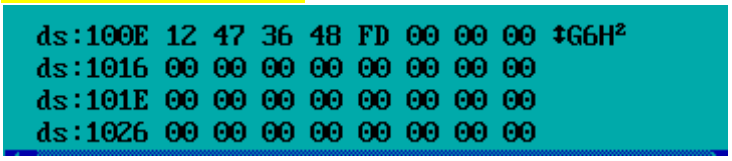
CONTENT TABLE

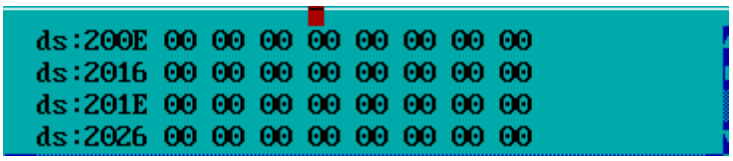
29. Set cursor pointer in middle of a screen and display no from 0-9 (delay).
30. Implement Binary up counter 00 – FF (middle of the screen).
31. Implement Binary down counter FF – 00 (middle of the screen).
32. Implement BCD up counter 00 – 99 (middle of the screen).
33. Implement BCD down counter 99 – 00 (middle of the screen).
34. Display system date (middle of the screen).
35. Display system time (middle of the screen).
36. Convert hexadecimal number to BCD number.
37. Convert BCD number to hexadecimal number.
38. Program to Add two ASCII numbers.
39. Program to subtract two ASCII numbers
40. Program to multiply two ASCII numbers
41. Program to divide two ASCII numbers
42. Program to Read a Character With ECHO.
43. Program to Read a Character Without ECHO
44. Program to Transfer A String from Source to Destination
45. Program to Find Whether A Given String Is A Palindrome or Not
46. Program to reverse a given string
47. Program to search a character in a given string

1. Copy a block of byte data from one location to another.

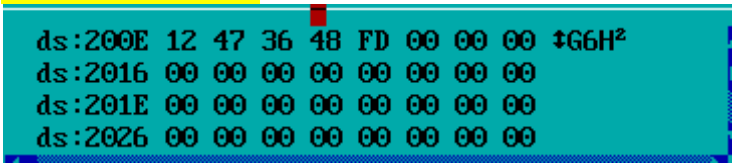
```
        .MODEL SMALL
        .STACK

  ORG 100H
      .DATA
      ORG  1000H
      SRC DB 12H, 47H, 54, 48H, 0FDH
      ORG 2000H
      DST DB 5 DUP(0)
.CODE
START:
      MOV AX, @DATA
      MOV DS, AX
      LEA   SI,   SRC
      LEA   DI,   DST
      MOV CL, 05H
L1:     MOV AL,[ SI]
      MOV [DI], AL
      INC SI
      INC DI
      DEC CL
      JNZ L1

      MOV AH, 4CH
      INT 21H
      END START
```

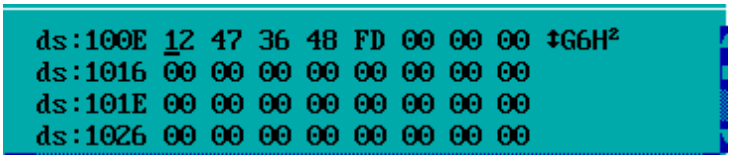
BEFORE EXECUTION :





AFTER EXECUTION :

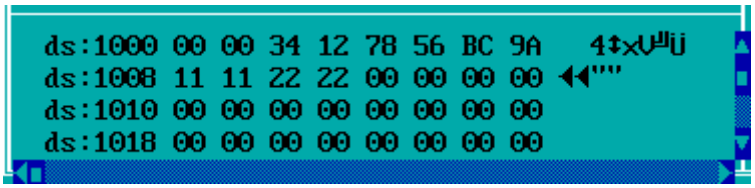


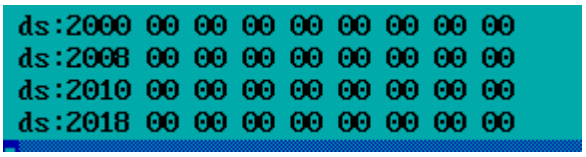


2. Copy a block of word data from one location to another.

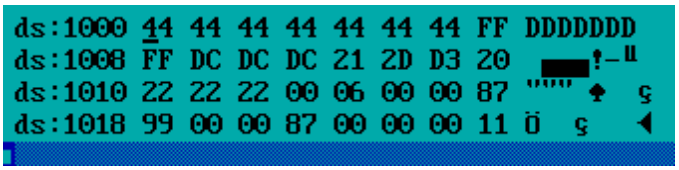
```
.MODEL SMALL
.STACK
ORG 100H
.DATA
ORG 1000H
SRC DW 1234H, 5678H, 9ABCH, 1111H, 2222H
ORG 2000H
DST DW 5 DUP(0)
.CODE
START:
MOV AX, @DATA
MOV DS, AX

LEA SI, SRC
LEA DI, DST
MOV CL, 05H
L1:
MOV AX, [SI]
MOV [DI]
ADD SI, 02
ADD DI, 02

DEC CL
JNZ L1

MOV AH, 4CH
INT 21H
END START
```

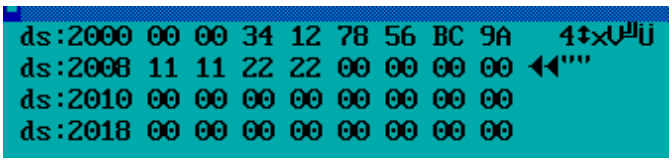
BEFORE EXECUTION





AFTER EXECUTION





3. **Exchange block of byte data.**

```
.MODEL SMALL
.STACK
        ORG 100H
.DATA
        ORG 1000H
        SRC DB 10H,44H,0ABH,50H,6DH
        ORG 2000H
        DST DB 99H,87H,0FFH,19H,29H
.CODE
START:
        MOV AX,@DATA
        MOV DS,AX

        LEA  SI,SRC
        LEA DI,DST
        MOV CL,05

BACK: MOV AL,[SI]
        XCHG AL,[DI]
        MOV [SI],AL
        INC SI
        INC DI
        DEC CL
        JNZ BACK

        MOV AH,4CH
        INT 21H
        END START
```

BEFORE EXECUTION:

```
ds:1000 10 44 AB 50 6D 00 00 00 ►D½Pm
ds:1008 00 00 00 00 00 00 00 00
ds:1010 00 00 00 00 00 00 00 00
ds:1018 00 00 00 00 00 00 00 00
```

```
ds:2000 99 87 FF 19 29 00 00 00 Öç ↓)
ds:2008 00 00 00 00 00 00 00 00
ds:2010 00 00 00 00 00 00 00 00
ds:2018 00 00 00 00 00 00 00 00
```

AFTER EXECUTION:

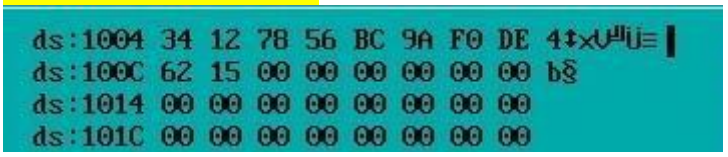
```
ds:1000 99 87 FF 19 29 00 00 00 Öç ↓)
ds:1008 00 00 00 00 00 00 00 00
ds:1010 00 00 00 00 00 00 00 00
ds:1018 00 00 00 00 00 00 00 00
```

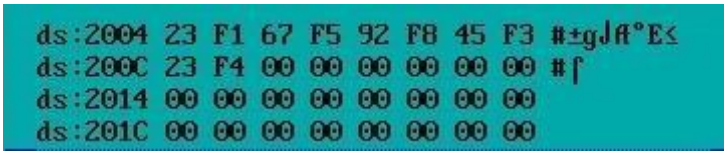
```
ds:2000 10 44 AB 50 6D 00 00 00 ►D½Pm
ds:2008 00 00 00 00 00 00 00 00
ds:2010 00 00 00 00 00 00 00 00
ds:2018 00 00 00 00 00 00 00 00
```

4. Exchange block of word data.

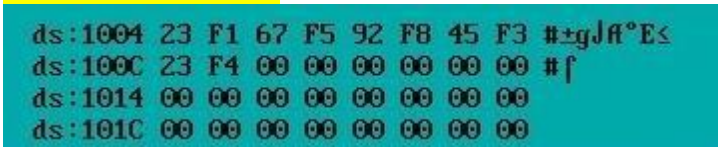
```
.MODEL SMALL
.STACK
        ORG 100H
.DATA
        ORG 1000H
        SRC DW 1234H,5678H,9ABCH,0EF0H,1562H
        ORG 2000H
        DST DW 0F123H,0F567H,0F892H,0F345H,0F423H
.CODE
START:
        MOV AX,@DATA
        MOV DS,AX

        LEA    SI,SRC
        LEA   DI,DST
        MOV CL,05H

BACK:
        MOV AX,[SI]
        XCHG AX,[DI]
        MOV [SI],AX
        ADD SI,02
        ADD DI,02
        DEC CL
        JNZ BACK

        MOV AH,4CH
        INT 21H
        END START
```

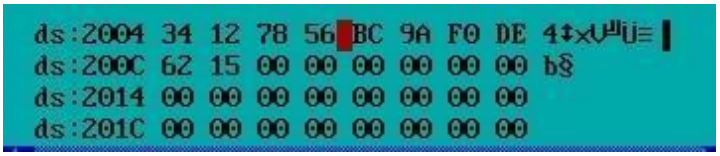
BEFORE EXECUTION:





AFTER EXECUTION:





5. To add two data bytes.

```
.MODEL SMALL
.STACK
        ORG 100H
.DATA
        X DB 48H
        Y DB 20H
        Z DW 0000H
.CODE
START:

        MOV AX,@DATA
        MOV DS,AX

        MOV AH,00
        MOV AL,X
        MOV BH,00
        MOV BL,Y
        ADD AX,BX
        MOV Z,AX

        MOV AH,4CH
        INT 21H
        END START
```

BEFORE EXECUTION:

```
ds:000A E6 01 AF 15 B0 01 AF 15
ds:0012 83 02 13 10 92 01 01 01
ds:001A 01 00 02 FF FF FF FF FF
ds:0022 FF FF FF FF FF FF FF FF
```

AFTER EXECUTION:

```
ds:000C 68 00 00 00 00 00 00 00 h
ds:0014 00 00 00 00 00 00 00 00
ds:001C 00 00 00 00 00 00 00 00
ds:0024 00 00 00 00 00 00 00 00
```

