

CONTENT TABLE
<u>1. Copy a block of byte data from one location to another.</u>
<u>2. Copy a block of word data from one location to another.</u>
<u>3. Exchange block of byte data.</u>
<u>4. Exchange block of word data.</u>
<u>5. To add two data bytes.</u>
<u>6. To add two data words.</u>
<u>7. Find sum of a byte array.</u>
<u>8. Find the sum of a word array.</u>
<u>9. Display string using interrupt.</u>
<u>10. Find the smallest number in an array.</u>
<u>11. Find the largest number in an array.</u>
<u>12. Display if the given number is even or odd.</u>
<u>13. Display if the given number is positive or negative.</u>
<u>14. In array find number of positive numbers and negative numbers.</u>
<u>15. In array find number of even numbers and odd numbers.</u>
<u>16. Program to separate even numbers and odd numbers in an array.</u>
<u>17. Program to separate positive numbers and negative numbers in an array.</u>
<u>18. Print a character or symbol using interrupt.</u>
<u>19. Program to sort an array in ascending order.</u>
<u>20. Program to sort an array in descending order.</u>
<u>21. Program to reverse a binary number.</u>
<u>22. Count the number of 0's and 1's in a number.</u>
<u>23. Add the correspond element of 2 arrays and save result in a sum of array.</u>
<u>24. Check if the given number belongs to 2 out of 5 code.</u>
<u>25. Program to divide a 32-bit number by 16-bit number.</u>
<u>26. Search for a key element in an array.</u>
<u>27. Print msg using macro.</u>
<u>28. Read number from keyboard.</u>

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

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 :

```
ds:100E 12 47 36 48 FD 00 00 00 00 ‡G6H²
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
```

```
ds:200E 00 00 00 00 00 00 00 00
ds:2016 00 00 00 00 00 00 00 00
ds:201E 00 00 00 00 00 00 00 00
ds:2026 00 00 00 00 00 00 00 00
```

AFTER EXECUTION :

```
ds:200E 12 47 36 48 FD 00 00 00 00 ‡G6H²
ds:2016 00 00 00 00 00 00 00 00
ds:201E 00 00 00 00 00 00 00 00
ds:2026 00 00 00 00 00 00 00 00
```

```
ds:100E 12 47 36 48 FD 00 00 00 00 ‡G6H²
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
```

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

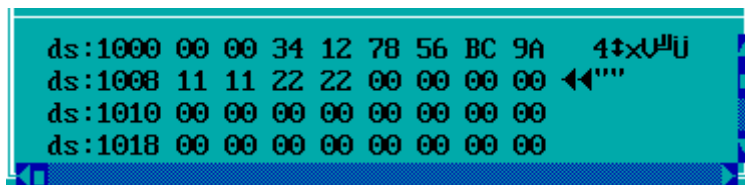
```
.MODEL SMALL
.STACK
.ORG 100H
.DATA
.ORG 1000H
SRC DW 1234H, 5678H, 9ABCH, 1111H, 2222H
.ORG 2000H
DST DW 5 DUP(0)
.CODE
START:
MOV AX, @DATA
MOV DS, AX

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

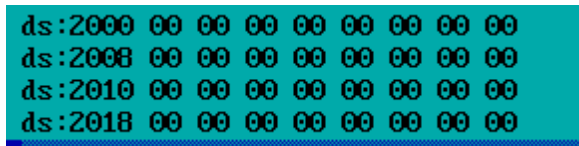
DEC CL
JNZ L1

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

BEFORE EXECUTION

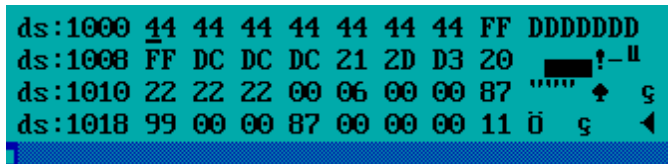


```
ds:1000 00 00 34 12 78 56 BC 9A
ds:1008 11 11 22 22 00 00 00 00
ds:1010 00 00 00 00 00 00 00 00
ds:1018 00 00 00 00 00 00 00 00
```

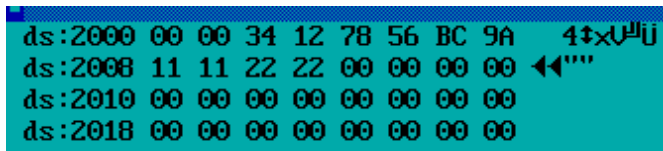


```
ds:2000 00 00 00 00 00 00 00 00
ds:2008 00 00 00 00 00 00 00 00
ds:2010 00 00 00 00 00 00 00 00
ds:2018 00 00 00 00 00 00 00 00
```

AFTER EXECUTION



```
ds:1000 44 44 44 44 44 44 44 FF
ds:1008 FF DC DC DC 21 2D D3 20
ds:1010 22 22 22 00 06 00 00 87
ds:1018 99 00 00 87 00 00 00 11
```



```
ds:2000 00 00 34 12 78 56 BC 9A
ds:2008 11 11 22 22 00 00 00 00
ds:2010 00 00 00 00 00 00 00 00
ds:2018 00 00 00 00 00 00 00 00
```

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 00 ▶D½Pm
ds:1008 00 00 00 00 00 00 00 00 00
ds:1010 00 00 00 00 00 00 00 00 00
ds:1018 00 00 00 00 00 00 00 00 00
```

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

AFTER EXECUTION:

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

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

```
ds:1004 34 12 78 56 BC 9A F0 DE 4F 00 00 00 00 00 00
ds:100C 62 15 00 00 00 00 00 00 00 00 00 00 00 00
ds:1014 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:101C 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
ds:2004 23 F1 67 F5 92 F8 45 F3 8E 00 00 00 00 00 00
ds:200C 23 F4 00 00 00 00 00 00 00 00 00 00 00 00
ds:2014 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:201C 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

AFTER EXECUTION:

```
ds:1004 23 F1 67 F5 92 F8 45 F3 8E 00 00 00 00 00 00
ds:100C 23 F4 00 00 00 00 00 00 00 00 00 00 00 00
ds:1014 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:101C 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
ds:2004 34 12 78 56 BC 9A F0 DE 4F 00 00 00 00 00 00
ds:200C 62 15 00 00 00 00 00 00 00 00 00 00 00 00
ds:2014 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:201C 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

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:0000 E6 01 AF 15 B0 01 AF 15  µC>§  µC>§
ds:0012 83 02 13 10 92 01 01 01  âC!!!µCµCµC
ds:001A 01 00 02 FF FF FF FF FF  µ µ
ds:0022 FF FF FF FF FF FF FF FF
```

AFTER EXECUTION:

