Tutorial Problem Set (8085 based)
SOLUTION

1) Write an 8085 program to add ten numbers stored in consecutive memory address starting from 8067H and store the sixteen bit result at the end of the table.
Sol^{n}:

```
LXI H,8067H
MOV A,M
MVI B,0AH
MVI C,00H
BACK: INX H
ADD M
JNC NEXT
INR C
NEXT: DCR B
JNZ BACK
INX H
MOV M,A
INX H
MOV M,C
RST 5
```

2) Write an 8085 program to add ten numbers stored in the consecutive memory locations starting from 8081H and display the result in the two output ports. (you can assume any address for the port)
Sol^{n}:

```
LXI H,8067H
MOV A,M
MVI B,10H
MVI C,00H
BACK: INX H
ADD M
JNC NEXT
INR C
NEXT: DCR B
JNZ BACK
OUT 43
MOV D,A
MOV A,C
OUT 40
MOV A,D
OUT 41
RST 5
```

3) Add all the positive numbers stored in the memory location 80A1H to 80AAH. Display the 16-bit result in any ports.
Sol^{n}:

```
LXI H,80A1H
MVI A,00H
MVI B,0AH
MVI C,00H
MVI D,00H
BACK: MOV A,M
ANI 80H
JNZ NEXT
MOV A,M
ADD D
MOV D,A
JNC LABEL
INR C
LABEL: NOP
NEXT: INX H
DCR B
JNZ BACK
MOV A,C
OUT 40
MOV A,D
OUT 41
RST 5
```

4) Add all the numbers with bit D5 and D3, 1 and 0 respectively, stored in the memory location 90B1H to 90BAH. Display the 16-bit result in any ports.
Sol^{n}:

```
LXI H,80B1H
MVI A,00H
MVI B,0AH
MVI C,00H
MVI D,00H
START: MOV A,M
MVI E,02H
BACK: RLC
DCR E
JNZ BACK
MOV E,A
ANI 80H
JZ JUMP
MOV A,E
RLC
RLC
ANI 80H
JNZ NEXT
MOV A,M
ADD D
MOV D,A
JNC GO
INR C
GO: NOP
JUMP: NOP
NEXT: INX H
DCR B
JNZ START
RST 5
```

5) There are two table of data stored at 80A1H and 80B1H having ten data each. Write a program to store the data in the first table to third table starting from address 80C1H if the corresponding data in the

first table is greater than the second table else store FFH in the third table.

Sol^{n}:

```
LXI B,80A1H
LXI H,80B1H
LXI D,80C1H
BACK: LDAX B
STAX D
CMP M
JNC NEXT
MVI A,0FFH
STAX D
NEXT: INX H
INX B
INX D
LDAX B
ADI 00H
JNZ BACK
RST 5
```

6) Sixteen bit data are stored in two tables starting at 8050H and 8070H, ten data in each table. Add corresponding data and store it in the third table starting at 8090H. (Never forget the reverse order convention in storing the 16-bit data)

Sol^{n}:

```
LXI B,8050H
LXI H,8070H
LXI D,8090H
MVI A,0AH
STA 8090H
BACK: LDAX B
ADD M
STAX D
INX B
INX H
INX D
LDAX B
ADC M
STAX D
INX D
JNC NEXT
MVI A,01H
STAX D
NEXT: INX B
INX H
INX D
LDA 8090H
DCR A
STA 8090H
JNZ BACK
RST 5
```

7) Add sixteen bit data stored in two tables and store the result in the corresponding index of the third table if the result in the corresponding index of the third table only if the result is greater than 00FFH, else store 0000H (you can assume any address for the tables)

Sol^{n}:

```
LXI B,8050H
LXI H,8070H
LXI D,8090H
BACK: LDAX B
ADD M
STAX D
INX B
INX H
INX D
LDAX B
ADC M
STAX D
NEXT: INX B
INX H
INX D
LDAX B
ADI 00H
JNZ BACK
LXI D,8091H
MVI B,05
LABEL: LDAX D
CPI 00H
JNC JUMP
MVI A,00H
STAX D
DCX D
MVI A,00H
STAX D
INX D
JUMP: INX D
INX D
DCR B
JNZ LABEL
RST 5
```

8) In two tables 16-bit data are stored, each table having ten numbers each. Subtract the data from one table to other and store the result in the third table.