6. **<u>To add two data words.</u>**

```
.MODEL SMALL
.STACK
        ORG 100H
.DATA
        ORG 1000H
        B DW 0FFFFH
        ORG 2000H
        C DW 1FFFH
        ORG 3000H
        SUM DW 0000H
        ORG 4000H
        D DW 0000H
.CODE
START:
        MOV AX,@DATA
        MOV DS,AX

        MOV AX,[B]
        ADD AX,[C]
        MOV SUM,AX
        ADC D,0000

        MOV AH,4CH
        INT 21H
        END START
```

BEFORE EXECUTION

```
ds:1008 FF FF 00 00 00 00 00 00
ds:1010 00 00 00 00 00 00 00 00
ds:1018 00 00 00 00 00 00 00 00
ds:1020 00 00 00 00 00 00 00 00
```

```
ds:2008 FF 1F 00 00 00 00 00 00
ds:2010 00 00 00 00 00 00 00 00
ds:2018 00 00 00 00 00 00 00 00
ds:2020 00 00 00 00 00 00 00 00
```

AFTER EXECUTION

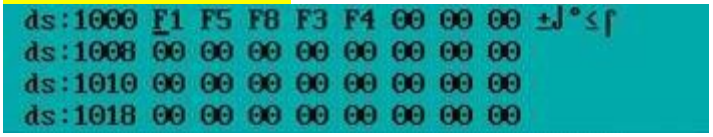
```
ds:3008 FE 1F 00 00 00 00 00 00
ds:3010 00 00 00 00 00 00 00 00
ds:3018 00 00 00 00 00 00 00 00
ds:3020 00 00 00 00 00 00 00 00
```

```
ds:4008 01 00 00 00 00 00 00 00
ds:4010 00 00 00 00 00 00 00 00
ds:4018 00 00 00 00 00 00 00 00
ds:4020 00 00 00 00 00 00 00 00
```

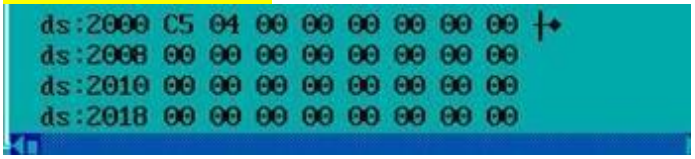
7. **Find sum of a byte array.**

```
  .MODEL SMALL
.STACK
        ORG 100H
.DATA
        ORG 1000H
        NOS DB 0F1H, 0F5H, 0F8H, 0F3H, 0F4H
        ORG 2000H
        SUM DW 0000H

.CODE
START: MOV AX, @DATA
        MOV DS, AX
        LEA SI, NOS
        MOV CL, 05
        MOV AX, 0000H
        MOV BH, 00H

L1:     MOV BL,[SI]
        ADD AX, BX
        INC SI
        DEC CL
        JNZ L1
        MOV SUM, AX

        MOV AH, 4CH
        INT 21H
        END START
```

BEFORE EXECUTION



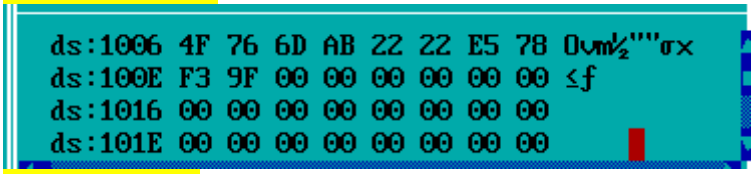
AFTER EXECUTION



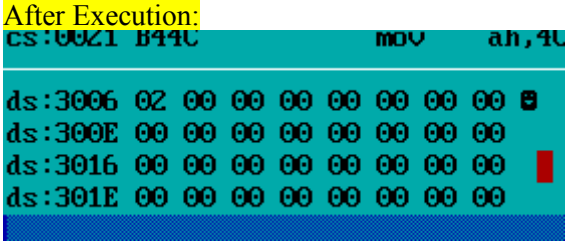
8. Find the sum of a word array.

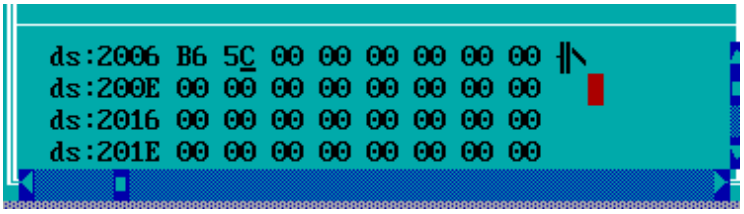
```
.MODEL SMALL
.STACK
        ORG 100H
.DATA
        ORG 1000H
        ARRY DW 764FH, 0AB6DH, 2222H, 78E5H, 9FF3H
        ORG 2000H
        SUM DW 0000H
        ORG 3000H
        D DW 0000H
.CODE
START:
        MOV AX, @DATA
        MOV DS, AX
        LEA SI, ARRY
        MOV CL, 05
        MOV AX, 0000H

  BACK: MOV BX, [SI]
        ADD AX, BX
        ADC D, 0000H
        ADD SI, 02
        DEC CL
        JNZ BACK
        MOV SUM, AX

        MOV AH, 4CH
        INT 21H
        END START
```

Before Execution:



After Execution:





9. Display string using interrupt.

```
.MODEL SMALL
.STACK 100H
.DATA
        STRING DB 'MANIPAL ACADEMY OF HIGHER EDUCATION$'
 .CODE
 START :
        MAIN PROC FAR
        MOV AX,@DATA
        MOV DS,AX

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

        MOV AH,4CH
        INT 21H
        MAIN ENDP
        END START
```

BEFORE EXECUTION:

```
ds:0110 21 00 4D 41 4E 49 50 41 ! MANIPA
ds:0118 4C 20 41 43 41 44 45 4D L ACADEM
ds:0120 59 20 4F 46 20 48 49 47 Y OF HIG
ds:0128 48 45 52 20 45 44 55 43 HER EDUC
ds:0130 41 54 49 4F 4E 24 00 00 ATION$
```

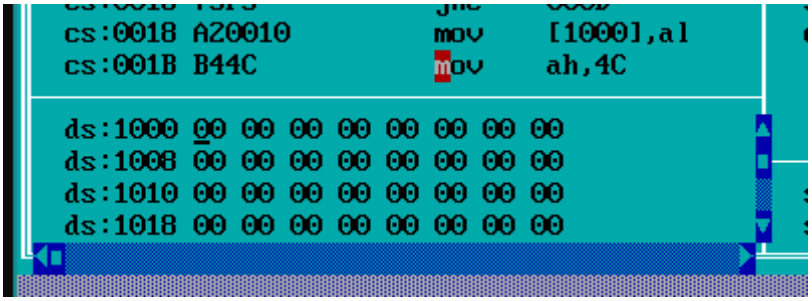
AFTER EXECUTION:

```
X:\>INTPGM
MANIPAL ACADEMY OF HIGHER EDUCATION
X:\>_
```

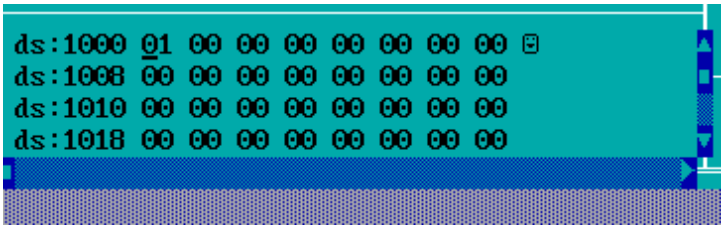
10. **Find the smallest number in an array.**

```
.MODEL SMALL
.STACK
        ORG 100H
.DATA
        ORG 1000H
        AR DB 12H, 0DFH, 56H, 0ABH, 67H
        ORG 2000H
        X DB 0FFH
.CODE
START:
        MOV AX, @DATA
        MOV DS, AX

        LEA SI, AR
        MOV CL, 05H

BACK: MOV AL, [SI]
        CMP AL, X
        JNC NEXTNO
        MOV X, AL

NEXTNO: INC SI
        DEC CL
        JNZ BACK

        MOV AH, 4CH
        INT 21H
        END START
```

BEFORE EXECUTION:



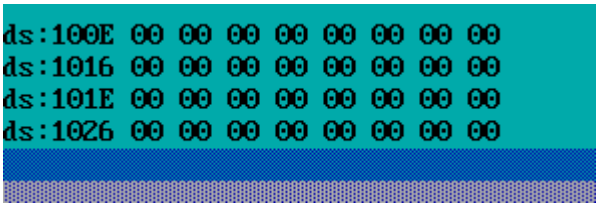
AFTER EXECUTION:



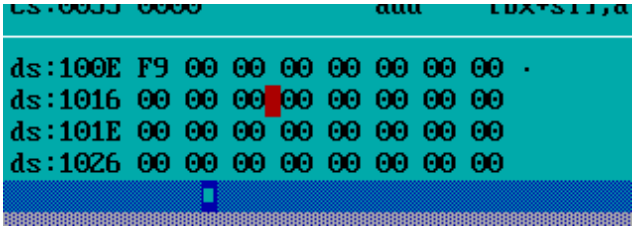
11. <u>Find the largest number in an array.</u>

```
.MODEL SMALL
.STACK
      ORG 100H
.DATA
      ORG 1000H
      AR DB 12H, 0DFH, 56H, 0ABH, 67H
      ORG 2000H
      X DB 00H
.CODE
START:
      MOV AX, @DATA
      MOV DS, AX

      LEA SI, AR
      MOV CL, 05H

BACK: MOV AL, [SI]
       CMP AL, X
      JC NEXTNO
       MOV X, AL

NEXTNO: INC SI
      DEC CL
      JNZ BACK

      MOV AH, 4CH
      INT 21H
      END START
```

BEFORE EXECUTION:



AFTER EXECUTION:



12. Display if the given number is even or odd.

```
.MODEL SMALL
.STACK
    ORG 100H
.DATA
    ORG 1000H
    X DB 45H
    ORG 2000H
    EVNMSG DB "THE NUMBER IS EVEN$"
    ODDMSG DB "THE NUMBER IS ODD$"
.CODE
START:MOV AX,@DATA
    MOV DS, AX

    MOV AL,X
    AND AL,01H
    JNZ XYZ ; JUMP TO XYZ IF Z=0

    MOV AH,09H
    LEA DX,ODDMSG
    INT 21H
    JMP PRGMEND
XYZ: MOV AH,09H
    LEA DX,ODDMSG
    INT 21H
PRGMEND:END START
```

BEFORE EXECUTION:

```
ds:1000 45 00 00 00 00 00 00 00 E
ds:1008 00 00 00 00 00 00 00 00
ds:1010 00 00 00 00 00 00 00 00
ds:1018 00 00 00 00 00 00 00 00
```