```
ds:0000 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. To add two data words.

```
.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

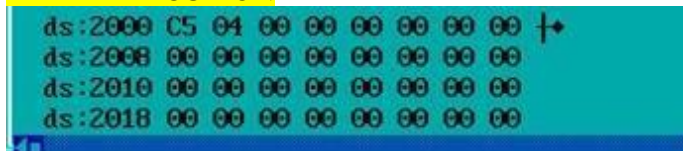
        MOV AH, 4CH
        INT 21H
        END START
```

BEFORE EXECUTION



```
ds:1000 F1 F5 F8 F3 F4 00 00 00 00 00 00
ds:1008 00 00 00 00 00 00 00 00 00 00 00
ds:1010 00 00 00 00 00 00 00 00 00 00 00
ds:1018 00 00 00 00 00 00 00 00 00 00 00
```

AFTER EXECUTION



```
ds:2000 C5 04 00 00 00 00 00 00 00 00 00
ds:2008 00 00 00 00 00 00 00 00 00 00 00
ds:2010 00 00 00 00 00 00 00 00 00 00 00
ds:2018 00 00 00 00 00 00 00 00 00 00 00
```

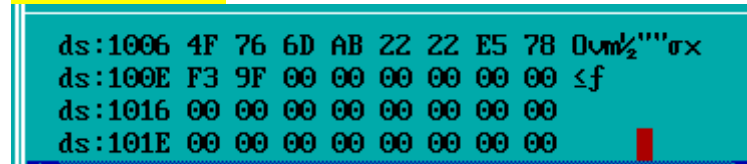
8. Find the sum of a word array.

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

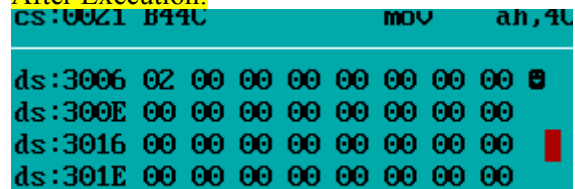
    MOV AH, 4CH
    INT 21H
    END START
```

Before Execution:

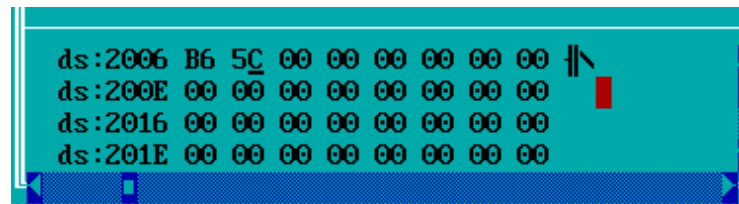


ds:1006 4F 76 6D AB 22 22 E5 78
ds:100E F3 9F 00 00 00 00 00 00
ds:1016 00 00 00 00 00 00 00 00
ds:101E 00 00 00 00 00 00 00 00

After Execution:



cs:0021 B44C mov ah,4C
ds:3006 02 00 00 00 00 00 00 00
ds:300E 00 00 00 00 00 00 00 00
ds:3016 00 00 00 00 00 00 00 00
ds:301E 00 00 00 00 00 00 00 00



ds:2006 B6 5C 00 00 00 00 00 00
ds:200E 00 00 00 00 00 00 00 00
ds:2016 00 00 00 00 00 00 00 00
ds:201E 00 00 00 00 00 00 00 00

9. Display string using interrupt.

```
.MODEL SMALL
.STACK 100H
.DATA
    STRING DB 'MANIPAL ACADEMY OF HIGHER EDUCATIONS'
.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 ATIONS\$

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

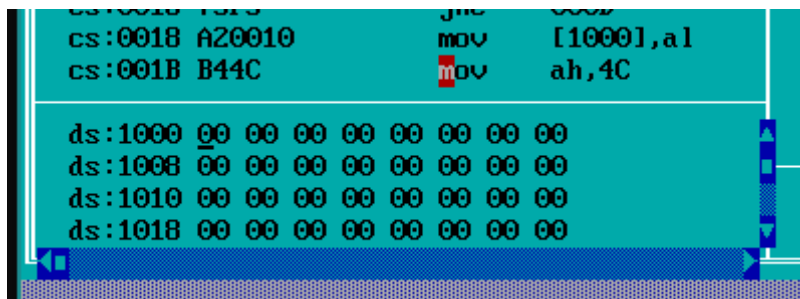
    LEA SI, AR
    MOV CL, 05H

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

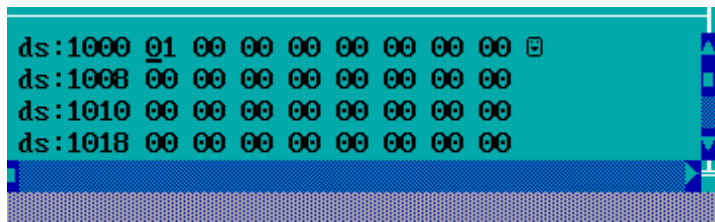
NEXTNO: INC SI
    DEC CL
    JNZ BACK

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

BEFORE EXECUTION:



AFTER EXECUTION:



11. **Find the largest number in an array.**

```
.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

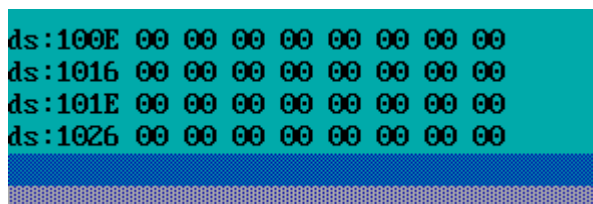
    LEA SI, AR
    MOV CL, 05H

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

NEXTNO: INC SI
        DEC CL
        JNZ BACK

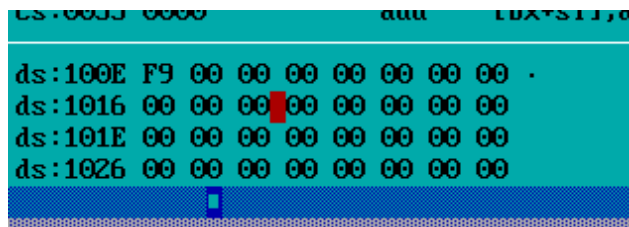
    MOV AH, 4CH
    INT 21H
    END START
```

BEFORE EXECUTION:



ds:100E	00	00	00	00	00	00	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:



ds:100E	F9	00	00	00	00	00	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

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 00 E
ds:1008 00 00 00 00 00 00 00 00 00
ds:1010 00 00 00 00 00 00 00 00 00
ds:1018 00 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 NUMBERS'  
    MSG2 DB 'NEGATIVE NUMBERS'
```

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