Sol^{n}:

```
LXI B,8050H
LXI H,8070H
LXI D,8090H
BACK: LDAX B
SUB M
STAX D
INX B
INX H
INX D
LDAX B
SBB M
STAX D
INX D
JNC NEXT
```

```
MVI A,01H
STAX D
NEXT: INX B
INX H
INX D
LDAX B
ADI 00H
JNZ BACK
RST 5
```

9) Subtract ten 16-bit data stored in one table from the other. Store the result in the third table if the result is positive else store 00.
Sol^n:

```
LXI B,8050H
LXI H,8070H
LXI D,8090H
BACK: LDAX B
SUB M
STAX D
INX B
INX H
INX D
LDAX B
SBB M
STAX D
NEXT: INX B
INX H
INX D
LDAX B
ADI 00H
JNZ BACK
LXI D,8091H
MVI B,05
LABEL: LDAX D
ANI 80H
JZ JUMP
MVI A,00H
STAX D
DCX D
MVI A,00H
STAX D
INX D
JUMP: INX D
INX D
DCR B
JNZ LABEL
RST 5
```

10) Transfer ten data, which has bit D5 and D0, 0 and 1 respectively from A430H to A440H, else store 00 instead of transformation.
Sol^n:

```
LXI H,0A430H
LXI D,0A440H
MVI B,0AH
BACK: MVI A,00H
STAX D
MOV A,M
MVI C,02H
LABEL: RLC
DCR C
JNZ LABEL
ANI 80H
JNZ JUMP
MOV A,M
RRC
ANI 80H
JZ NEXT
MOV A,M
STAX D
JUMP: NOP
NEXT: INX H
INX D
DCR B
JNZ BACK
RST 5
```

11) Transfer ten data with even parity from location 9270H to 9280H, else transfer the data by clearing bit D7 and setting bit D2.
Sol^n:

```
LXI H,9500H
MVI B,0AH
MVI C,00H
MVI D,00H
MOV A,M
BACK: INX H
INX H
ADD M
JNC JUMP
INR C
JUMP: DCR B
JNZ BACK
MOV E,A
LXI H,9501H
MVI B,0AH
MOV A,M
BACKA: INX H
INX H
ADD M
JNC JUMPA
INR D
JUMPA: DCR B
JNZ BACKA
ADD C
JNC GO
INR D
GO: INX H
MOV M,E
INX H
MOV M,A
```

```
INX H
MOV M,D
RST 5
```

12) Data is stored from 8040H to 8050H. Transfer the data to other location in reverse order.

Sol^n:

```
LXI H,8040H
LXI D,805AH
MVI A,00H
MVI B,10H
BACK: MOV A,M
STAX D
INX H
DCX D
DCR B
JNZ BACK
RST 5
```

13) Add ten 16-bit numbers stored in a table at 9500H and store the 24-bit result at the end of the table.

Sol^n:

```
LXI H,9500H
MVI B,0AH
MVI C,00H
MVI D,00H
MOV A,M
BACK: INX H
INX H
ADD M
JNC JUMP
INR C
JUMP: DCR B
JNZ BACK
MOV E,A
LXI H,9501H
MVI B,0AH
MOV A,M
BACKA: INX H
INX H
ADD M
JNC JUMPA
INR D
JUMPA: DCR B
JNZ BACKA
ADD C
JNC GO
INR D
GO: INX H
MOV M,E
INX H
MOV M,A
INX H
MOV M,D
RST 5
```

14) Data is stored from 8050H to 805AH. Insert 5 data after 8055H taking from 8040H, but do not lose the previous content.

Sol^n:

```
LXI H,8056H
LXI B,8030H
MVI D,05H
LABEL: MOV A,M
STAX B
INX H
INX B
DCR D
JNZ LABEL
LXI B,8040H
LXI H,8056H
MVI D,05H
BACK: STAX B
MOV M,A
INX H
INX B
DCR D
JNZ BACK
LXI B,8030H
MVI D,05H
PAST: STAX B
MOV M,A
INX H
INX B
DCR D
JNZ PAST
RST 5
```

15) Ten data are stored from 8080H. Transfer the first 5 numbers at the end of the second table and the rest at the starting of it.

Sol^n:

```
LXI H,8080H
LXI B,819AH
MVI D,05H
LABEL: MOV A,M
STAX B
INX H
INX B
DCR D
JNZ LABEL
LXI B,818BH
MVI D,05H
BACK: MOV A,M
STAX B
INX H
INX B
DCR D
JNZ BACK
RST 5
```

16) Transfer data from 9050H to 9060H only if data is between 30H and 70H else store 00H in the next table.

Sol^n:

```
LXI H,9050H
LXI B,9060H
MVI D,0AH
BACK: MVI A,00H
STAX B
MOV A,M
CPI 30H
JC NEXT
CPI 70H
JNC LABEL
STAX B
NEXT: NOP
LABEL: INX H
INX B
DCR D
JNZ BACK
RST 5
```

17) Transfer data from 8250H to 8260H if the number is less than 50H and greater than 80H else store 00H in the destination table.