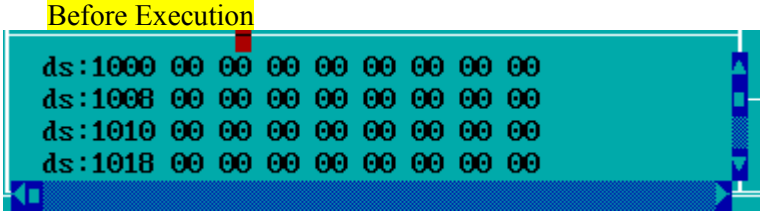
AFTER EXECUTION:

```
Turbo Debugger Version 3.1 Copyright (c) 1988,92 Borland International
THE NUMBER IS ODD
X:\>
```

13. Display if the given number is positive or negative.

```
.MODEL SMALL

.STACK
      ORG 100H

.DATA
      ORG 1000H
      X DB 48H
      ORG 2000H
      MSG1 DB 'POSITIVE NUMBER$'
      MSG2 DB 'NEGATIVE NUMBER$'

.CODE
START : MOV AX,@DATA
      MOV DS,AX

      MOV AL,X

      AND AL,80H
      JNZ POS1

      MOV AH,09H
      LEA DX,MSG1
      INT 21H
      JMP LAST

POS1 : MOV AH,09H
      LEA DX,MSG2
      INT 21H
      JMP LAST


LAST : MOV AH,4CH
      INT 21H
      END START
```

Before Execution



After Execution

14. In array find number of positive numbers and negative numbers.

```
model SMALL
.STACK
        ORG 100H
.data
        ORG 1000H
        X DB 82H , 0F1H, 81H, 18H, 88H, 12H
        ORG 1008H
        POS DB 00H;
        ORG 1010H
        NEG DB 00H;
.CODE
START:
        MOV AX, @DATA
        MOV DS, AX

        LEA SI, X
        MOV CL, 06H
        MOV BH, 00H ;POS
        MOV BL, 00H ;NEG

BACK: MOV AL, [SI]
        AND AL, 80H
        JNZ L1
        INC BH
        JMP L2
L1:     INC BL
L2:     INC SI
        DEC CL
        JNZ BACK

        MOV POS, BH ;ds:1016
        MOV NEG, BL ;ds:101E
        MOV AH, 4CH
        INT       21H
        END START
```

BEFORE EXECUTION:

```
ds:100E 82 F1 81 18 88 12 00 00
ds:1016 00 00 00 00 00 00 00 00
ds:101E 00 00 00 00 00 00 00 00
ds:1026 00 00 00 00 00 00 00 00
```

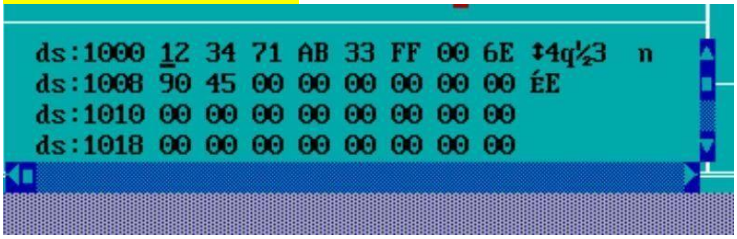
AFTER EXECUTION:

```
48AF:100E 82 F1 81 18 88 12 00 00
48AF:1016 02 00 00 00 00 00 00 00
48AF:101E 04 00 00 00 00 00 00 00
48AF:1026 00 00 00 00 00 00 00 00
```

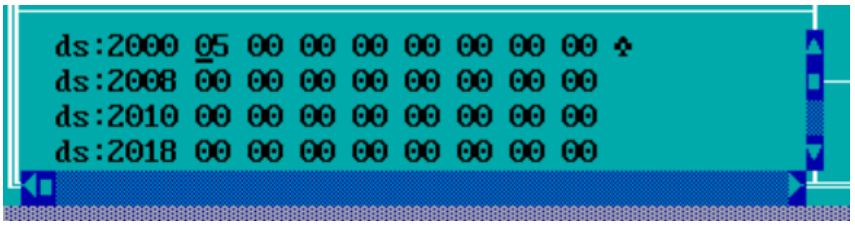
15. In array find number of even numbers and odd numbers.

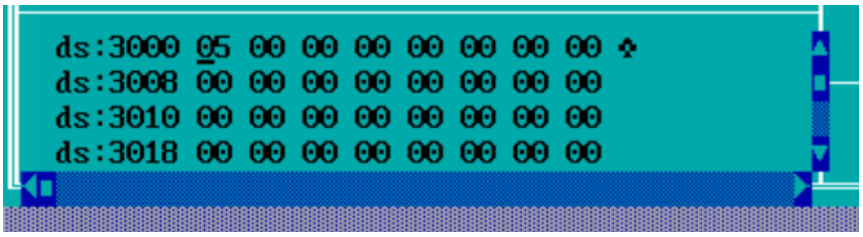
```
.MODEL SMALL
.STACK
    ORG 100H
.DATA
    ORG 1000H
    X DB 12H, 34H, 71H, 0ABH, 33H, 0FFH, 00H, 6EH, 90H, 45H
    ORG 2000H
    EVE DB 00H
    ORG 3000H
    ODD DB 00H
.CODE
START:
    MOV AX,@DATA
    MOV DS, AX

    MOV CL,0AH
    LEA SI, X
    MOV BL, 00H
    MOV DL, 00H

L1: MOV AL, [SI]
    AND AL,01H
    JNZ ODDSKIP
    INC BL
    JMP ENDSKIP

ODDSKIP:INC DL
    JMP ENDSKIP

ENDSKIP:INC SI
    DEC CL
    JNZ L1

    MOV EVE, BL
    MOV ODD, DL

    MOV AH,4CH
    INT 21H
    END START
```

BEFORE EXECUTION:



AFTER EXECUTION:

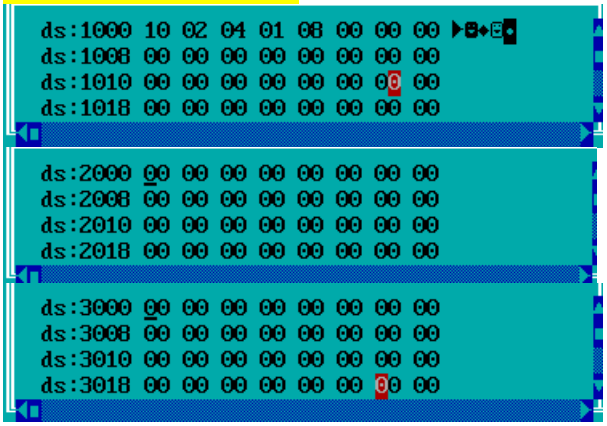




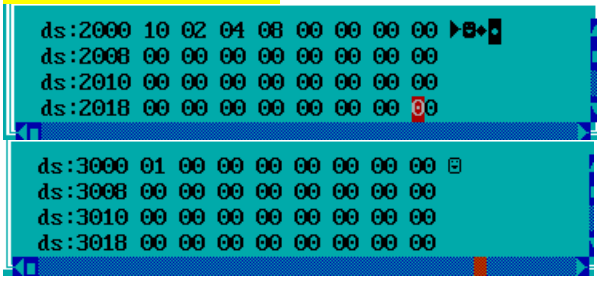
16. Program to separate even numbers and odd numbers in an array.

```
.MODEL SMALL
.STACK
 ORG 100H
.DATA
 ORG 1000H
 ARR DB 10H, 02H, 04H, 01H, 08H;
 ORG 2000H
 EVE DB 00H
 ORG 3000H
 ODD DB 00H
.CODE
START :
 MOV AX, @DATA
 MOV DS, AX

 LEA BX, ARR
 LEA SI, EVE
 LEA DI, ODD
 MOV CL, 05H

L1 : MOV AL, [BX]
 AND AL, 01H
 JNZ L2
 MOV AL, [BX]
 MOV [SI], AL
 INC SI
 JMP L3
L2 : MOV AL, [BX]
 MOV [DI], AL
 INC  DI
L3: INC BX
DEC CL
JNZ L1

 MOV AH, 4CH
 INT 21H
 END START
```

BEFORE EXECUTION:



AFTER EXECUTION:



17. Program to separate positive numbers and negative numbers in an array.

```
.MODEL SMALL
.STACK
ORG 100H
.DATA
ORG 1000H
X DB 10H, 08H, 05H, 02H, 22H
ORG 2000H
POSX DB 5 DUP(0)
ORG 3000H
NEGX DB 5 DUP(0)
.CODE
START :
MOV AX, @DATA
MOV DS, AX
LEA BX,X
LEA SI, POSX
LEA DI, NEGX
MOV CL, 05H
L1 : MOV AL, [BX]
AND AL, 80H
JNZ XYZ ; JUMP TO XYZ IF Z=0 OR ODD NUMBER
MOV AL, [BX]
MOV [SI], AL
INC SI
JMP PND
XYZ : MOV AL, [BX]
MOV [DI], AL
INC DI
PND : INC BX
DEC CL
JNZ L1
MOV AH, 4CH
INT 21H
END START
```

Before Execution

```
ds:1000 10 08 05 02 22 00 00 00
ds:1008 00 00 00 00 00 00 00 00
ds:1010 00 00 00 00 00 00 00 00
ds:1018 00 00 00 00 00 00 00 00
```

After Execution

```
48B0:2000 10 08 05 02 22 00 00 00
48B0:2008 00 00 00 00 00 00 00 00
48B0:2010 00 00 00 00 00 00 00 00
48B0:2018 00 00 00 00 00 00 00 00
```

```
ds:3000 00 00 00 00 00 00 00 00
ds:3008 00 00 00 00 00 00 00 00
ds:3010 00 00 00 00 00 00 00 00
ds:3018 00 00 00 00 00 00 00 00
```

18. Print a character or symbol using interrupt.

```
.MODEL SMALL
.STACK
        ORG 100H
.DATA
        ORG 1000H
        X DB 00
.CODE
START:
        MOV AX,@DATA
        MOV DS,AX

        MOV AH,01
        INT 21H

        MOV AH,02
        INC AL
        MOV DL,AL
        INT 21H

        MOV AH,4CH
        INT 21H
        END START
```

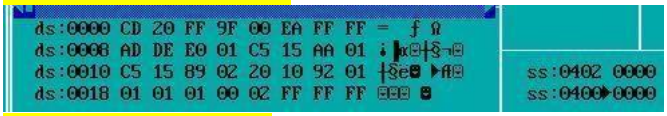
Output

```
X:\>EDIT INTRUPRI.ASM

X:\>INTRUPRI
HI
```