```
        MOV AL,X
```

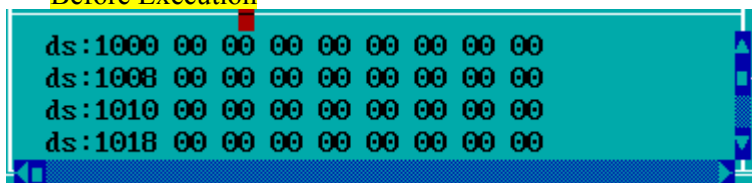
```
        AND AL,80H  
        JNZ POS1
```

```
        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

```
64 kb freed.  
X:\>PN.EXE  
POSITIVE NUMBER  
X:\>
```

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
```



```

.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

```

```
ds:1000 12 34 71 AB 33 FF 00 6E 40 12 3 n
ds:1008 90 45 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 05 00 00 00 00 00 00 00
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
```

```
ds:3000 05 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
```

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:

```
ds:1000 10 02 04 01 08 00 00 00 00 ▶◀Ⓢ
ds:1008 00 00 00 00 00 00 00 00 00
ds:1010 00 00 00 00 00 00 00 00 00
ds:1018 00 00 00 00 00 00 00 00 00

ds:2000 00 00 00 00 00 00 00 00 00
ds:2008 00 00 00 00 00 00 00 00 00
ds:2010 00 00 00 00 00 00 00 00 00
ds:2018 00 00 00 00 00 00 00 00 00

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

AFTER EXECUTION:

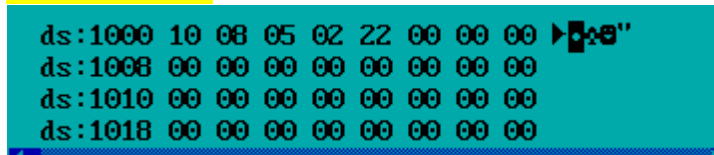
```
ds:2000 10 02 04 08 00 00 00 00
ds:2008 00 00 00 00 00 00 00
ds:2010 00 00 00 00 00 00 00
ds:2018 00 00 00 00 00 00 00

ds:3000 01 00 00 00 00 00 00
ds:3008 00 00 00 00 00 00 00
ds:3010 00 00 00 00 00 00 00
ds:3018 00 00 00 00 00 00 00
```

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

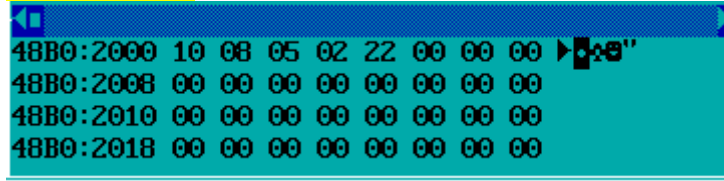
```
.MODEL SMALL
.STACK
ORG 100H
.DATA
ORG 1000H
X DB 10H, 08H, 05H, 02H, 22H
ORG 2000H
POSX DB 5 DUP(0)
ORG 3000H
NEGX DB 5 DUP(0)
.CODE
START :
MOV AX, @DATA
MOV DS, AX
LEA BX, X
LEA SI, POSX
LEA DI, NEGX
MOV CL, 05H
L1 : MOV AL, [BX]
AND AL, 80H
JNZ XYZ ; JUMP TO XYZ IF Z=0 OR ODD NUMBER
MOV AL, [BX]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

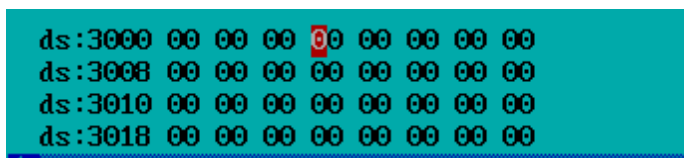


```
ds:1000 10 08 05 02 22 00 00 00
ds:1008 00 00 00 00 00 00 00 00
ds:1010 00 00 00 00 00 00 00 00
ds:1018 00 00 00 00 00 00 00 00
```

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. Program to sort an array in ascending order.

```
.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

ds:0000	CD 20 FF 9F 00 EA FF FF	= f 0	
ds:0008	AD DE E0 01 C5 15 AA 01	12 55 84 FF	
ds:0010	C5 15 89 02 20 10 92 01	00 00 00 00	ss:0402 0000
ds:0018	01 01 01 00 02 FF FF FF	00 00 00 00	ss:0400 0000

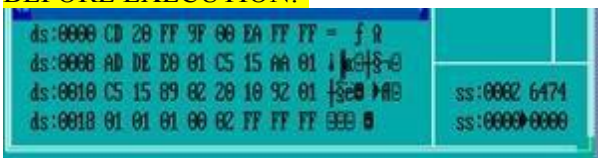
AFTER EXECUTION

48AF:1000	00 00 00 00 00 00 00 00		
48AF:1008	12 2E 55 84 FF 00 00 00	12 55 84 FF	
48AF:1010	00 00 00 00 00 00 00 00	00 00 00 00	49B0:0402 0000
48AF:1018	00 00 00 00 00 00 00 00	00 00 00 00	49B0:0400 0000

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, ARRAY[SI]
    MOV DL, ARRAY[SI+1]
    CMP AL, DL
    JNC NOSWAP
    MOV ARRAY[SI], DL
    MOV ARRAY[SI+1], AL
NOSWAP:
    INC SI
    DEC BX
    JNZ COMP
LOOP SCAN
MOV AH, 4CH
INT 21H
END
```

BEFORE EXECUTION:-



A screenshot of a memory dump window showing the state of memory before execution. The window has a title bar and a menu bar. The main area displays memory addresses and their corresponding values in hexadecimal. The address range is from 0000 to 001B. The values are: 0000: CD 20 FF 9F 00 EA FF FF, 0001: AD DE 00 01 C5 15 AA 01, 0002: C5 15 09 02 20 10 92 01, 0003: 01 01 01 00 02 FF FF FF. The right side of the window shows the segment register SS at 0002:6474 and the instruction pointer at 0000:0000.

ds:0000	CD 20 FF 9F 00 EA FF FF	
ds:0001	AD DE 00 01 C5 15 AA 01	
ds:0002	C5 15 09 02 20 10 92 01	ss:0002 6474
ds:0003	01 01 01 00 02 FF FF FF	ss:0000 0000

AFTER EXECUTION:-



A screenshot of a memory dump window showing the state of memory after execution. The window displays memory addresses and their corresponding values in hexadecimal. The address range is from 000E to 0026. The values are: 000E: 12 98 45 83 28 15 40 54, 000F: 00 00 00 00 00 00 00 00, 0010: 00 00 00 00 00 00 00 00, 0011: 00 00 00 00 00 00 00 00. The right side of the window shows the segment register SS at 0002:6474 and the instruction pointer at 0000:0000.

ds:000E	12 98 45 83 28 15 40 54	
ds:000F	00 00 00 00 00 00 00 00	
ds:0010	00 00 00 00 00 00 00 00	ss:0002 6474
ds:0011	00 00 00 00 00 00 00 00	ss:0000 0000



A screenshot of a memory dump window showing the state of memory after execution. The window displays memory addresses and their corresponding values in hexadecimal. The address range is from 000F to 0027. The values are: 000F: 83 54 45 40 28 15 12 00, 0010: 00 00 00 00 00 00 00 00, 0011: 00 00 00 00 00 00 00 00, 0012: 00 00 00 00 00 00 00 00. The right side of the window shows the segment register SS at 0002:6474 and the instruction pointer at 0000:0000.

ds:000F	83 54 45 40 28 15 12 00	
ds:0010	00 00 00 00 00 00 00 00	
ds:0011	00 00 00 00 00 00 00 00	ss:0002 6474
ds:0012	00 00 00 00 00 00 00 00	ss:0000 0000

21. Program to reverse a binary number.