Sol^n:

```
LXI H,8250H
LXI B,8260H
MVI D,0AH
BACK: STAX B
MOV A,M
CPI 50H
JC NEXT
CPI 80H
JNC LABEL
 MVI A,00H
STAX B
NEXT: NOP
LABEL: INX H
INX B
DCR D
JNZ BACK
RST 5
```

18) Write a program to count the number of ones of table of ten sixteen bit numbers at 8240H and store the count of one’s in corresponding location of a table at 8260H

Sol^n:

```
LXI H,8240H
LXI B,8260H
MVI E,02H
BACKB: NOP
BACKC: NOP
BACKA: MVI D,08H
MOV A,M
BACK: ANI 80H
JZ NEXT
LDAX B
INR A
STAX B
NEXT: MOV A,M
RLC
MOV M,A
DCR D
JNZ BACK
INX H
MVI D,08H
MOV A,M
BACKD: ANI 80H
JZ NEXTA
LDAX B
INR A
STAX B
NEXTA: MOV A,M
RLC
MOV M,A
DCR D
JNZ BACKD
INX H
INX B
DCR E
JNZ BACKC
RST 5
```

TUTORIAL SOLUTION OF 8086

1. Write an assembly language program to add all the elements of a table, which are between 50 and 150 only. Display the result as the decimal value.

Sol^n:

```
.MODEL SMALL
.STACK 64
.DATA
        COUNT EQU 05
        DATA1 DB 125,235,197,91,48
        SUM DW ?
.CODE
        MOV AX,@DATA
        MOV DS,AX
         MOV CX,COUNT
        MOV SI,OFFSET DATA1
        MOV AX,0000H
  BACK:MOV BL,[SI]
        CMP BL,50;comparision
        JB OVER
        CMP BL,150;comparision
        JA OVER
        ADD AL,[SI];addition
        JNC OVER
        INC AH
  OVER: INC SI
        DEC CX
        JNZ BACK
        ;decimal display
        MOV BX,0000
        MOV CX,0AH
   DCE:MOV DX,0000H
        DIV CX
        ADD DX,30H
        PUSH DX
        INC BX
        CMP AX,0000
        JA DCE
        MOV CX,BX
        MOV AH,02
  SHOW:POP DX
        INT 21H
        LOOP SHOW
        MOV AH,4CH
        INT 21H
END
```

2. A table of numbers is stored in memory. Write an assembly language program to add numbers from the table, which are between 30 and 100. Display the result in hex format.

Sol^n:

```
.MODEL SMALL
.STACK 64
.DATA
        COUNT EQU 05
        DATA1 DB 125,235,197,91,48
        SUM DW ?
        XYZ DB ?
.CODE
        MOV AX,@DATA
        MOV DS,AX
    MOV CX,COUNT
        MOV SI,OFFSET DATA1
        MOV AX,0000H
  BACK:MOV BL,[SI]
        CMP BL,30;comparision
        JB OVER
        CMP BL,100;comparision
        JA OVER
        ADD AL,[SI] ];addition
        JNC OVER
        INC AH
  OVER: INC SI
        DEC CX
        JNZ BACK
        ;decimal display
        MOV BX,0000
        MOV CX,10H
   DCE:MOV DX,0000H
        DIV CX
        CMP DX,000AH
        JA L1
        ADD DX,30H
        JMP L2
   L1:ADD DX,37H
   L2:PUSH DX
```

```
        INC BX
        CMP AX,0000
        JA DCE
        MOV CX,BX
        MOV AH,02
  SHOW:POP DX
        INT 21H
        LOOP SHOW
        CALL EXIT
        EXIT PROC
        MOV AH,4CH
        INT 21H
        RET
        EXIT ENDP
END
```

3. Write an assembly language program to get text input and display it on the center of a clear screen.

Sol^{n}:

```
.MODEL SMALL
.STACK 100
.DATA
        MSG DB ' ',10 DUP (?)
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV AH,0AH
        MOV CX,06H
        LEA DX,MSG
        INT 21H;string input
        MOV SI,OFFSET MSG
        MOV [SI+0008],'$'
        CALL VIDEO_MODE
        CALL CLEAR_SCREEN
        CALL SET_CURSOR
        CALL DISPLAY
        CALL EXIT_PROGRAM
        VIDEO_MODE PROC
        MOV AH,00H
        MOV AL,00H
        INT 10H;set video mode
        RET
        VIDEO_MODE ENDP
        CLEAR_SCREEN PROC
        MOV AH,06H
        MOV AL,00H
        MOV BH,07H
        MOV CX,0000H
        MOV DX,1827H
        INT 10H;clear whole screen
        RET
        CLEAR_SCREEN ENDP
        SET_CURSOR PROC
        MOV AH,02H
        MOV DH,12
        MOV DL,20
        MOV BH,00
        INT 10H;set cursor at centre
        RET
        SET_CURSOR ENDP
        DISPLAY PROC
        LEA DX,MSG
        ADD DX,02H
        MOV AH,09H
        INT 21H;dispay string
        RET
        DISPLAY ENDP
        EXIT_PROGRAM PROC
        MOV AH,4CH
        INT 21H
        RET
        EXIT_PROGRAM
END
```