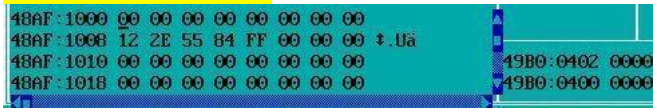
19. <u>Program to sort an array in ascending order.</u>

```
.MODEL SMALL
.STACK
        ORG 100H
.DATA
        ORG 1000H
        ARY DB 12H,55H,84H,0FFH,2EH
.CODE
START:
        MOV AX,@DATA
        MOV DS,AX

        MOV CL,04H
        MOV BL,CL

L2:     LEA    SI,ARY
        MOV CL,BL
L1:     MOV    AL,[SI]
        CMP AL,[SI+1]
        JC SKIP
        XCHG AL,[SI+1]
        MOV [SI],AL

SKIP: INC SI
        DEC CL
        JNZ L1
        DEC BL
        JNZ L2

        MOV AH,4CH
        INT 21H
        END START
```

BEFORE EXECUTION



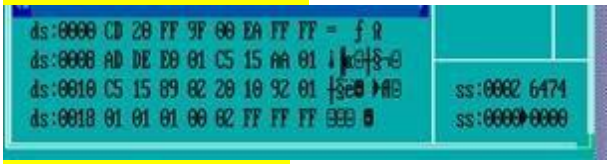
AFTER EXECUTION



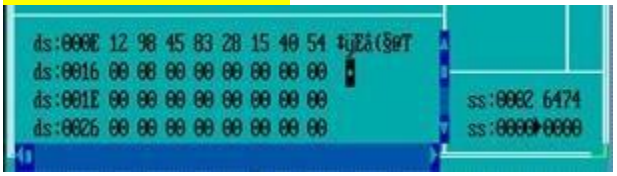
20. **Program to sort an array in descending order.**

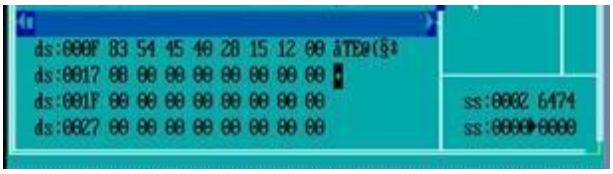
```
  .MODEL SMALL
 .DATA
        ARRAY DB 12H, 98H, 45H, 83H, 28H,15H,40H,54H
COUNT DW 8
.CODE
        MOV AX, @DATA
        MOV DS, AX
        MOV CX,COUNT
        DEC CX
SCAN:
        MOV BX,CX
        MOV SI,0
COMP:
        MOV AL,ARY[SI]
        MOV DL,ARY[SI+1]
        CMP AL,DL
        JNC NOSWAP
        MOV ARY[SI],DL
        MOV ARY[SI+1],AL
NOSWAP:
        INC SI
        DEC BX
        JNZ COMP
LOOP SCAN
MOV AH, 4CH
INT 21H
END
```

BEFORE EXECUTION:-



AFTER EXECUTION:-





21. <u>Program to reverse a binary number.</u>

```
 .MODEL SMALL
.STACK
        ORG 100H
.DATA
        ORG 1000H
        X DB 75H;0111 0101
         Y DB 00H
.CODE
START:
        MOV AX,@DATA
        MOV DS,AX

        MOV CL,08H
        MOV BL,00H
        MOV AL,X ; 75H=AL

BACK: ROR AL,01H
        RCL BL,01H
        DEC CL ;0
        JNZ BACK ; ZF=0

        MOV Y,BL ;1010 1110 AE

        MOV AH,4CH
        INT 21H
        END START
```

BEFORE EXECUTION

```
ds:100C 75 00 00 00 00 00 00 00 u
ds:1014 00 00 00 00 00 00 00 00
ds:101C 00 00 00 00 01 FF AB 84
ds:1024 13 00 00 00 00 00 00 00 !!
```

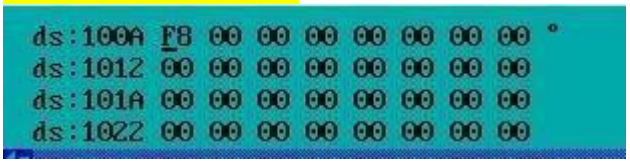
AFTER EXECUTION

```
ds:100C 75 AE 00 00 00 00 0
ds:1014 00 00 00 00 00 00 0
ds:101C 00 00 00 00 01 FF A
ds:1024 13 00 00 00 00 00 0
```

22. Count the number of 0's and 1's in a number.

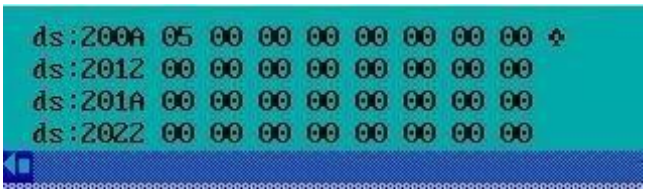
```
.MODEL SMALL
.STACK
      ORG 100H
.DATA
      ORG 1000H
      X DB 0F8H
      ORG 2000H
      Y DB 00H
      ORG 3000H
      Z DB 00H
.CODE
START:
      MOV AX,@DATA
      MOV DS,AX

      MOV  CL,08H
      MOV  BH,00H
      MOV  BL,00H
      MOV AL,X


BACK: ROR AL,01
      JC ADDBH
      INC BL
      JMP SKIP

ADDBH: INC BH

SKIP: DEC CL
      JNZ BACK

      MOV Y,BH
      MOV Z,BL

      MOV  AH,4CH
      INT 21H
      END START
```

BEFORE EXECUTION:

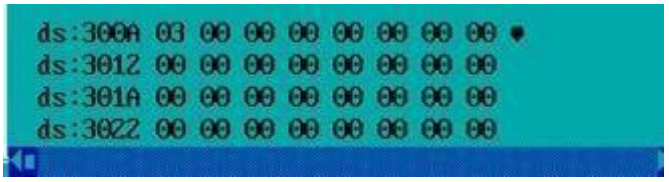


AFTER EXECUTION:

COUNT OF 1'S



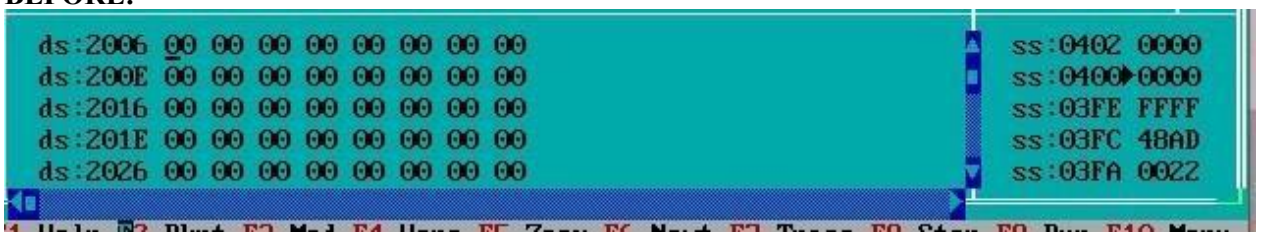
COUNT OF 0'S



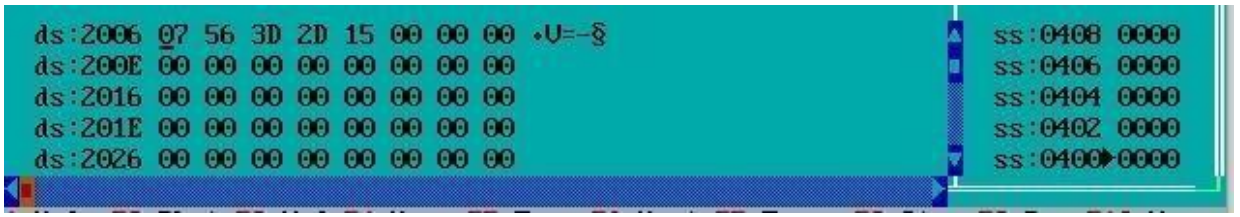
23. <u>Add the correspond element of 2 arrays and save result in a sum of array.</u>

```
.MODEL SMALL
.STACK
  ORG 100H
.DATA

     ORG 1000H
      X DB 01H,02H,09H,07H,05H
      Y DB 06H,54H,34H,26H,10H

      ORG 2000H
      SUM DB 5 DUP (0)

.CODE
START:
     MOV AX,@DATA
     MOV DS,AX
     MOV CL,05H

      LEA SI,X
      LEA DI,Y
      LEA BX,SUM
      MOV AL,00H


BACK:    MOV AL,[SI]
      ADD AL,[DI]
      MOV [BX],AL
      INC SI
      INC DI
      INC BX
      DEC CL
      JNZ BACK
      MOV AH,4CH
      INT 21H
      END START
```

BEFORE:



AFTER:



24. Check if the given number belongs to 2 out of 5 code.

```
.MODEL SMALL
.STACK
      ORG 100H
.DATA
      ORG 1000H
      X DB 82H
      YMSG DB "YES IT IS 2 OUT OF 5 CODE$"
      NMSG DB "NO IT IS NOT 2 OUT OF 5 CODE$"

.CODE
START: MOV AX,@DATA
     MOV DS, AX

     MOV CL, 05
     MOV CH,  00
     MOV AL, X
     MOV BL, AL

      AND AL, 0E0H
      JNZ  NOTCODE

      MOV AL, BL

BACK: ROR AL, 01
     JNC SKIP
     INC CH

 SKIP: DEC CL
      JNZ BACK

     CMP CH, 02H
      JNZ NOTCODE

     MOV AH, 09
     LEA DX, YMSG
     INT 21H
     JMP ENDX

NOTCODE: MOV AH, 09H
       LEA DX, NMSG
       INT 21H

ENDX: MOV AH,4CH
       INT 21H
       END START
```