```
.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 00 00
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
□
```

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

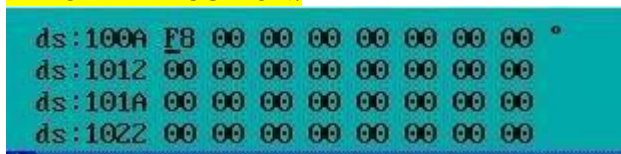
ADDBH: INC BH

SKIP: DEC CL
    JNZ BACK

    MOV Y,BH
    MOV Z,BL

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

BEFORE EXECUTION:

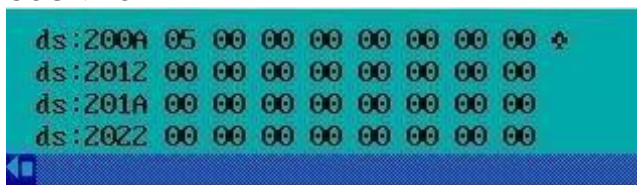


A screenshot of a memory dump window showing four lines of memory addresses and their contents. The first line, ds:100A, contains the value FB followed by eight 00s. The other three lines (ds:1012, ds:101A, ds:1022) each contain eight 00s.

Address	Content
ds:100A	FB 00 00 00 00 00 00 00 00
ds:1012	00 00 00 00 00 00 00 00
ds:101A	00 00 00 00 00 00 00 00
ds:1022	00 00 00 00 00 00 00 00

AFTER EXECUTION:

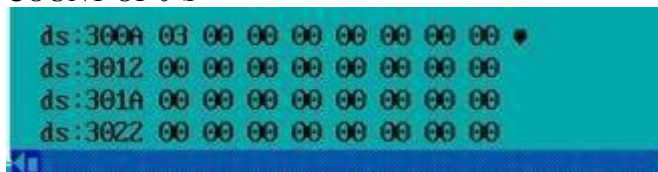
COUNT OF 1'S



A screenshot of a memory dump window showing four lines of memory addresses and their contents. The first line, ds:200A, contains the value 05 followed by seven 00s. The other three lines (ds:2012, ds:201A, ds:2022) each contain eight 00s.

Address	Content
ds:200A	05 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

COUNT OF 0'S



A screenshot of a memory dump window showing four lines of memory addresses and their contents. The first line, ds:300A, contains the value 03 followed by seven 00s. The other three lines (ds:3012, ds:301A, ds:3022) each contain eight 00s.

Address	Content
ds:300A	03 00 00 00 00 00 00 00
ds:3012	00 00 00 00 00 00 00 00
ds:301A	00 00 00 00 00 00 00 00
ds:3022	00 00 00 00 00 00 00 00

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

```

.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:

ds:	2006	00	00	00	00	00	00	00	00	ss:	0402	0000
ds:	200E	00	00	00	00	00	00	00	00	ss:	0400	0000
ds:	2016	00	00	00	00	00	00	00	00	ss:	03FE	FFFF
ds:	201E	00	00	00	00	00	00	00	00	ss:	03FC	48AD
ds:	2026	00	00	00	00	00	00	00	00	ss:	03FA	0022

AFTER:

```
ds:2006 07 56 3D 2D 15 00 00 00 •U=-8
ds:200E 00 00 00 00 00 00 00 00
ds:2016 00 00 00 00 00 00 00 00
ds:201E 00 00 00 00 00 00 00 00
ds:2026 00 00 00 00 00 00 00 00
ss:0408 0000
ss:0406 0000
ss:0404 0000
ss:0402 0000
ss:0400 0000
```

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

```
.MODEL SMALL
.STACK
    ORG 100H
.DATA
    ORG 1000H
    X DB 82H
    YMSGDB"YESITIS2OUTOF5CODE$"
    NMSGDB"NOITISNOT2OUTOF5CODE$"

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

      MOVCL,05
      MOVCH, 00
      MOV AL,X
      MOVBL,AL

      ANDAL, 0E0H
      JNZ NOTCODE

      MOV AL, BL

BACK: ROR AL, 01
      JNC SKIP
      INC CH

SKIP: DECCL
      JNZ BACK

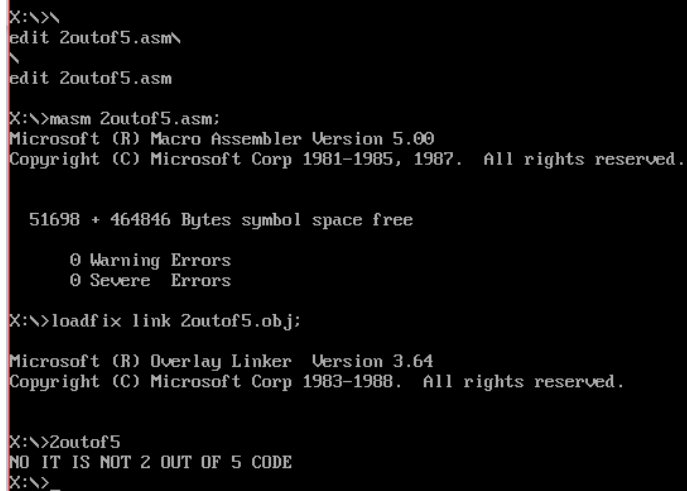
      CMP CH, 02H
      JNZ NOTCODE

      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 Zoutof5.asm\
\
edit Zoutof5.asm

X:\>masm Zoutof5.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 Zoutof5.obj;

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

X:\>Zoutof5
NO IT IS NOT 2 OUT OF 5 CODE
X:\>_
```

