX

        LEA DX,MSG1
        MOV AH,09H
        INT 21H
        CALL READBCD  ;Read row
        MOV BCD_R,BL

        LEA DX,msg2
        MOV AH,09H
        INT 21H
        CALL READBCD  ;Read column
        MOV BCD_C,BL

        MOV BL,BCD_R
        CALL BCD_TO_BIN  ;Convert row
        MOV BIN_R,BL

        MOV BL,BCD_C
        CALL BCD_TO_BIN   ; convert column
        MOV BIN_C,BL

        CLRSCR
        SETCURSOR BIN_R,BIN_C

        MOV AH,01H  ;wait for keypress
        INT 21H

        MOV AH,4CH
        INT 21H
```

```
READBCD PROC NEAR  //To read  the row and column values
    MOV AH,01H
    INT 21H
    MOV BL, AL
    MOV CL,04H
    SHL BL,CL
    MOV AH,01H
    INT 21H
    AND AL,0FH
    ADD BL,AL
    RET
READBCD ENDP

BCD_TO_BIN PROC NEAR  //Converts BCD to hexadecimal
    MOV DH,BL
    AND DH,0FH
    MOV AL, BL
    MOV CL,04H
    SHR AL,CL
    MOV DL,0AH
    MUL DL
    ADD AL, DH
    MOV BL, AL
    RET
BCD_TO_BIN ENDP

CODE ENDS
END START
```

Output:



Hardware Programs :

Commands used:

iopm filename.exe 0x40a0 - Grant permission for peripheral devices (address can be 0x40b0 also)

1.Parity of input using logic controller

Read the status of eight input bits from the Logic Controller Interface and display 'FF' if it is even parity bits otherwise display 00. Also display number of 1's in the input data.

Program:

```
PA equ 40A0H  ;Port address of defined to named constant PA. This port address is machine dependent. If 40B0 is
              ;written on the edges of monitor please write PA as 40B0.  If nothing is written then its 40A0
PB equ PA+1    ;PB is next to PA
PC equ PB+1   ;PC is next to PB
PCW equ PC+1   ;PCW is next to PC
CW equ 82h

CODE SEGMENT
      ASSUME CS:CODE
START:
      MOV AL,CW     ;INITIALIZE 8255 with control word
      MOV DX,PCW
      OUT DX,AL

      MOV DX,PB    ;Read the status of port b after setting leds
      IN AL,DX    ; al having port B contents

      MOV CL,0    ;number of 1's counters
      MOV CH,8    ;counter to roatate 8 times
      MOV BL,AL          ;Not necessary
UP1:
      ROL AL,1
      JNC DOWN          ; if bit is 0
      INC CL         ;increment counter of number of 1's if bit is 1
DOWN:
      DEC CH
      JNZ UP1

      MOV CH,CL            ;CH=COUNT
      SHR CL,1           ; if the last bit in cl register is 0, it is even else odd
      JC ODDPARITY
      MOV AL,0FFH    ; even parity
      JMP D1
ODDPARITY:
      MOV AL,00H ;odd parity
D1:   MOV DX,PA
      OUT DX,AL

      MOV AH,01H     ; wait for key press
      INT 21H
      MOV AL,CH      ; display number of 1's
      MOV DX,PA
      OUT DX,AL
      MOV AH,4CH
      INT 21H
CODE ENDS
END START
```

2.BCD up-down counter

Perform the BCD up-down Counter function using the Logic Controller Interface.

Program:

```
PA EQU 40A0H
PB EQU PA+1
PC EQU PB+1
PCW EQU PC+1
CW EQU 82H
N EQU 15H    ;counter maximum limit 15

CODE SEGMENT
        ASSUME CS:CODE
START:
        MOV AL,CW
        MOV DX,PCW
        OUT DX,AL

        MOV AL,00H   ;initial counter value
        MOV DX, PA
UP:
        CMP AL,15H
        JZ DC
        OUT DX,AL
        CALL DELAY
        PUSH AX
        MOV AH, 01     ; check if any key pressed to exit
        INT 16H
        JNZ EXIT   ;if there pressed key in buffer then zero flag is set
        POP AX
        ADD AL,01    ;increment counter
        DAA     ;decimal adjust accumulator after addition
        JMP UP
DC:
        CMP AL,00H
        JZ M1
        OUT DX,AL
        CALL DELAY   ;delay between successive values
        PUSH AX
        MOV AH, 01
        INT 16H
        JNZ EXIT
        POP AX
        SUB AL, 01   ;decrement counter
        DAS      ;decimal adjust after subtraction
        JMP DC
M1:
        OUT DX, AL     ;Display zero
EXIT:
        MOV AH,4CH
        INT 21H