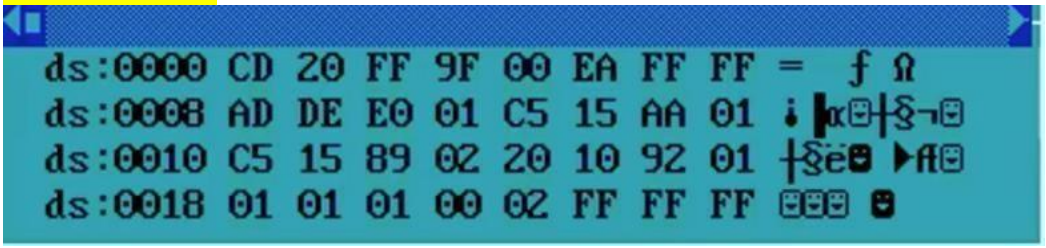
OUTPUT:

```
X:\>\
edit 2outof5.asm\
\
edit 2outof5.asm

X:\>masm 2outof5.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51698 + 464846 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

X:\>loadfix link 2outof5.obj;

Microsoft (R) Overlay Linker  Version 3.64
Copyright (C) Microsoft Corp 1983-1988.  All rights reserved.


X:\>2outof5
NO IT IS NOT 2 OUT OF 5 CODE
X:\>_
```

25. Program to divide a 32-bit number by 16-bit number.

```
.MODEL SMALL
.STACK
        ORG 100H
.DATA
        HWN DW 1724H
        LWN DW 2728H
        DNR DW 4428H
        R DW 0000H
        Q DW 0000H
.CODE
START:
        MOV AX,@DATA
        MOV DS,AX

        MOV DX,HWN
        MOV AX,LWN
        MOV CX,DNR
        DIV CX
        MOV R,DX
        MOV Q,AX

        MOV AH,4CH
        INT 21H
        END START
```

Before Execution:



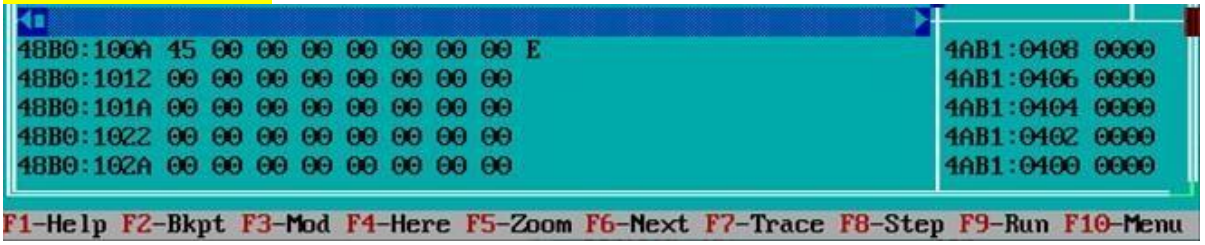
After Execution:

26. Search for a key element in an array.

```
.MODEL SMALL
.STACK
    ORG 100H
.DATA
    ORG 1000H
    X DB 11H,12H,18H,4FH,98H,0EDH,0E9H,88H,75H,0ABH
    KEY DB 45H
    ORG 2000H
    FND  DB 00H
    FLOC DW 0000H

.CODE
START:
     MOV AX, @DATA
     MOV DS, AX

     LEA SI, X
     DEC SI
     MOV AL, KEY
     MOV CX, 0AH

BACK: INC SI
     CMP AL, [SI]
     LOOPNZ BACK;

     CMP CX, 00H
     JZ NOTFND
     MOV FND, 0FFH
     MOV FLOC, SI
     JMP END1

NOTFND: MOV FND, 55H

END1: MOV AH, 4CH
      INT 21H
      END START
```

BEFORE EXECUTION

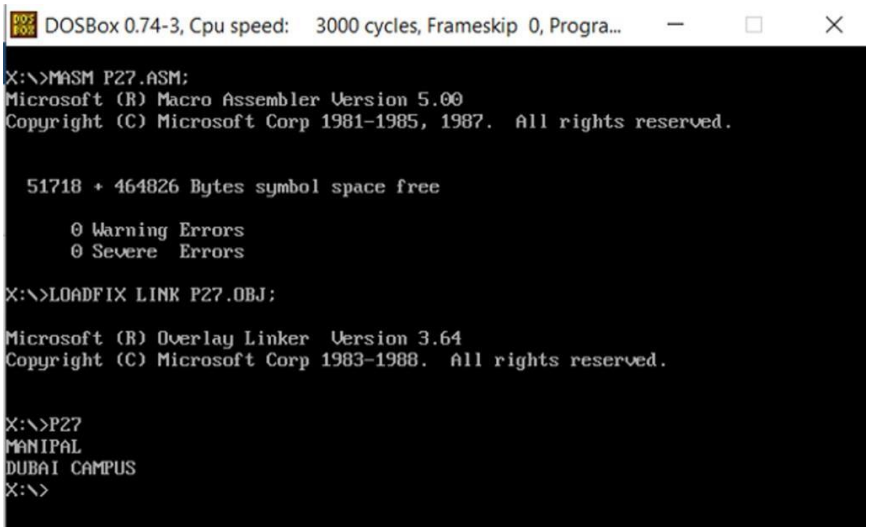
AFTER EXECUTION



27. Print msg using macro.

```
.MODEL SMALL
.STACK
   ORG 100H
.DATA
   MSG1 DB"MANIPAL $"
   MSG0 DB 10,13,"$"
   MSG2 DB "DUBAI CAMPUS $"

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

ENDM
.CODE
START:
     MOV AX,@DATA
     MOV DS,AX
     PRINTF MSG1
     PRINTF MSG0
     PRINTF MSG2

     MOV AH,4CH
     INT 21H
     END START
```

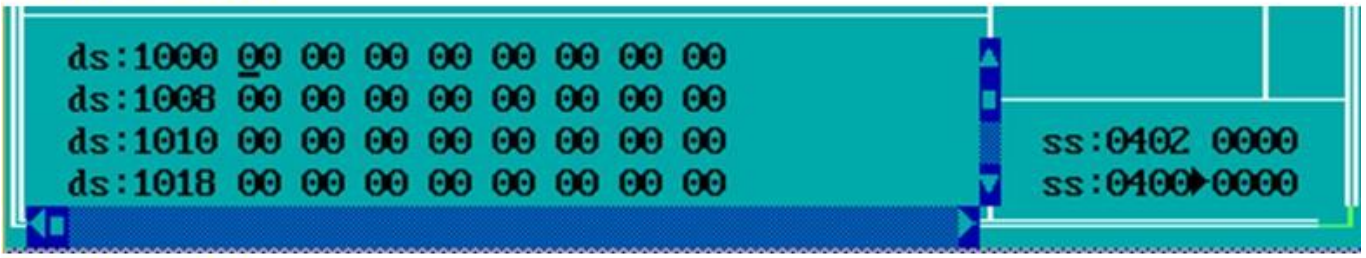
OUTPUT



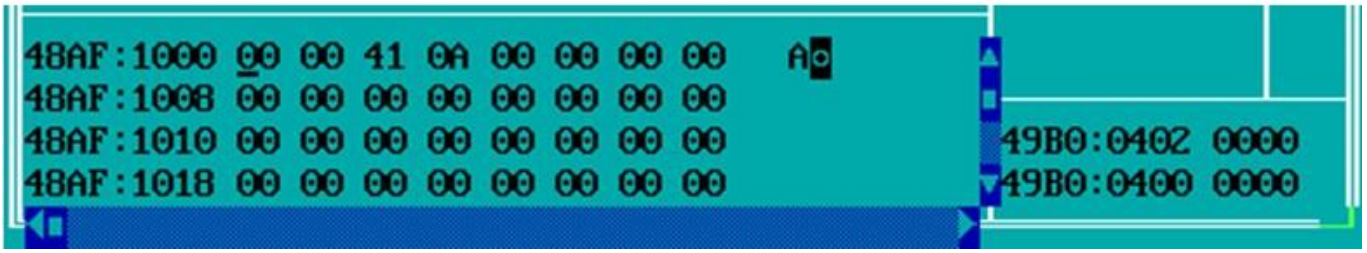
28. Read number from keyboard.

```
.MODEL SMALL
.STACK
        ORG 100H
.DATA
    ORG 1000H
    X DB 00H
    Y DB 00H
.CODE
START:
    MOV AX, @DATA
    MOV DS,AX

    MOV AH, 01
    INT 21H
    MOV X, AL

    CMP AL, 41H
    JC SKIP
    SUB AL, 37H
    MOV Y, AL
    JMP END1

SKIP:   SUB AL, 30H
        MOV Y, AL

END1:   MOV AH, 4CH
        INT 21H
        END START
```

Before Execution:



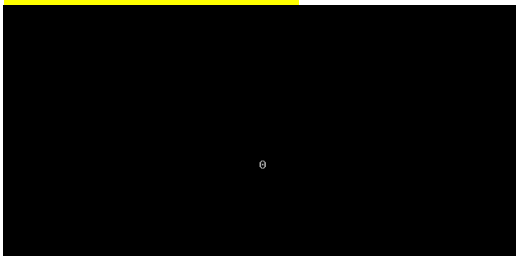
After Execution:



29. <u>Set cursor pointer in middle of a screen and display no from 0-9 (delay).</u>

```
 .MODEL SMALL
.STACK
      ORG 100H
.CODE
START:
      MOV AH,00H
      MOV  AL,03H
      INT 10H

      MOV   CH,"0"
      MOV CL,0AH

BACK: MOV AH,02H
      MOV DH,12
      MOV DL,40
      MOV BH,00
      INT 10H

      MOV   AH,02
      MOV  DL,CH
      INT 21H

      MOV BL,20H
L2:    MOV DX,0FFFFH
L1:    DEC DX
      JNZ L1
      DEC BL
      JNZ L2

      INC CH
      DEC CL
      JNZ BACK

      MOV AH,4CH
      INT 21H
      END START
```

AFTER EXECUTION:

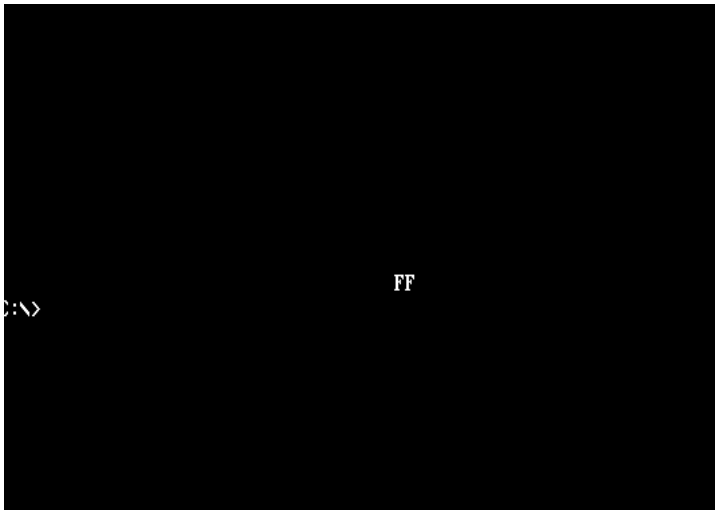


30. Implement Binary up counter 00 – FF (middle of the screen).

```
 .MODEL SMALL
.STACK
      ORG 100H
.DATA

.CODE
START: MOV AX, @DATA
      MOV DS, AX

      MOV BL, 00H

L2:    CALL SCROLL
       CALL CENTER

      MOV AL, BL

      CALL PRINT2
      CALL DELAY
      CALL DELAY
      CALL DELAY
      ADD BL, 01H
      CMP BL, 0FFH
      JB L2
      CALL CENTER
      MOV AL, BL
      CALL PRINT2
      CALL DELAY
      MOV AH,4CH
      INT 21H

PRINT2 PROC
      PUSH BX
      MOV BL, AL
      AND AL, 0F0H
      MOV CL, 04H
      ROR AL, CL
      CMP AL, 09H
      JG L5
      ADD AL, 30H
      JMP L6
L5: ADD AL, 37H
L6:    MOV DL, AL
      MOV AH, 02H
      INT 21H

      AND BL, 0FH
      CMP BL, 09H
      JG L7
      ADD BL, 30H
      JMP L8
L7:  ADD  BL,  37H
L8: MOV DL, BL
      MOV AH, 02H
      INT 21H
      POP BX
      RET
PRINT2 ENDP

CENTER PROC
      PUSH AX
      MOV DL, 25H
      MOV DH, 0CH
      MOV BH, 00H
      MOV AH, 02H
      INT 10H
      POP AX
      RET
```

```
CENTER ENDP

DELAY PROC NEAR
        PUSH AX
        PUSH BX
L9:  MOV  BX,  0FFFFH
L10: DEC BX
        JNZ L10
        POP  BX
        POP  AX
        RET
DELAY ENDP

SCROLL PROC
         MOV AH, 07H
        MOV AL, 00H
        MOV BH, 0FH
        MOV CX, 0000H
        MOV DH, 31H
        MOV DL, 79H
        INT 10H
         RET
SCROLL ENDP

END START
```

AFTER EXECUTION:

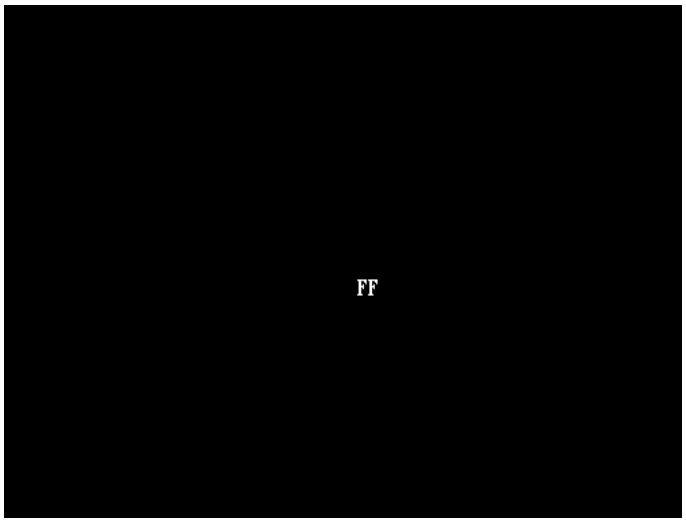


31. Implement Binary down counter FF – 00 (middle of the screen).

```
.MODEL SMALL
.STACK
      ORG 100H
.DATA

.CODE
START: MOV AX, @DATA
      MOV DS, AX

      MOV BL, 0FFH

L2:      CALL SCROLL
         CALL  CENTER

      MOV AL, BL

      CALL PRINT2
      CALL DELAY
      CALL DELAY
      CALL DELAY
      DEC BL
      CMP BL, 00H
      JNE L2
      CALL CENTER
      MOV AL, BL
      CALL PRINT2
      CALL DELAY
      MOV AH,4CH
      INT 21H

PRINT2 PROC
      PUSH BX
      MOV BL, AL
      AND AL, 0F0H
      MOV CL, 04H
      ROR AL, CL
      CMP AL, 09H
      JG L5
      ADD AL, 30H
      JMP L6
L5: ADD AL, 37H
L6: MOV DL, AL
      MOV AH, 02H
      INT 21H

      AND BL, 0FH
      CMP BL, 09H
      JG L7
      ADD BL, 30H
      JMP L8
L7:  ADD  BL,  37H
L8: MOV DL, BL
      MOV AH, 02H
      INT 21H
      POP BX
      RET
PRINT2 ENDP

CENTER PROC
      PUSH AX
      MOV DL, 25H
      MOV DH, 0CH
      MOV BH, 00H
      MOV AH, 02H
      INT 10H
      POP AX
      RET
CENTER ENDP

DELAY PROC NEAR
      PUSH AX
      PUSH BX
L9: MOV BX, 0FFFFH
```

```
L10: DEC BX
     JNZ L10
     POP BX
     POP AX
     RET
DELAY ENDP

SCROLL PROC
      MOV AH, 07H
     MOV AL, 00H
     MOV BH, 0FH
     MOV CX, 0000H
     MOV DH, 31H
     MOV DL, 79H
     INT 10H
      RET
SCROLL ENDP

END START
```

AFTER EXECUTION:



32. <u>Implement BCD up counter 00 – 99 (middle of the screen).</u>

```
 .MODEL SMALL
.STACK
      ORG 100H
.DATA
          X DB 0
.CODE
          EXTRN R1DIG:FAR
      EXTRN D1DIG:FAR
START:
      MOV AX, @DATA
      MOV DS, AX

      CALL R1DIG      ; AL = 0X
      MOV CL, 04
      ROR AL, CL
      MOV BL, AL      ; BL = X0
      CALL R1DIG      ; AL = 0Y
      OR AL, BL       ; AL = XY
      MOV X, AL

L1:    MOV AH, 07H
      MOV AL, 00H
      MOV BH, 0FH
      MOV CX, 00H
      MOV DH, 31H
      MOV DL, 79H
      INT 10H

      MOV AH, 02H
      MOV BH, 00H
      MOV DH, 0CH
      MOV DL, 25H
      INT 10H

          MOV AL, X
          AND AL, 0F0H
          MOV CL, 04H
          ROR AL, CL
          CALL D1DIG
          MOV AL, X
      AND AL, 0FH
          CALL D1DIG

      MOV AL, X
      CMP AL, 99H
      JZ EXIT
          CALL DELAY
          MOV AL, X
      ADD AL, 01
```

```
        DAA
        MOV X, AL
    JMP L1

EXIT: MOV AH, 4CH
    INT 21H

DELAY PROC NEAR
    MOV DX, 1000H
B2:     MOV CX, 0090H
B1:   LOOP B1
    DEC DX
    JNZ B2
    RET
DELAY ENDP
    END START
```

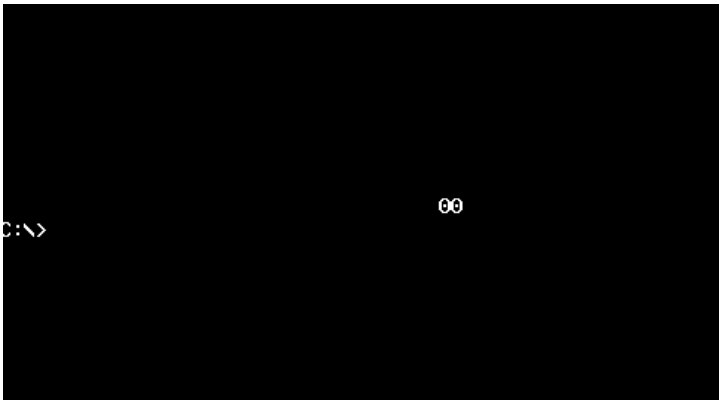
AFTER EXECUTION:

33. Implement BCD down counter 99 – 00 (middle of the screen).

```
 .MODEL SMALL
.STACK
      ORG 100H
.DATA
          X DB 0
.CODE
          EXTRN R1DIG:FAR
      EXTRN D1DIG:FAR
START:
      MOV AX, @DATA
      MOV DS, AX

     MOV X,99H

L1:    MOV AH, 07H
      MOV AL, 00H
      MOV BH, 0FH
      MOV CX, 00H
      MOV DH, 31H
      MOV DL, 79H
      INT 10H

      MOV AH, 02H
      MOV BH, 00H
      MOV DH, 0CH
      MOV DL, 25H
      INT 10H

          MOV AL, X
          AND AL, 0F0H
          MOV CL, 04H
          ROR AL, CL
          CALL D1DIG
          MOV AL, X
      AND AL, 0FH
          CALL D1DIG

      MOV AL, X
      CMP AL, 00H
      JZ EXIT
          CALL DELAY
          MOV AL, X
      SUB AL, 01
          DAS
          MOV X, AL
      JMP L1

EXIT: MOV AH, 4CH
      INT 21H
```

```
DELAY PROC NEAR
        MOV DX, 1000H
B2:     MOV CX, 0090H
B1:     LOOP B1
        DEC DX
        JNZ B2
        RET
DELAY ENDP
        END START
```

AFTER EXECUTION



34. Display system date (middle of the screen).

```
.MODEL SMALL
.DATA
.CODE
START: MOV AX,@DATA
MOV DS,AX

;Day Part
DAY:
MOV AH,2AH ; To get System Date
INT 21H
MOV AL,DL ; Day is in DL
AAM
MOV BX,AX
CALL DISP

MOV DL,'/'
MOV AH,02H ; To Print / in DOS
INT 21H

;Month Part
MONTH:
MOV AH,2AH ; To get System Date
INT 21H
MOV AL,DH     ; Month is in DH
AAM
MOV BX,AX
CALL DISP

MOV DL,'/' ; To Print / in DOS
MOV AH,02H
INT 21H

;Year Part
YEAR:
MOV AH,2AH ; To get System Date
INT 21H
ADD CX,0F830H ; To negate the effects of 16bit value,
MOV AX,CX     ; since AAM is applicable only for AL (YYYY -> YY)
AAM
MOV BX,AX
CALL DISP

;To terminate the Program

MOV AH,4CH     ; To Terminate the Program
INT 21H


;Display Part

DISP PROC
MOV DL,BH      ; Since the values are in BX, BH Part
ADD DL,30H     ; ASCII Adjustment
MOV AH,02H     ; To Print in DOS
INT 21H
MOV DL,BL      ; BL Part
ADD DL,30H     ; ASCII Adjustment
MOV AH,02H     ; To Print in DOS
INT 21H
RET
DISP ENDP      ; End Disp Procedure


END START      ; End of MAIN
```