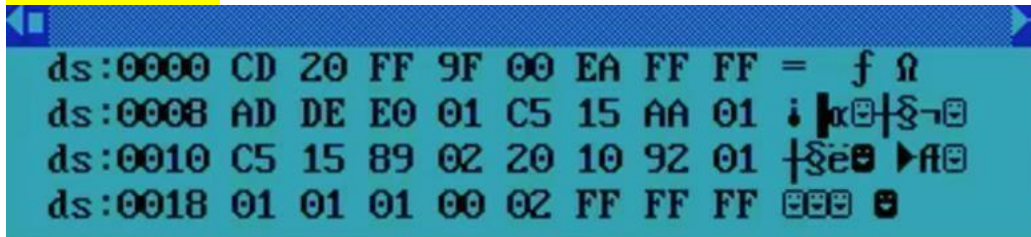
25. Program to divide a 32-bit number by 16-bit number.

```
.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

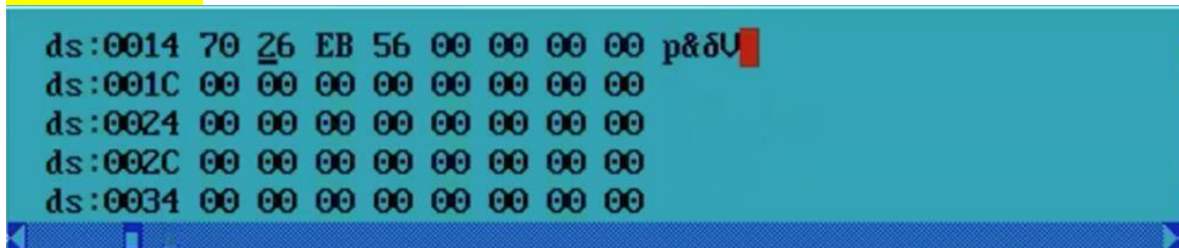
    MOV AH,4CH
    INT 21H
    END START
```

Before Execution:



Address	Hex	ASCII
ds:0000	CD 20 FF 9F 00 EA FF FF	= f Ω
ds:0008	AD DE E0 01 C5 15 AA 01	␣␣␣␣␣␣␣␣
ds:0010	C5 15 89 02 20 10 92 01	␣␣␣␣␣␣␣␣
ds:0018	01 01 01 00 02 FF FF FF	␣␣␣␣␣␣␣␣

After Execution:



Address	Hex
ds:0014	70 26 EB 56 00 00 00 00
ds:001C	00 00 00 00 00 00 00 00
ds:0024	00 00 00 00 00 00 00 00
ds:002C	00 00 00 00 00 00 00 00
ds:0034	00 00 00 00 00 00 00 00

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

ds:0000 CD 20 FF 9F 00 EA FF FF = f 0	ss:0402 0000
ds:0008 AD DE E0 01 C5 15 AA 01	ss:0400 0000
ds:0010 C5 15 89 02 20 10 92 01	ss:03FE FFFF
ds:0018 01 01 01 00 02 FF FF FF	ss:03FC 48AD
ds:0020 FF FF FF FF FF FF FF FF	ss:03FA 0011

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

AFTER EXECUTION

4BB0:100A 45 00 00 00 00 00 00 00 E	4AB1:0408 0000
4BB0:1012 00 00 00 00 00 00 00 00	4AB1:0406 0000
4BB0:101A 00 00 00 00 00 00 00 00	4AB1:0404 0000
4BB0:1022 00 00 00 00 00 00 00 00	4AB1:0402 0000
4BB0:102A 00 00 00 00 00 00 00 00	4AB1:0400 0000

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

27. Print msg using macro.

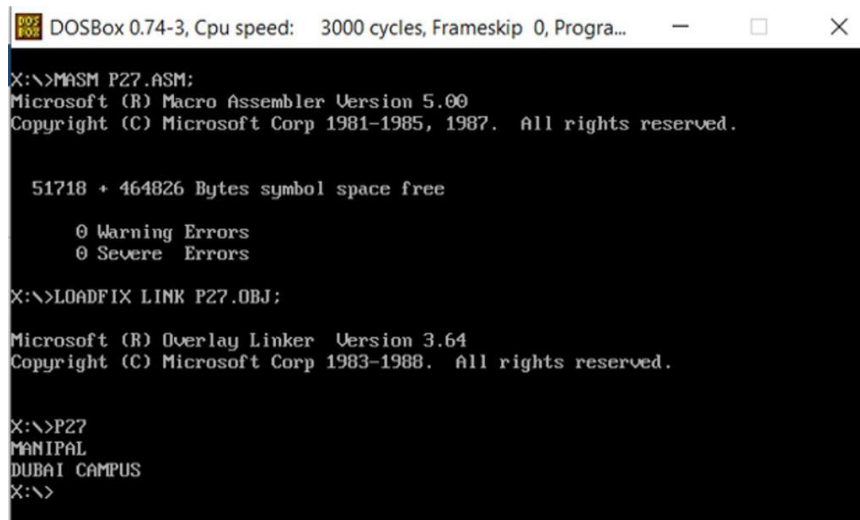
```
.MODEL SMALL
.STACK
    ORG 100H
.DATA
    MSG1 DB "MANIPAL $"
    MSG0 DB 10,13,"$"
    MSG2 DB "DUBAI CAMPUS $"

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



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
X:\>MASM P27.ASM:
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

51718 + 464826 Bytes symbol space free

0 Warning Errors
0 Severe Errors

X:\>LOADFIX LINK P27.OBJ:
Microsoft (R) Overlay Linker Version 3.64
Copyright (C) Microsoft Corp 1983-1988. All rights reserved.

X:\>P27
MANIPAL
DUBAI CAMPUS
X:\>
```

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:

ds:1000	00	00	00	00	00	00	00	00	00
ds:1008	00	00	00	00	00	00	00	00	00
ds:1010	00	00	00	00	00	00	00	00	00
ds:1018	00	00	00	00	00	00	00	00	00

ss:0402	0000
ss:0400	0000

After Execution:

48AF:1000	00	00	41	0A	00	00	00	00	00
48AF:1008	00	00	00	00	00	00	00	00	00
48AF:1010	00	00	00	00	00	00	00	00	00
48AF:1018	00	00	00	00	00	00	00	00	00