4. Write an assembly language program to accept string input and convert to upper case if it has lower case letters.

Sol^{n}:

```
.MODEL SMALL
.STACK 100
.DATA
        MSG DB ' ',10 DUP (?)
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV AH,0AH
        MOV CX,06H
        LEA DX,MSG
        INT 21H;string input
        MOV SI,OFFSET MSG
        MOV [SI+0008],'$'
        MOV CX,0010
   BACK:MOV AL,[SI]
        CMP AL,61H
        JB OVER
        CMP AL,7AH
        JA OVER
        AND AL,11011111B;convert  to uppercase
```

```
OVER:MOV [SI],AL
        INC SI
        LOOP BACK
        CALL VIDEO_MODE
        CALL CLEAR_SCREEN
        CALL SET_CURSOR
        CALL DISPLAY
        CALL EXIT_PROGRAM
        VIDEO_MODE PROC
        MOV AH,00H
        MOV AL,00H
        INT 10H;set video mode
        RET
        VIDEO_MODE ENDP

        CLEAR_SCREEN PROC
        MOV AH,06H
        MOV AL,00H
        MOV BH,07H
        MOV CX,0000H
        MOV DX,1827H
        INT 10H;clear whole screen
        RET
        CLEAR_SCREEN ENDP
        SET_CURSOR PROC
        MOV AH,02H
        MOV DH,12
        MOV DL,20
        MOV BH,00
        INT 10H;set cursor at centre
        RET
        SET_CURSOR ENDP
        DISPLAY PROC
        LEA DX,MSG
        ADD DX,02H
        MOV AH,09H
        INT 21H;display string
        RET
        DISPLAY ENDP
        EXIT_PROGRAM PROC
        MOV AH,4CH
        INT 21H
        RET
        EXIT_PROGRAM ENDP
        END
```

5. Write an assembly language program to get input and display on location 10,20 on the screen

Sol^{n}:

```
.MODEL SMALL
.STACK 100
.DATA
        MSG DB ' ',10 DUP (?)
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV AH,0AH
        MOV CX,06H
        LEA DX,MSG
        INT 21H;input string
        MOV SI,OFFSET MSG
        MOV [SI+0008],'$'
        CALL VIDEO_MODE
        CALL CLEAR_SCREEN
        CALL SET_CURSOR
        CALL DISPLAY
        CALL EXIT_PROGRAM
        VIDEO_MODE PROC
        MOV AH,00H
        MOV AL,00H
        INT 10H;set video mode
        RET
        VIDEO_MODE ENDP
        CLEAR_SCREEN PROC
        MOV AH,06H
        MOV AL,00H
        MOV BH,07H
        MOV CX,0000H
        MOV DX,1827H
        INT 10H;clear whole screen
        RET
        CLEAR_SCREEN ENDP
        SET_CURSOR PROC
        MOV AH,02H
        MOV DH,20
        MOV DL,10
        MOV BH,00
        INT 10H;set cursor at centre
        RET
        SET_CURSOR ENDP
        DISPLAY PROC
        LEA DX,MSG
        ADD DX,02
        MOV AH,09H
        INT 21H;dispay string
        RET
        DISPLAY ENDP
        EXIT_PROGRAM PROC
        MOV AH,4CH
        INT 21H
        RET
        EXIT_PROGRAM ENDP
        END
```

6. Write an assembly language program to convert the text stored in memory to upper case only if the characters are found in lower case. Display the converted text in the screen.

Sol^{n}:

```
.MODEL SMALL
.STACK 64
.DATA
        DATA DB 'mY NamE is KiShor','$'
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV SI,OFFSET DATA
        MOV CX,0017
  BACK:MOV AL,[SI]
        CMP AL,61H
        JB OVER
        CMP AL,7AH
        JA OVER
        AND AL,11011111B;convert to upper case
   OVER:MOV [SI],AL
        INC SI
        LOOP BACK
        CALL VIDEO_MODE
        CALL CLEAR_SCREEN
        CALL SET_CURSOR
        CALL DISPLAY
        CALL EXIT_PROGRAM
        VIDEO_MODE PROC
        MOV AH,00H
        MOV AL,00H
        INT 10H;set video mode
        RET
        VIDEO_MODE ENDP
        CLEAR_SCREEN PROC
        MOV AH,06H
        MOV AL,00H
        MOV BH,07H
        MOV CX,0000H
        MOV DX,1827H
        INT 10H;clear whole screen
        RET
        CLEAR_SCREEN ENDP
        SET_CURSOR PROC
        MOV AH,02H
        MOV DH,12
        MOV DL,20
        MOV BH,00
        INT 10H;set cursor at centre
        RET
        SET_CURSOR ENDP
        DISPLAY PROC
        LEA DX,DATA
        MOV AH,09H
        INT 21H;dispay string
        RET
        DISPLAY ENDP
        EXIT_PROGRAM PROC
        MOV AH,4CH
        INT 21H
        RET
        EXIT_PROGRAM ENDP
        END
```

7. Write an assembly language program to convert the text stored in the memory to lower case if the characters are in upper case. Display the result text in the screen