DELAY PROC NEAR
        PUSH AX
        PUSH BX
        PUSH CX
        PUSH DX
```

```
        MOV CX,5FFFH
L3:
        MOV BX,0000H
L2:
        CMP BX,2FFFH
        JZ L1
        INC BX
        JMP L2
L1:
        LOOP L3
        POP DX
        POP CX
        POP BX
        POP AX
        RET
  DELAY ENDP
CODE ENDS
END START
```

3.Ring counter

Perform the Ring Counter function using the Logic Controller Interface.

Program:

```
PA EQU 40A0H
PB EQU PA+1
PC EQU PB+1
PCW EQU PC+1
CW EQU 82H

CODE SEGMENT
        ASSUME CS:CODE
START:
        MOV AL,CW
        MOV DX,PCW
        OUT DX,AL
        MOV AL, 01    ;Set LSB bit as 1
        MOV DX,PA
UP:
        OUT DX,AL
        ROL AL,01    ;shift bit to left
        CALL DELAY
        PUSH AX
        MOV AH, 01    ;check for key press
        INT 16H
        POP AX
        JZ UP
EXIT :
        MOV AH, 4CH
        INT 21H

DELAY PROC NEAR
        PUSH AX
        PUSH BX
        PUSH CX
        PUSH DX
        MOV CX,5FFFH
L3:
```

```
        MOV BX,0000H
L2:
        CMP BX,2FFFH
        JZ L1
        INC BX
        JMP L2
L1:
        LOOP L3
        POP DX
        POP CX
        POP BX
        POP AX
        RET
  DELAY ENDP
CODE ENDS
END START
```

4.Multiplication using logic controller

*Read the status of two 8-bit inputs (X & Y) from the Logic Controller Interface and display X*Y.*

Program:

```
PA EQU 40A0H
PB EQU PA+1
PC EQU PB+1
PCW EQU PC+1
CW EQU 82H
DATA SEGMENT
        X DB ?
        Y DB ?
        PROD DB ?
DATA ENDS

CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
START:
        MOV AX, DATA
        MOV DS, AX
        MOV AL,CW
        MOV DX,PCW
        OUT DX, AL
        MOV DX, PB
        IN AL,DX     ;Read operand-1
        MOV X,AL
        MOV AH,01H  ;Wait for keypress
        INT 21H
        MOV DX,PB
        IN AL,DX    ;Read operand-2
        MOV Y,AL
        MOV AL,X
        MUL Y
        MOV PROD,AL
        MOV DX,PA
        OUT DX, AL    ;display product  (Maximum 8 bits can be displayed)
        MOV AH, 4CH
        INT 21H
CODE ENDS
END START
```

5.Display messages on 7-segment display

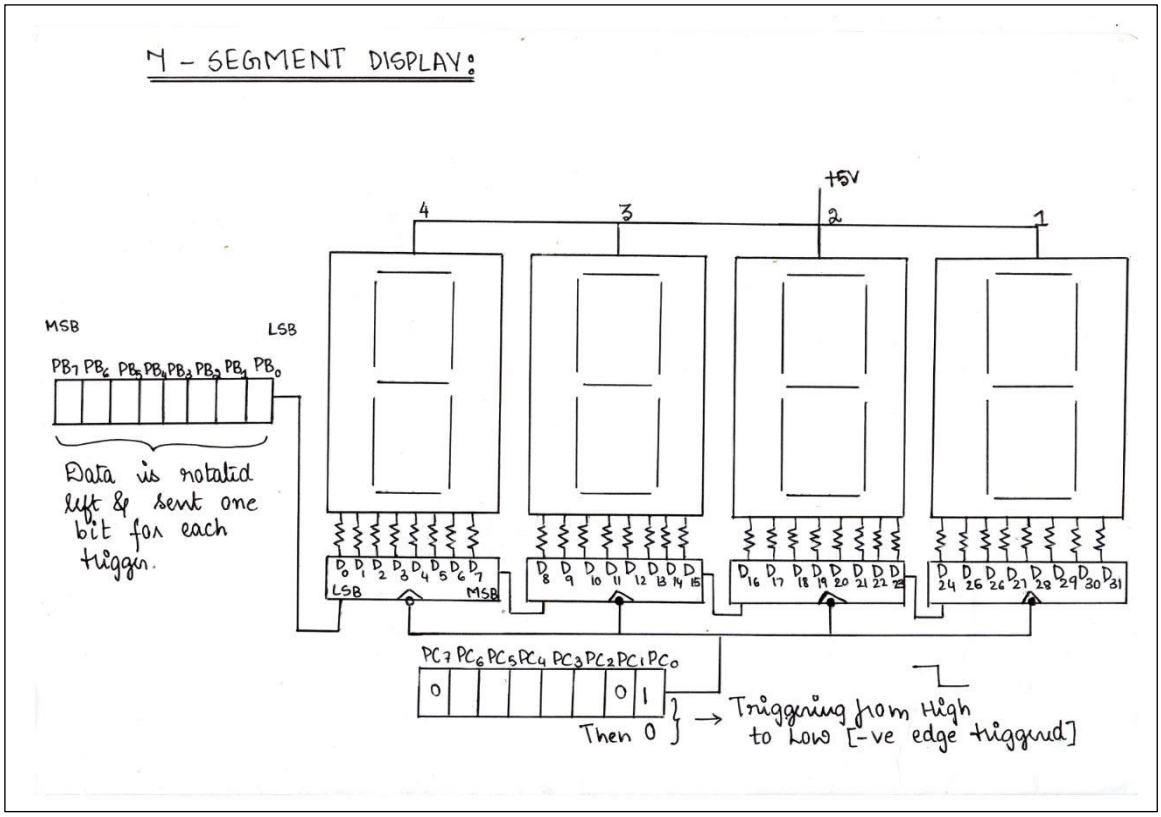
Display messages FIRE and HELP alternately with flickering effects on a 7-segment display interface for a suitable period of time. Ensure a flashing rate that makes it easy to read both the messages.

Program:

7-segment display circuit diagram is given in page no-25

```
PA EQU 40A0H
PB EQU PA+1
PC EQU PB+1
PCW EQU PC+1
CW EQU 80H

DATA SEGMENT
      SSFIRE DB 8EH, 0F9H, 0AFH, 86H   ;7-segment code for FIRE
      SSHELP DB 89H, 86H, 0C7H, 8CH    ;7-segment code for HELP
DATA ENDS

CODE SEGMENT
      ASSUME CS:CODE, DS:DATA
START:
      MOV AX, DATA
      MOV DS, AX
      MOV AL,CW
      MOV DX,PCW
      OUT DX,AL
  UP: LEA SI, SSFIRE
      CALL SSDISPLAY    ;Display FIRE
      CALL DELAY
      LEA SI, SSHELP
      CALL SSDISPLAY   ;Display HELP
      CALL DELAY
      MOV AH, 01H    ;stop flickering when keypress is found
      INT 16H
      JZ UP
      MOV AH, 4CH
      INT 21H

SSDISPLAY PROC NEAR
      MOV CX,04    ;Indicated displaying 4 characters
      MOV DI, SI
      ADD DI, 03  ;Point DI to last character (only for old device). This is not required for new device
UP1:
      MOV AL, [DI]
      MOV BL, 08    ;Sending 8 bits of each character
M1:
      ROL AL, 01H   ;get the last character to be sent first
      MOV DX, PB
      OUT DX, AL
      PUSH AX
      MOV AL, 01H   ;port c high
      MOV DX, PC
      OUT DX, AL
      MOV AL, 00H  ;  port c low. To trigger the port c using negative edge
      OUT DX, AL
```

```
        POP AX
        DEC BL
        JNZ M1
        DEC DI   ;for new device INC DI
        LOOP UP1
        RET
SSDISPLAY ENDP

DELAY PROC NEAR
        PUSH AX
        PUSH BX
        PUSH CX
        PUSH DX
        MOV CX,5FFFH
L3:
        MOV BX,0000H
L2:
        CMP BX,2FFFH
        JZ L1
        INC BX
        JMP L2
L1:
        LOOP L3
        POP DX
        POP CX
        POP BX
        POP AX
        RET
  DELAY ENDP
  CODE ENDS
END START
```



6.Display scrolling messages on 7-segment display

Assume any suitable message of 12 characters length and display it in the rolling fashion on a 7-segment display interface. Ensure a scrolling rate that makes it easy to read whole message.

Program:

```
PA EQU 40A0H
PB EQU PA+1
PC EQU PB+1
PCW EQU PC+1
CW EQU 80H

DATA SEGMENT
        SSCODE DB 0FFH, 0FFH, 0FFH, 0FFH     ;----
                DB 08EH, 0CFH, 0AFH, 86H      ; FIRE
                DB 89H, 86H, 0C7H, 8CH        ;HELP
                DB 0AFH, 0CFH, 0C8H, 90H    ;RING
                DB 0FFH, 0FFH, 0FFH       ;---
DATA ENDS

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

```
START:
        MOV AX, DATA
        MOV DS, AX
        MOV AL, CW
        MOV DX, PCW
        OUT DX, AL
L16:
       MOV CX, 16         ;16 combination of 4 characters each
       LEA SI, SSCODE
UP:
       CALL SSDISPLAY
       CALL DELAY
       INC SI     ;mov to next combination
       LOOP UP
       MOV AH, 01H
       INT 16H
       JZ L16
       MOV AH, 4CH
       INT 21H

SSDISPLAY PROC NEAR
       PUSH CX
       MOV CX,04
       MOV DI, SI
       ADD DI, 03
UP1:
       MOV AL, [DI]
       MOV BL, 08
M1:
       ROL AL, 01H
       MOV DX, PB
       OUT DX, AL
       PUSH AX
       MOV AL, 01H
       MOV DX, PC
       OUT DX, AL
       MOV AL, 00H
       OUT DX, AL
       POP AX
       DEC BL
       JNZ M1
       DEC DI
       LOOP UP1
       POP CX
       RET
SSDISPLAY ENDP

DELAY PROC NEAR
       PUSH AX
       PUSH BX
       PUSH CX
       PUSH DX
       MOV CX,5FFFH
L3:
```

```
        MOV BX,0000H
L2:
        CMP BX,2FFFH
        JZ L1
        INC BX
        JMP L2
L1:
        LOOP L3
        POP DX
        POP CX
        POP BX
        POP AX
        RET
 DELAY ENDP
CODE ENDS
END START
```

7.Display current system time on 7-segment display

Write an assembly language program to display current system time in a seven segment display.

Program:

```
PA EQU 40A0H
PB EQU PA+1
PC EQU PB+1
PCW EQU PC+1
CW EQU 80H

DATA SEGMENT
        SSTABLE DB 0C0H,0F9H,0A4H,0B0H,99H,92H,83H,0F8H,80H,98H    ;code of 7-segment to display 0-9 numbers
        SSCODE DB 00H,00H,00H,00H    ;store time in7-segment format
        HOUR DB ?
        MIN DB ?
        AINDEX DB 4 DUP(?)     ;store system time
DATA ENDS

CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
START:
        MOV AX,DATA
        MOV DS,AX
        MOV AL,CW
        MOV DX,PCW
        OUT DX,AL
        MOV AH,2CH    ;get system time
        INT 21H
        MOV HOUR,CH
        MOV MIN,CL
        LEA SI,AINDEX
        MOV AL,HOUR
        AAM         ;convert packed BCD to unpacked
        MOV [SI],AH
        MOV[SI+1],AL
        MOV AL,MIN
```

```
        AAM
        MOV[SI+2],AH
        MOV[SI+3],AL

        MOV CX,04H
        LEA SI,AINDEX
        LEA DI,SSCODE
        LEA BX,SSTABLE
UP:MOV AL,[SI]
        XLAT       ; convert BCD format time to 7-segment format   (AL← [AL + BX])
        MOV [DI],AL
        INC SI
        INC DI
        LOOP UP

        LEA SI,SSCODE
        CALL SSDISPLAY   ;display time on 7 segment display
        MOV AH,4CH
        INT 21H

SSDISPLAY PROC NEAR
        PUSH CX
        MOV CX,04H
        MOV DI,SI
        ADD DI,03H
UP1:MOV AL,[DI]
        MOV BL,08H
L1:
        ROL AL,01H
        MOV DX,PB
        OUT DX,AL
        PUSH AX
        MOV AL,01H
        MOV DX,PC
        OUT DX,AL
        MOV AL,00H
        MOV DX,PC
        OUT DX,AL
        POP AX
        DEC BL
        JNZ L1
        DEC DI
        LOOP UP1
        POP CX
        RET
SSDISPLAY ENDP

DELAY PROC NEAR
        PUSH AX
        PUSH BX
        PUSH CX
        PUSH DX
        MOV CX,5FFFH
L3:
```

```
        MOV BX,0000H
L2:
        CMP BX,2FFFH
        JZ L1
        INC BX
        JMP L2
L1:
        LOOP L3
        POP DX
        POP CX
        POP BX
        POP AX
        RET
  DELAY ENDP
CODE ENDS
END START
```