49B0:0402	0000
49B0:0400	0000

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

```
.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

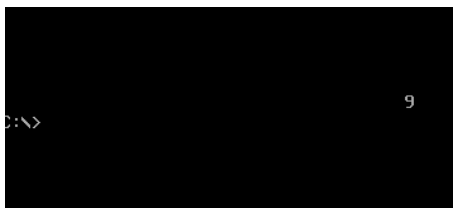
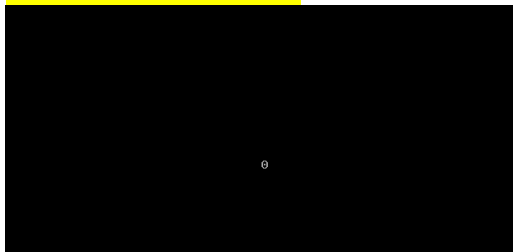
    MOV AH,02
    MOV DL,CH
    INT 21H

    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
```

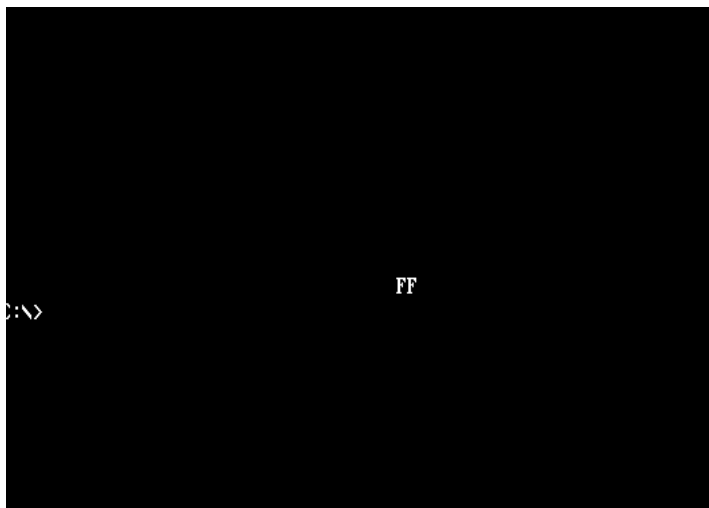
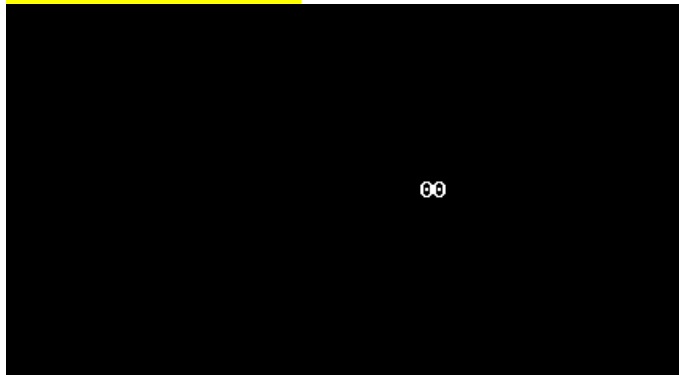
```
CENTER ENDP

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

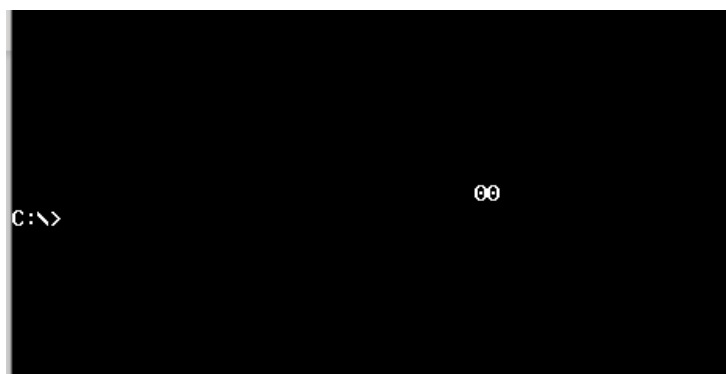
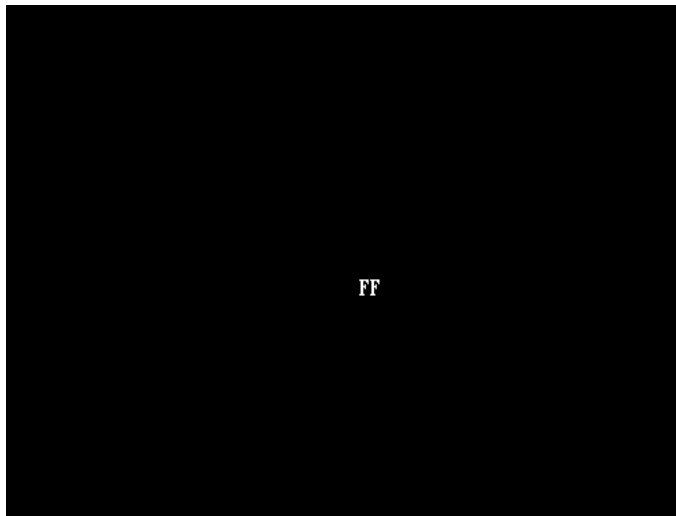
DELAY PROC NEAR
    PUSH AX
    PUSH BX
L9:  MOV BX, 0FFFFH
```

```
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 Implement BCD up counter 00 – 99 (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

    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

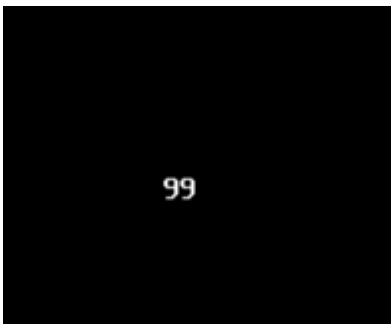
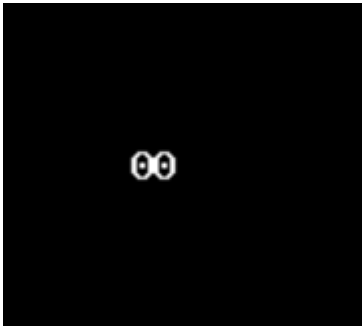
    MOV AL, X
    CMP AL, 99H
    JZ EXIT
    CALL DELAY
    MOV AL, X
    ADD AL, 01
```

```
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



99_



C:\>

00

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

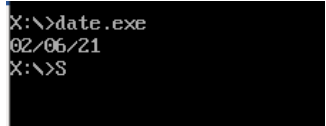
;To terminate the Program

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
```



```
X:\>date.exe
02/06/21
X:\>
```