Sol^{n}:

```
.MODEL SMALL
.STACK 64
.DATA
        DATA DB 'mY NamE is KiShor','$'
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV SI,OFFSET DATA
        MOV CX,0017
  BACK:MOV AL,[SI]
        CMP AL,41H
        JB OVER
        CMP AL,5AH
        JA OVER
        OR AL,00100000B;convert to lowercase
   OVER:MOV [SI],AL
        INC SI
        LOOP BACK
        CALL VIDEO_MODE
        CALL CLEAR_SCREEN
        CALL SET_CURSOR
        CALL DISPLAY
        CALL EXIT_PROGRAM
        VIDEO_MODE PROC
        MOV AH,00H
        MOV AL,00H
        INT 10H;set video mode
        RET
        VIDEO_MODE ENDP
        CLEAR_SCREEN PROC
        MOV AH,06H
        MOV AL,00H
        MOV BH,07H
        MOV CX,0000H
        MOV DX,1827H
        INT 10H;clear whole screen
        RET
        CLEAR_SCREEN ENDP
        SET_CURSOR PROC
        MOV AH,02H
        MOV DH,12
        MOV DL,20
        MOV BH,00
        INT 10H;set cursor at centre
        RET
        SET_CURSOR ENDP
        DISPLAY PROC
        LEA DX,DATA
        MOV AH,09H
        INT 21H;display string
        RET
        DISPLAY ENDP
        EXIT_PROGRAM PROC
        MOV AH,4CH
        INT 21H
        RET
        EXIT_PROGRAM ENDP
        END
```

8. Write a program to add the sequence 1+3+4+... up to 100 steps display the result in hexadecimal format.

Sol^n:

```
.MODEL SMALL
.STACK 64
.DATA
        SUM DW 0000
        CARRY DW 0000
        XYZ DB ?
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV CX,0064H
        MOV AX,01
        MOV BX,03
        ADD AX,BX
        MOV SUM,AX
 BACK:MOV DX,AX
        MOV AX,BX
        MOV BX,DX
        ADD SUM,BX
        JNC NEXT
        INC CARRY
 NEXT:ADD AX,BX;addition of series
        LOOP BACK
        MOV AX,CARRY
        CALL DISP
        MOV AX,SUM
        CALL DISP
        CALL EXIT
        DISP PROC;hex display
        MOV BX,0000
        MOV CX,10H
   DCE:MOV DX,0000H
        DIV CX
        CMP DX,000AH
        JA L1
        ADD DX,30H
        JMP L2
   L1:ADD DX,37H
   L2:PUSH DX
        INC BX
        CMP AX,0000
        JA DCE
        MOV CX,BX
        MOV AH,02
  SHOW:POP DX
        INT 21H
        LOOP SHOW
        RET
        DISP ENDP
        EXIT PROC
        MOV AH,4CH
        INT 21H
        RET
        EXIT ENDP
END
```

9. Write a program to add the sequence 1+3+4+... up to the desired steps entered by the user and display the result in decimal format. Assume user enters numbers from 1 to 9.

Sol^n:

```
.MODEL SMALL
.STACK 64
.DATA
        SUM DW 0000
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV AH,01H
        INT 21H
        SUB AL,30H
        MOV CH,00H
        MOV CL,AL
        MOV AH,02H
        MOV DL,20H
        INT 21H
        MOV AX,01
        MOV BX,03
        ADD AX,BX
        MOV SUM,AX
 BACK:MOV DX,AX
        MOV AX,BX
        MOV BX,DX
        ADD SUM,BX
        ADD AX,BX;addition of series
        LOOP BACK
        MOV AX,SUM;decimal display
   MOV BX,0000
        MOV CX,0AH
   DCE:MOV DX,0000H
        DIV CX
        ADD DX,30H
        PUSH DX
        INC BX
        CMP AX,0000
        JA DCE
        MOV CX,BX
        MOV AH,02
  SHOW:POP DX
        INT 21H
        LOOP SHOW
        MOV AH,4C
        INT 21H
END
```

10. Write an assembly language program to display graphical ASCII characters from 32 to 127 on a defined window (5, 10 and 20, 70) with white on blue attribute.