8.Drive stepper motor in clockwise direction

Drive a Stepper Motor interface to rotate the motor in clockwise direction by N steps. Introduce suitable delay between successive steps.

Program:

```
PA EQU 40A0H
PB EQU PA+1
PC EQU PB+1
PCW EQU PC+1
CW EQU 80H

DATA SEGMENT
        N DB 100      ;Number of steps.  Each step is 1.8*
DATA ENDS

CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
START:
        MOV AX,DATA
        MOV DS,AX
        MOV AL,CW
        MOV DX,PCW
        OUT DX,AL
        MOV BL, N
        MOV AX,9911H    ;AL=current position (PA0), AH = Next position of stepper motor(PA0 and PA3)
        MOV DX,PA
        OUT DX,AL
        CALL DELAY
UP:
        ROR AL,1    ;rotate clockwise to get next position
        XCHG AL,AH    ;AH was holding next position, so put it in AL
        OUT DX,AL
        CALL DELAY
        ROR AL,1
        XCHG AL,AH
        OUT DX, AL
        CALL DELAY
```

```
        DEC BL
        JNZ UP
EXIT:
        MOV AH,4CH
        INT 21H

DELAY PROC NEAR
        PUSH BX
        MOV CX,3FFFH
THERE:
        MOV BX,0FFFH
HERE:
        DEC BX
        JNZ HERE
        LOOP THERE
        POP BX
        RET
  DELAY ENDP
CODE ENDS
END START
```

9.Drive stepper motor in anticlockwise direction

Drive a Stepper Motor interface to rotate the motor in anticlockwise direction by N steps. Introduce suitable delay between successive steps.

Program:

```
PA EQU 40A0H
PB EQU PA+1
PC EQU PB+1
PCW EQU PC+1
CW EQU 80H

DATA SEGMENT
        N DB 100      ;Number of steps.  Each step is 1.8*
DATA ENDS

CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
START:
        MOV AX,DATA
        MOV DS,AX
        MOV AL,CW
        MOV DX,PCW
        OUT DX,AL
        MOV BL, N
        MOV AX,3311H    ;AL=current position (PA0), AH = Next position of stepper motor(PA0 and PA1)
        MOV DX,PA
        OUT DX,AL
        CALL DELAY
UP:
        ROL AL,1    ;rotate anticlockwise to get next position
        XCHG AL,AH    ;AH was holding next position, so put it in AL
        OUT DX,AL
```

```
        CALL DELAY
        ROL AL,1
        XCHG AL,AH
        OUT DX, AL
        CALL DELAY
        DEC BL
        JNZ UP
EXIT:MOV AH,4CH
        INT 21H

DELAY PROC NEAR
        PUSH BX
        MOV CX,3FFFH
THERE:
        MOV BX,0FFFH
HERE:
        DEC BX
        JNZ HERE
        LOOP THERE
        POP BX
        RET
  DELAY ENDP
CODE ENDS
END START
```

10.Drive stepper motor in anticlockwise & Clockwise direction

Drive a Stepper Motor interface to rotate the motor n steps in anticlockwise direction and then n steps in clockwise direction. Introduce suitable delay between successive steps.