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



ds:0024 1C 13 1E 34 20 15 33 38 LEH 13.
ds:002C 24 FF FF 00 00 00 00 00 \$

AFTER EXECUTION

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Prc

C:\>F17

THE BCD EQUIVALENT IS:65535

C:\>

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

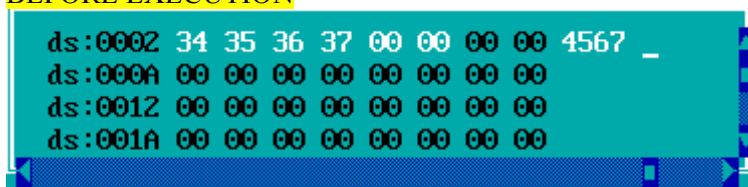
MOV CX , DIGIT
LEA SI , BCD

    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

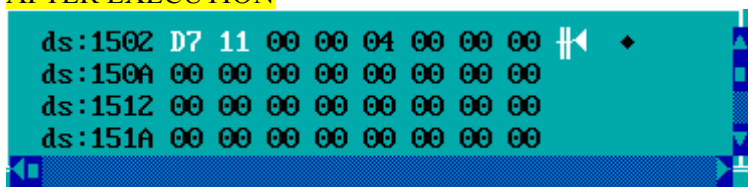
MOV AH , 4CH
INT 21H
END
```

BEFORE EXECUTION



ds:0002 34 35 36 37 00 00 00 00 4567 _
ds:000A 00 00 00 00 00 00 00 00
ds:0012 00 00 00 00 00 00 00 00
ds:001A 00 00 00 00 00 00 00 00

AFTER EXECUTION



ds:1502 D7 11 00 00 04 00 00 00
ds:150A 00 00 00 00 00 00 00 00
ds:1512 00 00 00 00 00 00 00 00
ds:151A 00 00 00 00 00 00 00 00

38. Program to Add two ASCII numbers.

```

.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

cs:0000 B8AE48	mov	ax,48AE	ax 4903	c=?
cs:0003 8ED8	mov	ds,ax	bx 0039	z=0
cs:0005 A00610	mov	al,[1006]	cx 0000	s=0
cs:0008 8A1E0710	mov	bl,[1007]	dx 0000	o=0
cs:000C 02C3	add	al,bl	si 0000	p=0
cs:000E 37	aaa		di 0000	a=?
cs:000F A20810	mov	[1008],al	bp 0000	i=?
cs:0012 B44C	mov	ah,4C	sp 0064	d=0
cs:0014 CD21	int	21	ds 48AE	
cs:0016 0000	add	[bx+si],al	es 489D	
cs:0018 0000	add	[bx+si],al	ss 49AF	
cs:001A 0000	add	[bx+si],al	cs 48AD	
cs:001C 0000	add	[bx+si],al	ip 0012	
ds:1008 03 00 00 00 00 00 F9 00 ♥				
ds:1010 00 00 00 00 00 00 00 00				
ds:1018 00 00 00 00 00 00 00 00			ss:0066 0000	
ds:1020 00 00 00 00 00 00 00 00			ss:0064 0000	

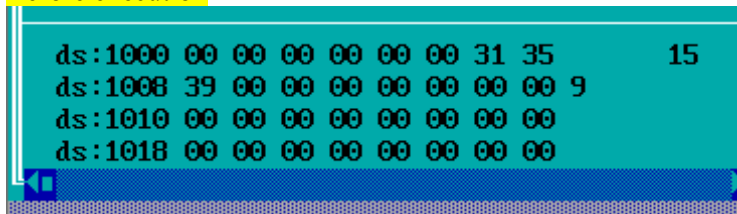
39. Program to subtract two ASCII numbers

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

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

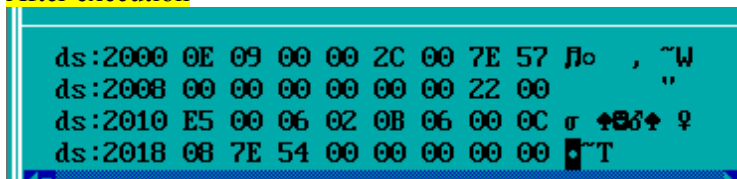
.CODE
MOV AX,@DATA
MOV DS,AX
MOV AL,NUM1
MOV BL,NUM2
sub AL,BL
AAS
MOV SUB,AL
MOV AH,4CH
INT 21H
END
```

Before execution



ds:1000	00 00 00 00 00 00 00 31 35	15
ds:1008	39 00 00 00 00 00 00 00 00	9
ds:1010	00 00 00 00 00 00 00 00 00	
ds:1018	00 00 00 00 00 00 00 00 00	

After execution



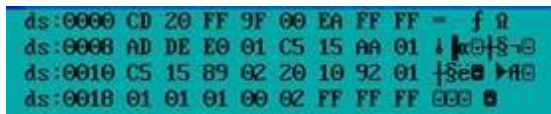
ds:2000	0E 09 00 00 2C 00 7E 57	Jo , ~W
ds:2008	00 00 00 00 00 00 22 00	"
ds:2010	E5 00 06 02 0B 06 00 0C	σ 484 9
ds:2018	08 7E 54 00 00 00 00 00	~T

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 0
ds:0008 AD DE E0 01 C5 15 AA 01 | 00 00 00 00
ds:0010 C5 15 89 02 20 10 92 01 | 00 00 00 00
ds:0018 01 01 01 00 02 FF FF FF 00 00

AFTER EXECUTION



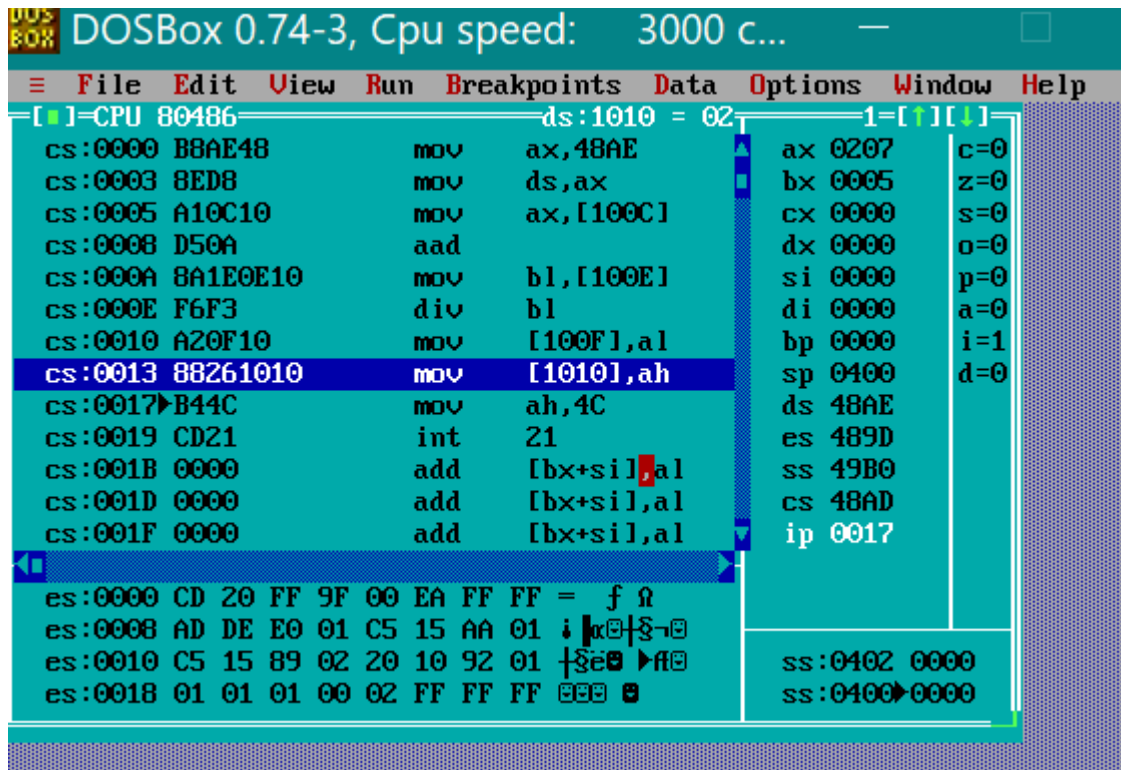
4BAE:1010 1E 00 00 00 00 00 00 00 ▲
4BAE:1018 00 00 00 00 00 00 00 00
4BAE:1020 00 00 00 00 00 00 00 00
4BAE:1028 00 00 00 00 00 00 00 00

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
```