Sol^n:

```
.MODEL SMALL
.STACK 100
.DATA
.CODE
        MOV AX,@DATA
        MOV DS,AX
        CALL VIDEO_MODE
        CALL CLEAR_SCREEN
        CALL SET_CURSOR
        CALL DISPLAY
        CALL EXIT_PROGRAM
        VIDEO_MODE PROC
        MOV AH,00H
        MOV AL,00H
        INT 10H;set video mode
        RET
        VIDEO_MODE ENDP
        CLEAR_SCREEN PROC
        MOV AH,06H
        MOV AL,00H
        MOV BH,0BFH
        MOV CX,0000H
        MOV DX,1827H
        INT 10H;clear whole screen
        RET
        CLEAR_SCREEN ENDP
        SET_CURSOR PROC
        MOV AH,02H
        MOV DH,10
        MOV DL,05
        MOV BH,00
        INT 10H;set cursor at centre
        RET
        SET_CURSOR ENDP
        DISPLAY PROC;display characters
        MOV CX,5FH
        MOV BH,32
  LABEL1:MOV AH,02H
        MOV DL,BH
        INT 21H
        INC BH
        LOOP LABEL1
        RET
        DISPLAY ENDP
        EXIT_PROGRAM PROC
        MOV AH,4CH
        INT 21H
        RET
        EXIT_PROGRAM ENDP
        END
```

11. You have an array of data in one table. Change each element to decimal ASCII and store it in the next table. Display the final result in the clear screen.
Sol^n:

```
.MODEL SMALL
.STACK 100
.DATA
        COUNT DB ?
        DATA1 DB 'kishor'
        DATA2 DW 5 DUP(?)
        SUM DW ?
        TEMP DW ?
        TEMP1 DW ?
        HUN DB 00
        TEN DB 00
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV AH,06H;clear screen
        MOV AL,00H
        MOV BH,07H
        MOV CX,0000H
        MOV DX,1827H
        INT 10H;dos function
        LEA SI,DATA1
        LEA DI,DATA2;for storing BCD eqvt
        MOV DX,0000H
        MOV COUNT,05H
 BACK:MOV HUN,00
        MOV TEN,00
        MOV AH,00H
        MOV AL,[SI]
  L1:CMP AX,64H
        JB NEXT1
        INC HUN
        SUB AX,64H
        JMP L1
 NEXT1:CMP AX,0AH
        JB NEXT2
        INC TEN
        SUB AX,0AH
        JMP NEXT1
NEXT2:MOV TEMP1,AX
        MOV AX,0001H
        MUL HUN
        MOV BX,100H
        MUL BX
        MOV TEMP,AX
        MOV AX,0001H
        MUL TEN
        MOV BX,10H
        MUL BX
        ADD AX,TEMP
        ADD AX,TEMP1
        MOV [DI],AX;storing BCD value of data
        INC SI
        INC DI
        INC DI
        DEC COUNT
        JNZ BACK
        LEA SI,DATA1
        MOV HUN,05
BACK2:MOV AL,[SI] ;display BCD value of data
        MOV AH,00
        MOV BX,0000
        MOV CX,0AH
 DCE:MOV DX,0000H
        DIV CX
        ADD DX,30H
        PUSH DX
        INC BX
        CMP AX,0000
        JA DCE
        MOV CX,BX
        MOV AH,02
SHOW:POP DX
        INT 21H
        LOOP SHOW
        MOV AH,02
        MOV DL,20H
        INT 21H
        INC SI
        DEC HUN
        JNZ BACK2
        MOV AH,4CH
        INT 21H
        END
```

12. Write an assembly language program to count the number of vowels in a string entered by the user. Display the result in decimal format.

Sol^{n}:

```
.MODEL SMALL
.STACK 64
.DATA
        MSG DB 10 DUP(?)
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV AH,0AH
        MOV CX,06H
        LEA DX,MSG
        INT 21H;string input
        MOV SI,OFFSET MSG
        MOV CX,0006H
        MOV BX,0000H
        ADD SI,02H
   BACK:MOV AH,[SI]
        CMP AH,61H;compairing vowel
        JE NEXT
        CMP AH,65H;compairing vowel
        JE NEXT
        CMP AH,69H;compairing vowel
        JE NEXT
        CMP AH,6FH;compairing vowel
        JE NEXT
        CMP AH,75H;compairing vowel
        JE NEXT
        JMP GOTO1
   NEXT:INC BX;count vowel
   GOTO1:INC SI
        LOOP BACK
        MOV AH,02
        MOV DL,0AH
        INT 21H;print space
        ;Display  count in BCD
        MOV AX,BX
     MOV BX,0000
        MOV CX,0AH
   DCE:MOV DX,0000H
        DIV CX
        ADD DX,30H
        PUSH DX
        INC BX
        CMP AX,0000
        JA DCE
        MOV CX,BX
        MOV AH,02
  SHOW:POP DX
        INT 21H
        LOOP SHOW
        MOV AH,4CH
        INT 21H
END
```