Program:

```
PA EQU 40A0H
PB EQU PA+1
PC EQU PB+1
PCW EQU PC+1
CW EQU 80H

DATA SEGMENT
        N DB 100      ;Number of steps.  Each step is 1.8*
DATA ENDS

CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
START:
        MOV AX,DATA
        MOV DS,AX
        MOV AL,CW
        MOV DX,PCW
        OUT DX,AL
        MOV BL, N
        MOV AX,3311H    ;AL=current position (PA0), AH = Next position of stepper motor(PA0 and PA31)
        MOV DX,PA
        OUT DX,AL
        CALL DELAY
```

```
UP1:ROL AL,1    ;rotate anticlockwise to get next position
      XCHG AL,AH    ;AH was holding next position, so put it in AL
      OUT DX,AL
      CALL DELAY
      ROL AL,1
      XCHG AL,AH
      OUT DX, AL
      CALL DELAY
      DEC BL
      JNZ UP1

      MOV BL, N
      MOV AX,9911H    ;AL=current position (PA0), AH = Next position of stepper motor(PA0 and PA3)
      MOV DX,PA
      OUT DX,AL
      CALL DELAY
UP2:ROR AL,1    ;rotate clockwise to get next position
      XCHG AL,AH    ;AH was holding next position, so put it in AL
      OUT DX,AL
      CALL DELAY
      ROR AL,1
      XCHG AL,AH
      OUT DX, AL
      CALL DELAY
      DEC BL
      JNZ UP2

EXIT:
      MOV AH,4CH
      INT 21H

DELAY PROC NEAR
      PUSH BX
      MOV CX,3FFFH
THERE:
      MOV BX,0FFFH
HERE:
      DEC BX
      JNZ HERE
      LOOP THERE
      POP BX
      RET
  DELAY ENDP
CODE ENDS
END START
```

11.Scanning 3x8 Keypad

Scan 3 x 8 keypad for key closure and to store the code of the key pressed in a memory location or display on screen. Also display row and column numbers of the key pressed.

Program:

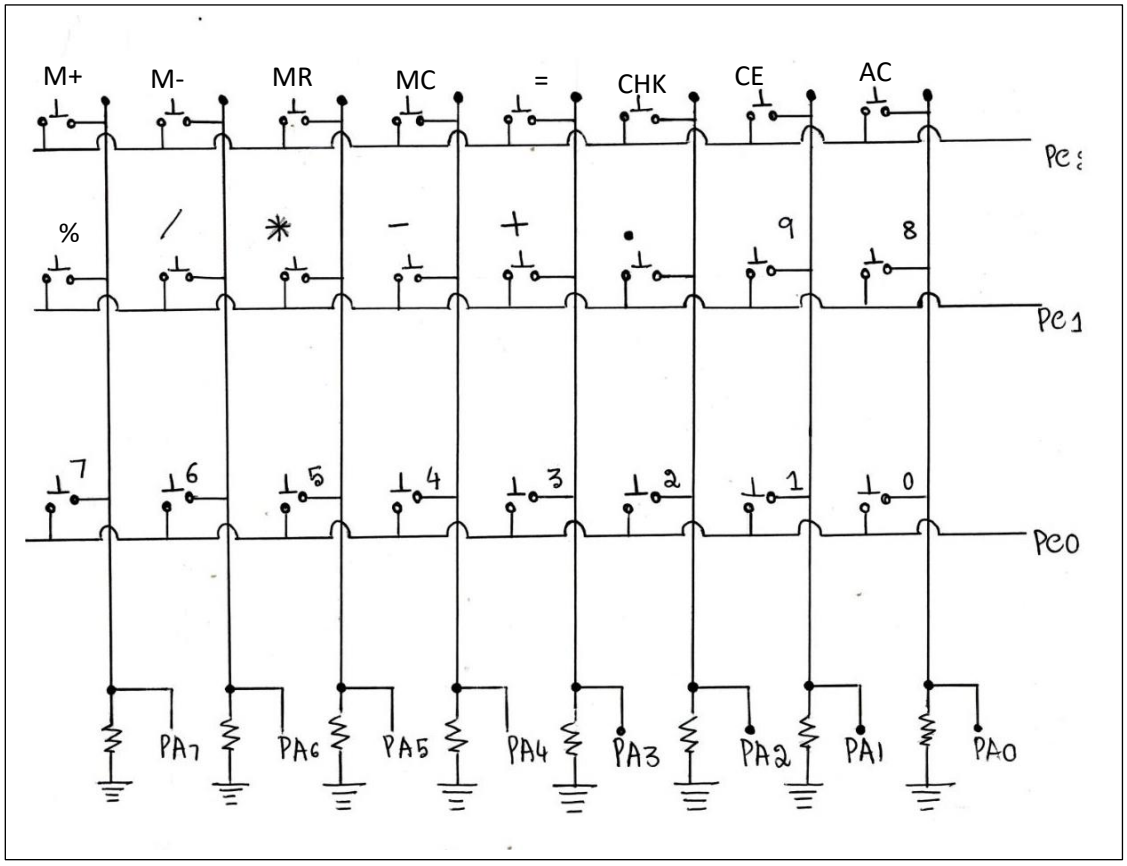
```
PA EQU 40A0H
PB EQU PA+1
```

3x8 Keypad layout is given in page no-35

```
PC EQU PB+1
PCW EQU PC+1
CW EQU 90H

DATA SEGMENT
        ROW DB ?        ;to store row of pressed key
        COL DB ?        ;to store column of pressed key
        VAL DB ?        ;to store value of pressed key
        MSG1 DB 10,13,'ROW : $'
        MSG2 DB 10,13,'COLUMN : $'
        MSG3 DB 10,13,'VALUE : $'
DATA ENDS

CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
START:
        MOV AX,DATA
        MOV DS,AX
        MOV AL,CW
        MOV DX,PCW
        OUT DX,AL

        CALL KEYPRESS       ;Read a key from keypad interface
        MOV ROW,BL
        MOV COL,BH
        MOV VAL,CL

        LEA DX,MSG1     ;display row number
        MOV AH,09H
        INT 21H
        MOV BL,ROW
        CALL DISPHEXA
        LEA DX,MSG2    ;display column number
        MOV AH,09H
        INT 21H
        MOV BL,COL
        CALL DISPHEXA
        LEA DX,MSG3  ;display value of key
        MOV AH,09H
        INT 21H
        MOV BL,VAL
        CALL DISPHEXA
        MOV AH,4CH
        INT 21H

KEYPRESS PROC NEAR
UP:
        MOV AL,01H
        MOV DX,PC
        OUT DX,AL    ;Scan row-1  (PC0 is high)
        MOV DX,PA
        IN AL,DX
        CMP AL,00H     ;check for keypress in row-1
        JNZ FIRSTROW
```

```
        MOV AL,02H
        MOV DX,PC
        OUT DX,AL     ;Scan row-2  (PC1 is high)
        MOV DX,PA
        IN AL,DX
        CMP AL,00H      ;Check for keypress in row-2
        JNZ SECONDROW

        MOV AL,04H
        MOV DX,PC
        OUT DX,AL    ;Scan row-3  (PC2 is high)
        MOV DX,PA
        IN AL,DX
        CMP AL,00H   ;Check for keypress in row-3
        JNZ THIRDROW
        JMP UP   ;scan continuously until key is pressed

FIRSTROW:
        CALL DELAY    ;delay between two key press
        MOV BL,01H
        MOV BH,01H
        MOV CL,00H   ;Initialize value of 1st key in row-1 with 0
UP1:
        ROR AL,01H     ;counting column number
        JC L1
        INC BH
        INC CL
        JMP UP1
SECONDROW:
        CALL DELAY
        MOV BL,02H
        MOV BH,01H
        MOV CL,08H      ;Initialize value of 1st key in row-2 with 08H
UP2:
        ROR AL,01H
        JC L1
        INC BH
        INC CL
        JMP UP2
THIRDROW:
        CALL DELAY
        MOV BL,03H
        MOV BH,01H
        MOV CL,10H    ; Initialize value of 1st key in row-3 with 10H
UP3:
        ROR AL,01H
        JC L1
        INC BH
        INC CL
        JMP UP3
L1:
        RET
KEYPRESS ENDP
```

```
DELAY PROC NEAR
        PUSH CX
        PUSH BX
        MOV CX,0FFFFH
L5:
        MOV BX,04FFHH
L4:
        DEC BX
        JNZ L4
        LOOP L5
        POP BX
        POP CX
        RET
 DELAY ENDP

 DISPHEXA PROC NEAR
        PUSH AX
        MOV DL,BL
        MOV CL,04
        SHR DL, CL
        CMP DL,09
        JBE L10
        ADD DL,07H
L10:
        ADD DL,30H
        MOV AH,02H
        INT 21H
        MOV DL, BL
        AND DL,0FH
        CMP DL,09H
        JBE L11
        ADD DL,07H
L11: ADD DL,30H
        MOV AH,02H
        INT 21H
        POP AX
        RET
 DISPHEXA ENDP
CODE ENDS
END START
```



12.Addition using 3x8 keypad

Scan 3 x 8 keypad for key closure and simulate ADD operation as in a calculator.