DOSBox 0.74-3, Cpu speed: 3000 c...

File Edit View Run Breakpoints Data Options Window Help

[CPU 80486] ds:1010 = 02 1=[↑][↓]

Address	Instruction	Register/Value
cs:0000	B8AE48	mov ax,48AE
cs:0003	8ED8	mov ds,ax
cs:0005	A10C10	mov ax,[100C]
cs:0008	D50A	aad
cs:000A	8A1E0E10	mov bl,[100E]
cs:000E	F6F3	div bl
cs:0010	A20F10	mov [100F],al
cs:0013	88261010	mov [1010],ah
cs:0017	B44C	mov ah,4C
cs:0019	CD21	int 21
cs:001B	0000	add [bx+sil],al
cs:001D	0000	add [bx+sil],al
cs:001F	0000	add [bx+sil],al

ax 0207 c=0
bx 0005 z=0
cx 0000 s=0
dx 0000 o=0
si 0000 p=0
di 0000 a=0
bp 0000 i=1
sp 0400 d=0
ds 48AE
es 489D
ss 49B0
cs 48AD
ip 0017

es:0000 CD 20 FF 9F 00 EA FF FF = f 0
es:0008 AD DE E0 01 C5 15 AA 01 : 0 0 0 0 0 0 0 0
es:0010 C5 15 89 02 20 10 92 01 : 0 0 0 0 0 0 0 0
es:0018 01 01 01 00 02 FF FF FF : 0 0 0 0 0 0 0 0

ss:0402 0000
ss:0400 0000

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

48AE:1000	42 00 00 00 00 00 00 00	B	
48AE:1008	00 00 00 00 00 00 00 00		
48AE:1010	00 00 00 00 00 00 00 00		49AF:0402 0000
48AE:1018	00 00 00 00 00 00 00 00		49AF:0400 0000

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

48AE:1000	57 00 00 00 00 00 00 00	W	
48AE:1008	00 00 00 00 00 00 00 00		
48AE:1010	00 00 00 00 00 00 00 00		49AF:0402 0000
48AE:1018	00 00 00 00 00 00 00 00		49AF:0400 0000

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 00
```

```
ds:200A 00 00 00 00 00 00 00 00 00  
ds:2012 00 00 00 00 00 00 00 00 00  
ds:201A 00 00 00 00 00 00 00 00 00  
ds:2022 00 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 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 PALINDROMES"
    MSG2 DB "THE STRING IS NOT A PALINDROMES"

.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



A screenshot of a memory dump window showing four lines of memory addresses and their contents. The first line is ds:1012 containing the string 'MADAAM' followed by zeros. The other three lines (ds:101A, ds:1022, ds:102A) contain only zeros. A blue arrow points to the right at the bottom of the window.

Address	Content
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



A screenshot of a Windows command prompt showing the execution of a program. The user enters 'STR2' and the program outputs 'THE STRING IS NOT A PALINDROME'. The prompt then shows the user entering an underscore character.

```
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
```

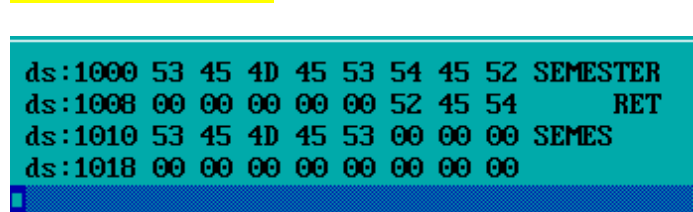
BEFORE EXECUTION



A screenshot of a memory dump window showing the state of memory before execution. The window has a blue title bar and a yellow border. The background is black with white text. The text shows four lines of memory addresses and their contents: 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, and ds:1018 00 00 00 00 00 00 00 00. A small blue cursor icon is visible at the bottom left.

```
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



A screenshot of a memory dump window showing the state of memory after execution. The window has a blue title bar and a yellow border. The background is black with white text. The text shows four lines of memory addresses and their contents: 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, and ds:1018 00 00 00 00 00 00 00 00. A small blue cursor icon is visible at the bottom left.

```
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
```

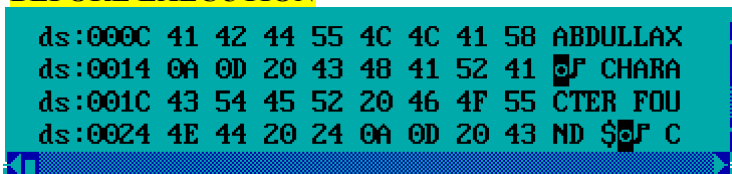
47. Program to search a character in a given string

```
.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 0A CHARA
ds:001C 43 54 45 52 20 46 4F 55 CTER FOU
ds:0024 4E 44 20 24 0A 0D 20 43 ND $0A C
```

AFTER EXECUTION



```
C:\>STR4

    CHARACTER NOT FOUND
C:\>_
```