13. Write an assembly language program to convert the vowels to uppercase from a string entered by the user.

Sol^{n}:

```
.MODEL SMALL
.STACK 64
.DATA
        MSG DB 10 DUP(?)
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV AH,0AH
        MOV CX,06H
        LEA DX,MSG
        INT 21H;input string
        MOV SI,OFFSET MSG
        MOV CX,0006H
        MOV BX,0000H
        ADD SI,02H
   BACK:MOV AH,[SI]
        CMP AH,61H;compairing vowel
        JE NEXT
        CMP AH,65H;compairing vowel
        JE NEXT
        CMP AH,69H;compairing vowel
        JE NEXT
        CMP AH,6FH;compairing vowel
        JE NEXT
        CMP AH,75H;compairing vowel
        JE NEXT
        JMP GOTO1
   NEXT:AND AH,11011111B;convert to uppercase
        MOV [SI],AH
   GOTO1:INC SI
        LOOP BACK
        MOV [SI],'$'
        CALL VIDEO_MODE
        CALL CLEAR_SCREEN
        CALL SET_CURSOR
        CALL DISPLAY
        CALL EXIT_PROGRAM
        VIDEO_MODE PROC
        MOV AH,00H
        MOV AL,00H
        INT 10H;set video mode
        RET
        VIDEO_MODE ENDP
        CLEAR_SCREEN PROC
        MOV AH,06H
        MOV AL,00H
        MOV BH,07H
        MOV CX,0000H
        MOV DX,1827H
        INT 10H;clear screen
        RET
        CLEAR_SCREEN ENDP
        SET_CURSOR PROC
        MOV AH,02H
        MOV DH,12
        MOV DL,20
        MOV BH,00
        INT 10H;set cursor at centre
        RET
        SET_CURSOR ENDP
        DISPLAY PROC
        LEA DX,MSG
        ADD DX,02H
        MOV AH,09H
        INT 21H;display string
        RET
        DISPLAY ENDP
        EXIT_PROGRAM PROC
        MOV AH,4CH
        INT 21H
        RET
        EXIT_PROGRAM ENDP
        END
```

14. Write an assembly language program to get string input from the user convert it to capital case display the attributed string at the center of the defined window (2,10 to 22,70).

Sol^{n}:

```
.MODEL SMALL
.STACK 64
.DATA
        MSG DB 10 DUP(?)
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV AH,0AH
        MOV CX,06H
        LEA DX,MSG
        INT 21H;string input
        MOV SI,OFFSET MSG
        ADD SI,02H
        MOV CX,0006H
 BACK:MOV AL,[SI]
        CMP AL,61H
        JB OVER
        CMP AL,7AH
        JA OVER
        AND AL,11011111B;convert to uppercase
OVER:MOV [SI],AL
        INC SI
        LOOP BACK
        MOV [SI],'$'
        CALL VIDEO_MODE
        CALL CLEAR_SCREEN
        CALL SET_CURSOR
        CALL DISPLAY
        CALL EXIT_PROGRAM
        VIDEO_MODE PROC
        MOV AH,00H
        MOV AL,00H
        INT 10H;set video mode
        RET
        VIDEO_MODE ENDP
        CLEAR_SCREEN PROC
        MOV AH,06H
        MOV AL,00H
        MOV BH,07H
        MOV CX,0000H
        MOV DX,1827H
        INT 10H;clear screen
        RET
        CLEAR_SCREEN ENDP
        SET_CURSOR PROC
        MOV AH,02H
        MOV DH,12
        MOV DL,20
        MOV BH,00
        INT 10H;set cursor at co ordinate
        RET
        SET_CURSOR ENDP
        DISPLAY PROC
        LEA DX,MSG
        ADD DX,02H
        MOV AH,09H
        INT 21H;display string
        RET
        DISPLAY ENDP
        EXIT_PROGRAM PROC
        MOV AH,4CH
        INT 21H
        RET
        EXIT_PROGRAM ENDP
        END
```

15. Write an assembly language program to get string input from the user convert it to lower case display the attributed string at the lower left corner of the defined window (3, 10 to 21, 10).

Sol^n:

```
.MODEL SMALL
.STACK 64
.DATA
        MSG DB 10 DUP(?)
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV AH,0AH
        MOV CX,06H
        LEA DX,MSG
        INT 21H;input string
        MOV SI,OFFSET MSG
        ADD SI,02H
        MOV CX,0006H
  BACK:MOV AL,[SI]
        CMP AL,41H
        JB OVER
        CMP AL,5AH
        JA OVER
        OR AL,00100000B;convert to lowercase
 OVER:MOV [SI],AL
        INC SI
        LOOP BACK
        MOV [SI],'$'
        CALL VIDEO_MODE
        CALL CLEAR_SCREEN
        CALL SET_CURSOR
        CALL DISPLAY
        CALL EXIT_PROGRAM
        VIDEO_MODE PROC
        MOV AH,00H
        MOV AL,00H
        INT 10H;set video mode
        RET
        VIDEO_MODE ENDP
        CLEAR_SCREEN PROC
        MOV AH,06H
        MOV AL,00H
        MOV BH,07H
        MOV CX,0000H
        MOV DX,1827H
        INT 10H;clear screen
        RET
        CLEAR_SCREEN ENDP
        SET_CURSOR PROC
        MOV AH,02H
        MOV DH,10
        MOV DL,02
        MOV BH,00
        INT 10H;set cursor at co ordinate
        RET
        SET_CURSOR ENDP
        DISPLAY PROC
        LEA DX,MSG
        ADD DX,02H
        MOV AH,09H
        INT 21H;display string
        RET
        DISPLAY ENDP
        EXIT_PROGRAM PROC
        MOV AH,4CH
        INT 21H
        RET
        EXIT_PROGRAM ENDP
        END
```

16. Write an assembly language program that takes a string input from user and clear the screen and move the string from right edge of the screen to left edge. The movement should be noticeable.