35. Display system time (middle of the screen).

```
.MODEL SMALL
.DATA
.CODE
START: MOV AX,@DATA
MOV DS,AX
mov ah,3    ; Get the current cursor position
mov bh,0
int 10h
mov ah,2    ; Set cursor position
mov bh,0
mov dl,39
int 10h
;Hour Part
HOUR:
MOV AH,2CH    ; To get System Time
INT 21H
MOV AL,CH     ; Hour is in CH
AAM
MOV BX,AX
CALL DISP
MOV DL,':'
MOV AH,02H    ; To Print : in DOS
INT 21H
;Minutes Part
MINUTES:
MOV AH,2CH    ; To get System Time
INT 21H
MOV AL,CL     ; Minutes is in CL
AAM
MOV BX,AX
CALL DISP
MOV DL,':'    ; To Print : in DOS
MOV AH,02H
INT 21H
;Seconds Part
Seconds:
MOV AH,2CH    ; To get System Time
INT 21H
MOV AL,DH     ; Seconds is in DH
AAM
MOV BX,AX
CALL DISP
 ;To terminate the Program
MOV AH,4CH     ; To Terminate the Program
INT 21H
;Display Part
DISP PROC
MOV DL,BH      ; Since the values are in BX, BH Part
ADD DL,30H     ; ASCII Adjustment
MOV AH,02H     ; To Print in DOS
```

```
INT 21H
MOV DL,BL       ; BL Part
ADD DL,30H      ; ASCII Adjustment
MOV AH,02H      ; To Print in DOS
INT 21H
RET
DISP ENDP       ; End Disp Procedure
END START       ; End of MAIN
```

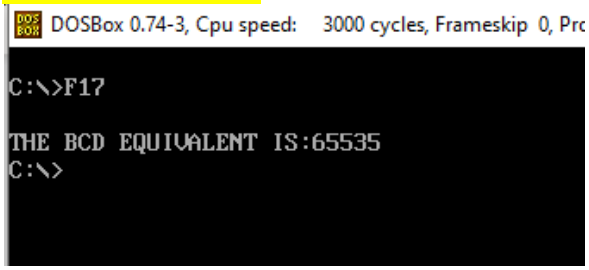
```
X:\>time.exe
                                        21:24:05
X:\>S_
```

36. Convert hexadecimal number to BCD number.

```
  .MODEL SMALL
.STACK

DATA  SEGMENT
BINV  DW  00DFH
BCD DB 2 DUP(0)
DATA ENDS
ASSUME CS:CODE, DS:DATA
CODE SEGMENT
START:MOV AX,DATA
      MOV DS,AX
      MOV AX,BINV
      MOV CL,64H
      DIV CL
      MOV BCD+1,AL
      MOV AL,AH
      MOV AH,00H
      MOV CL,0AH
      DIV CL
      MOV CL,04
      ROR AL,CL
      ADD AL,AH
      MOV AH,4CH
      INT 21H
CODE ENDS
END START
```

BEFORE EXECUTION

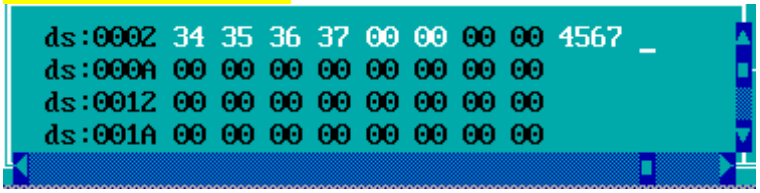
AFTER EXECUTION



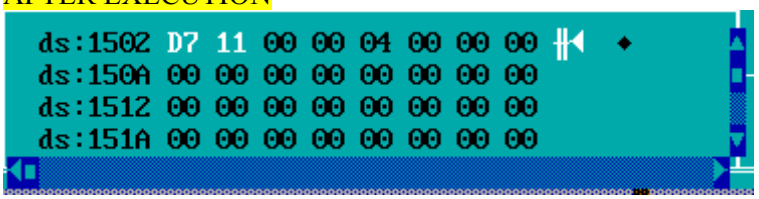
37. Convert BCD number to hexadecimal number

```
.MODEL SMALL
.STACK

.DATA

BCD DB '4567'
ORG 1500H
HEX_NUM DW 0
MULT DW 1000
DIGIT DW 4
.CODE
MOV AX , @DATA
MOV DS , AX

MOV BX , 10

MOV CX , DIGIT
LEA SI , BCD

  LAST:
MOV AL , [SI]
AND AX , 000FH
MUL MULT
ADD HEX_NUM , AX
MOV AX , MULT
MOV DX , 00
DIV BX
MOV MULT, AX
INC SI
LOOP LAST

MOV AX , HEX_NUM

MOV AH , 4CH
INT 21H
END
```

BEFORE EXECUTION



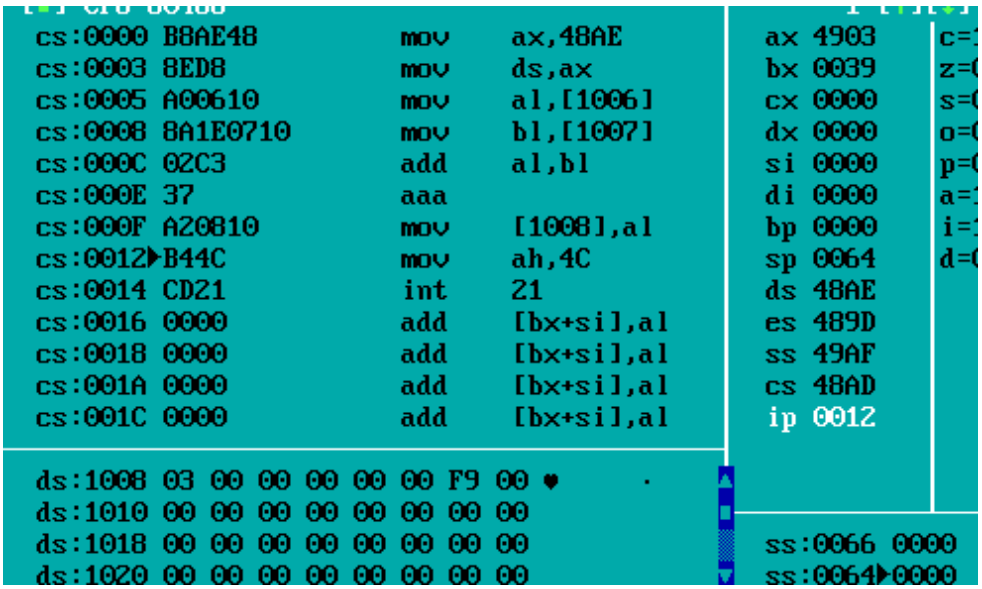
AFTER EXECUTION



38. <u>Program to Add two ASCII numbers.</u>

```
.MODEL SMALL
.STACK 100
.DATA
ORG 1000H
NUM1 DB "4"
NUM2 DB "9"
SUM DB ?

.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV AL,NUM1
        MOV BL,NUM2
        ADD AL,BL
        AAA
        MOV SUM,AL
        MOV  AH,4CH
        INT 21H
        END
```

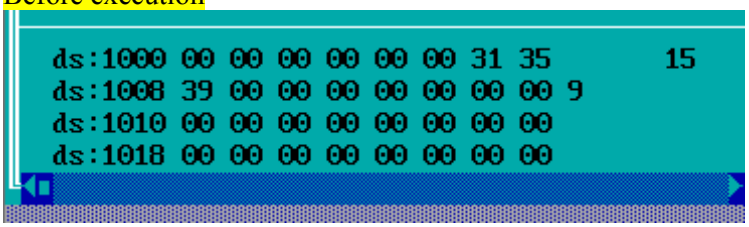
AFTER EXECUTION



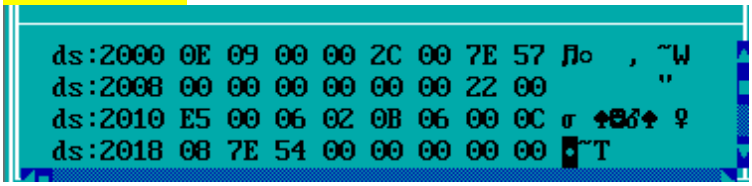
39. <u>Program to subtract two ASCII numbers</u>

```
.MODEL SMALL
.STACK
 ORG 100H
.DATA
ORG 1000H

NUM1 DB "15"
NUM2 DB "9"
SUB DB ?

.CODE
   MOV AX,@DATA
   MOV DS,AX
   MOV AL,NUM1
   MOV BL,NUM2
   sub AL,BL
   AAS
   MOV SUB,AL
   MOV AH,4CH
   INT 21H
END
```

Before execution



After execution



40. Program to multiply two ASCII numbers.

```
.MODEL SMALL
.STACK
.DATA
ORG 1000H
A DB 36H
B DB 35H
C DW 00H
.CODE
START:
MOV AX, @DATA
MOV DS,AX

MOV AH,00
MOV AL,A
MOV BH,00
MOV BL,B
AND AL, 0FH
AND BL,0FH
MUL BL
MOV C,AX
  MOV AH,4CH
  INT 21H
   END START
```