Program:

```
PA EQU 40A0H
PB EQU PA+1
PC EQU PB+1
PCW EQU PC+1
CW EQU 90H

DATA SEGMENT
        OP1 DB ?
```

```
        OP2 DB ?
        RES DB ?
        MSG1 DB 10,13,'SUM : $'
DATA ENDS

CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
START:
        MOV AX,DATA
        MOV DS,AX
        MOV AL,CW
        MOV DX,PCW
        OUT DX,AL

        CALL KEYPRESS   ;read 1st digit of operand-1
        MOV OP1,CL
        MOV CL,04
        SHL OP1,CL
        CALL KEYPRESS   ;read 2nd digit of operand-1
        ADD OP1,CL
        CALL KEYPRESS   ;To read plus sign (Reading any key here will not affect addition operation)
        CALL KEYPRESS   ;read 1st  digit of operand-2
        MOV OP2,CL
        MOV CL,04
        SHL OP2,CL
        CALL KEYPRESS    ;read 2nd digit of operand-2
        ADD OP2,CL

        MOV AL,OP1
        ADD AL,OP2    ;compute the sum
        MOV RES,AL

        LEA DX,MSG1
        MOV AH,09H
        INT 21H
        MOV BL,RES
        CALL DISPHEXA  ;display sum
        MOV AH,4CH
        INT 21H

KEYPRESS PROC NEAR
UP:
        MOV AL,01H
        MOV DX,PC
        OUT DX,AL    ;Scan row-1
        MOV DX,PA
        IN AL,DX
        CMP AL,00H     ;check for keypress in row-1
        JNZ FIRSTROW

        MOV AL,02H
        MOV DX,PC
        OUT DX,AL     ;Scan row-2
        MOV DX,PA
```

```
        IN AL,DX
        CMP AL,00H      ;Check for keypress in row-2
        JNZ SECONDROW

        MOV AL,04H
        MOV DX,PC
        OUT DX,AL    ;Scan row-3
        MOV DX,PA
        IN AL,DX
        CMP AL,00H   ;Check for keypress in row-3
        JNZ THIRDROW
        JMP UP   ;scan continuously until key is pressed

FIRSTROW:
        CALL DELAY    ;delay between two key press
        MOV BL,01H
        MOV BH,01H
        MOV CL,00H   ;Initialize value of 1st key in row-1 with 0
UP1:
        ROR AL,01H     ;counting column number
        JC L1
        INC BH
        INC CL
        JMP UP1
SECONDROW:
        CALL DELAY
        MOV BL,02H
        MOV BH,01H
        MOV CL,08H      ;Initialize value of 1st key in row-2 with 08H
UP2:
        ROR AL,01H
        JC L1
        INC BH
        INC CL
        JMP UP2
THIRDROW:
        CALL DELAY
        MOV BL,03H
        MOV BH,01H
        MOV CL,10H    ; Initialize value of 1st key in row-3 with 10H
UP3:
        ROR AL,01H
        JC L1
        INC BH
        INC CL
        JMP UP3
L1:
        RET
KEYPRESS ENDP

DELAY PROC NEAR
        PUSH CX
        PUSH BX
        MOV CX,0FFFFH
```

```
L5:
        MOV BX,04FFHH
L4:
        DEC BX
        JNZ L4
        LOOP L5
        POP BX
        POP CX
        RET
  DELAY ENDP

 DISPHEXA PROC NEAR
        PUSH AX
        MOV DL,BL
        MOV CL,04
        SHR DL, CL
        CMP DL,09
        JBE L10
        ADD DL,07H
L10:
        ADD DL,30H
        MOV AH,02H
        INT 21H
        MOV DL, BL
        AND DL,0FH
        CMP DL,09H
        JBE L11
        ADD DL,07H
L11: ADD DL,30H
        MOV AH,02H
        INT 21H
        POP AX
        RET
  DISPHEXA ENDP
CODE ENDS
END START
```

***