Before Execution

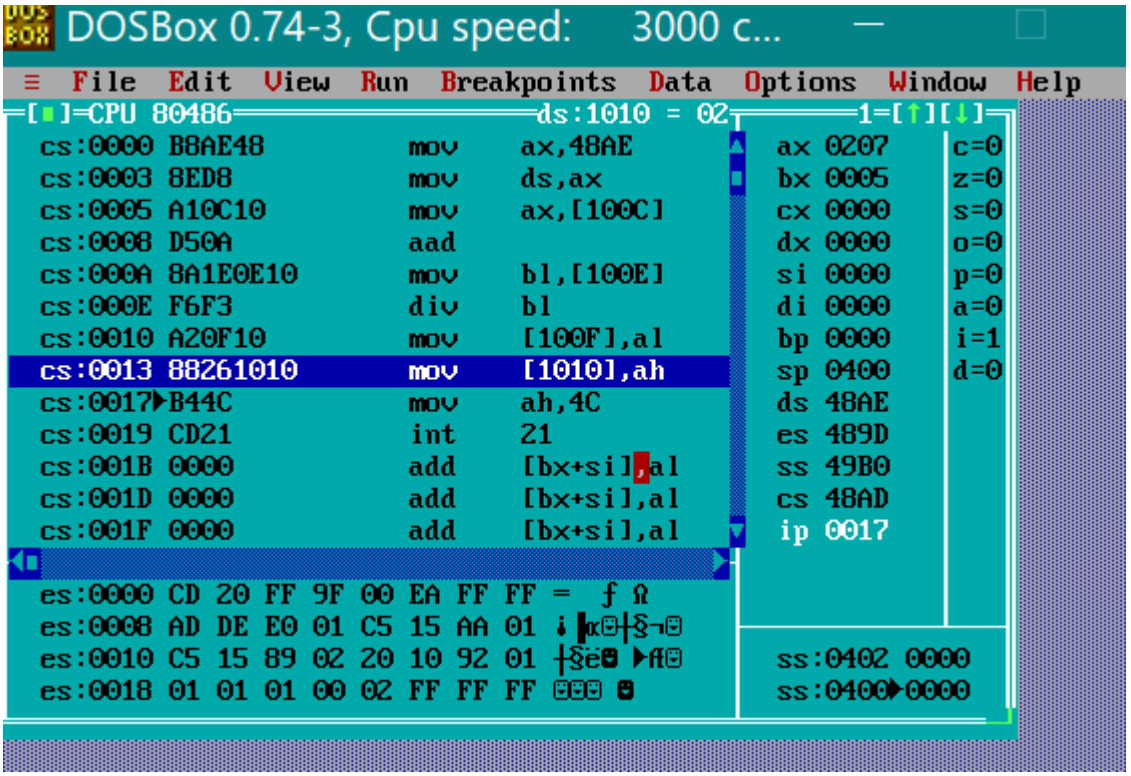
```
ds:0000 CD 20 FF 9F 00 EA FF FF =  f Ω
ds:0008 AD DE E0 01 C5 15 AA 01 ↓ ♠α⊖┤S¬⊖
ds:0010 C5 15 89 02 20 10 92 01 ┼Se■ ►A⊖
ds:0018 01 01 01 00 02 FF FF FF ⊖⊖⊖ ■
```

AFTER EXECUTION

```
48AE:1010 1E 00 00 00 00 00 00 00 ▲
48AE:1018 00 00 00 00 00 00 00 00
48AE:1020 00 00 00 00 00 00 00 00
48AE:1028 00 00 00 00 00 00 00 00
```

41. Program to divide two ASCII numbers

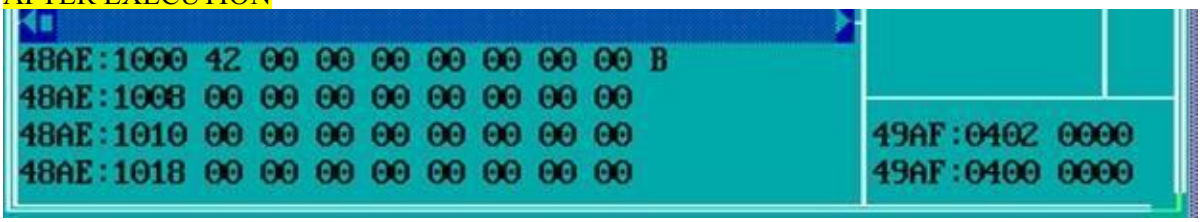
```
.MODEL SMALL
.STACK
      ORG 100H
.DATA
     ORG 1000H
     NUM1 DW 0307H
     NUM2 DB 5
      Q DB ?
      R DB ?
.CODE
START:
      MOV AX,@DATA
      MOV DS,AX

      MOV AX,NUM1
      AAD
      MOV BL,NUM2
      DIV BL
      MOV Q,AL
      MOV R,AH

      MOV AH,4CH
      INT 21H
      END START
```



42. Program to Read a Character With ECHO.

```
.MODEL SMALL
.STACK
    ORG 100H
.DATA
    ORG 1000H
    X DB 00H     ;NO READ USING AH=01
.CODE
START:
    MOV AX, @DATA
    MOV DS, AX

    MOV AH, 01     ;WITH ECHO
    INT 21H
    MOV X, AL

    MOV AH, 4CH
    INT 21H
    END START
```

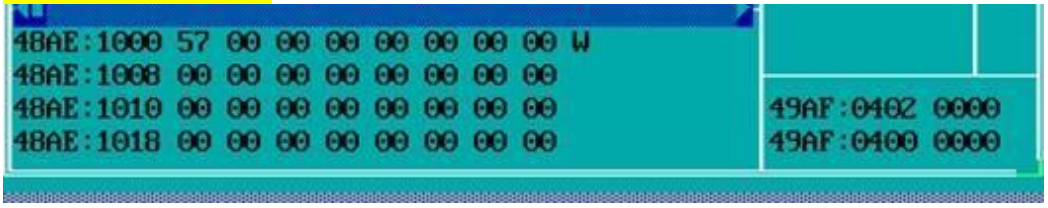
AFTER EXECUTION



43. Program to Read a Character Without ECHO

```
.MODEL SMALL
.STACK
    ORG 100H
.DATA
    ORG 1000H
    Y DB 00H     ;NO READ USING AH=08
.CODE
START:
    MOV AX, @DATA
    MOV DS, AX

    MOV AH, 08 ;WITHOUT ECHO
    INT 21H
    MOV Y,AL

    MOV AH, 4CH
    INT 21H
    END START
```

AFTER EXECUTION



44. Program to Transfer A String from Source to Destination

```
.MODEL SMALL

.DATA
        ORG 1000H
        SRC DB "MANIPAL DUBAI 1234567"
        LEN EQU $-SRC
        ORG 2000H
        DST DB 50 DUP(0)

.CODE
START:
         MOV AX, @DATA
      MOV DS, AX
      MOV ES, AX

      LEA SI, SRC
      LEA DI, DST
      MOV CX, LEN
      CLD
      REP MOVSB

     MOV AH, 4CH
      INT 21H
      END START
```

BEFORE EXECUTION

```
ds:100A 4D 41 4E 49 50 41 4C 20 MANIPAL
ds:1012 44 55 42 41 49 20 31 32 DUBAI 12
ds:101A 33 34 35 36 37 00 00 00 34567
ds:1022 00 00 00 00 00 00 00 00
```

```
ds:200A 00 00 00 00 00 00 00 00
ds:2012 00 00 00 00 00 00 00 00
ds:201A 00 00 00 00 00 00 00 00
ds:2022 00 00 00 00 00 00 00 00
```

AFTER EXECUTION

```
ds:200A 4D 41 4E 49 50 41 4C 20 MANIPAL
ds:2012 44 55 42 41 49 20 31 32 DUBAI 12
ds:201A 33 34 35 36 37 00 00 00 34567
ds:2022 00 00 00 00 00 00 00 00
```

45. Program to Find Whether A Given String Is A Palindrome or Not

```
.MODEL SMALL

.DATA
      ORG 100AH
      SRC DB "MADAAM"
      LEN EQU $-SRC
      ORG 1012H
      DST DB 20 DUP (?)
      ORG 3000H
      MSG1 DB "THE STRING IS A PALINDROME$"
      MSG2 DB "THE STRING IS NOT A PALINDROME$"

.CODE
START:
       MOV AX, @DATA
    MOV DS, AX
    MOV ES, AX

    CALL PAL

       MOV AH, 4CH
    INT 21H

PAL PROC NEAR
    MOV CX, LEN
    LEA SI, SRC
    LEA DI, DST
       ADD DI, CX
       DEC DI

BACK: MOV AL, [SI]
    MOV [DI], AL
    INC SI
    DEC DI
    LOOP BACK

       MOV CX, LEN
    LEA SI, SRC
       LEA DI, DST
    CLD
       REPE CMPSB
       JNZ SKIP
       LEA DX, MSG1 ; YES
    MOV AH, 09H
    INT 21H
       JMP L1
SKIP: LEA DX, MSG2 ; NO
    MOV AH, 09H
    INT 21H
L1:     RET
PAL ENDP
    END START
```

BEFORE EXECUTION

```
ds:1012 4D 41 44 41 41 4D 00 00 MADAAM
ds:101A 00 00 00 00 00 00 00 00
ds:1022 00 00 00 00 00 00 00 00
ds:102A 00 00 00 00 00 00 00 00
```

AFTER EXECUTION

```
C:\>STR2
THE STRING IS NOT A PALINDROME
C:\>_
```

46. Program to reverse a given string

```
.MODEL SMALL
.DATA
        ORG 1000H
        STR1 DB "SEMESTER"
        LEN EQU $-STR1
    SPACE DB 5 DUP(0)
    STR2 DB LEN DUP(0)

.CODE
START:

    MOV AX,@DATA
    MOV DS,AX

    LEA SI, STR1
    ADD SI, LEN-1
    LEA DI, STR2
    MOV CX, LEN

BACK: MOV AL,[SI]
    MOV [DI], AL
    DEC SI
    INC DI
    LOOP BACK

    MOV AH, 4CH
    INT 21H
    END START
```

BEFORE EXECUTION

```
ds:1000 53 45 4D 45 53 54 45 52 SEMESTER
ds:1008 00 00 00 00 00 00 00 00
ds:1010 00 00 00 00 00 00 00 00
ds:1018 00 00 00 00 00 00 00 00
```

AFTER EXEUTION

```
ds:1000 53 45 4D 45 53 54 45 52 SEMESTER
ds:1008 00 00 00 00 00 52 45 54      RET
ds:1010 53 45 4D 45 53 00 00 00 SEMES
ds:1018 00 00 00 00 00 00 00 00
```

47. Program to search a character in a given string

```
.MODEL SMALL
.DATA
      STR DB "ABDULLA"
   LEN EQU ($-STR)
      CHAR DB "X"
   MSG1 DB 0AH, 0DH," CHARACTER FOUND $"
   MSG2 DB 0AH, 0DH," CHARACTER NOT FOUND $"

.CODE
START:
   MOV AX,@DATA
   MOV DS, AX
   MOV ES, AX

   MOV CX, LEN
   LEA DI, STR
   MOV AL, CHAR
   CLD
      REPNE SCASB            ; INC DI, DEC CX
   JE FOUND
      MOV DX, OFFSET MSG2 ; NOT
   MOV AH, 09H
   INT 21H
   JMP EXIT
FOUND: MOV DX, OFFSET MSG1     ; YES
   MOV AH, 09H
   INT 21H
EXIT: MOV AH, 4CH
   INT 21H
   END START
```

BEFORE EXECUTION

```
ds:000C 41 42 44 55 4C 4C 41 58 ABDULLAX
ds:0014 0A 0D 20 43 48 41 52 41 ♪♪ CHARA
ds:001C 43 54 45 52 20 46 4F 55 CTER FOU
ds:0024 4E 44 20 24 0A 0D 20 43 ND $♪♪ C
```

AFTER EXECUTION

```
C:\>STR4

 CHARACTER NOT FOUND
C:\>_
```