Sol^n:

```
.MODEL SMALL
.STACK 200
.DATA
        MSG DB 'KISHOR$'
        TEMP1 DW 65535;for delay
        TEMP2 DW 65535;for delay
        TEMP3 DW 65535;for delay
        TEMP4 DW 65535;for delay
        TEMP5 DW 65535;for delay
        LOC DW 0021H
.CODE
        MOV AX,@DATA
        MOV DS,AX
        CALL VIDEO_MODE
        MOV CX,25
  BACK:CALL CLEAR_SCREEN
        MOV DX,LOC
        CALL SET_CURSOR
        CALL DISPLAY
   BACK3:DEC TEMP3;delay loop
        JNZ BACK3
  BACK1:DEC TEMP1;delay loop
        JNZ BACK1
  BACK2:DEC TEMP2;delay loop
        JNZ BACK2
  BACK4:DEC TEMP4;delay loop
        JNZ BACK4
  BACK5:DEC TEMP5;delay loop
        JNZ BACK5
        SUB LOC,02
        LOOP BACK
        CALL EXIT_PROGRAM
        VIDEO_MODE PROC
        MOV AH,00H
        MOV AL,00H
        INT 10H;set video mode
        RET
        VIDEO_MODE ENDP
        CLEAR_SCREEN PROC
        MOV AH,06H
        MOV AL,00H
        MOV BH,07H
        MOV CX,0000H
        MOV DX,1827H
        INT 10H;clear screen
        RET
        CLEAR_SCREEN ENDP
        SET_CURSOR PROC
        MOV AH,02H
        MOV BH,00
        INT 10H;set cursor
        RET
        SET_CURSOR ENDP
        DISPLAY PROC
        MOV AH,09H
        LEA DX,MSG
        INT 21H;display string
        RET
        DISPLAY ENDP
        EXIT_PROGRAM PROC
        MOV AH,4CH
        INT 21H
        RET
        EXIT_PROGRAM ENDP
        END
```

17. Write an assembly language program to generate a multiplication table of any number entered by the user. Display the table in the screen.

Sol^n:

```
.MODEL SMALL
.STACK 64
.DATA
        NUM DB 2 DUP(?)
        TEMP DW ?
        TEMP2 DW ?
        TEMP3 DW ?
        TEMP4 DW ?
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV CX,0002
        LEA DI,NUM
   KJ:MOV AH,01H;input two digit
        INT 21H
        MOV [DI],AL
        INC DI
        LOOP KJ
        MOV AH,02H
        MOV DL,20H
        INT 21H
        MOV SI,OFFSET NUM
        MOV DH,[SI]
        INC SI
        MOV DL,[SI]
        SUB DH,30H;converting to eqvt HEX
        SUB DL,30H
        MOV CL,04H
        ROL DH,CL
        OR DH,DL
        MOV CL,DH
        MOV CH,00H
  LABELS:CMP CL,10H
        JB NEXT
        INC CH
        SUB CL,10H
        JMP LABELS
NEXT:ADD CL,0AH
        DEC CH
        JNZ NEXT
        MOV CH,00
        MOV TEMP,CX;HEX eqvt to temp
        CALL VIDEO_MODE
        CALL CLEAR_SCREEN
        MOV CX,000AH
        MOV BL,01H
        MOV DH,00H
        MOV DL,00H;set cursor at top at first
LABEL1:CALL SET_CURSOR
        MOV AX,TEMP
        MUL BL;multiplied at ax
        CALL DECIMAL
        INC BL
        LOOP LABEL1
        MOV AH,4CH
        INT 21H
        VIDEO_MODE PROC
        MOV AH,00H
        MOV AL,00H
        INT 10H;set video mode
        RET
        VIDEO_MODE ENDP
        CLEAR_SCREEN PROC
        MOV AH,06H
        MOV AL,00H
        MOV BH,07H
        MOV CX,0000H
        MOV DX,1827H
        INT 10H;clear whole screen
        RET
        CLEAR_SCREEN ENDP
        SET_CURSOR PROC
        MOV AH,02H
        MOV BH,00
        INC DH
        MOV DL,00
        INT 10H;set cursor
        RET
        SET_CURSOR ENDP
        DECIMAL PROC;display eqvt BCD
        MOV TEMP4,BX
        MOV BX,0000H
        MOV TEMP3,CX
        MOV CX,000AH
        MOV TEMP2,DX
  DCE: MOV DX,0000H
        DIV CX
        ADD DX,30H
        PUSH DX
        INC BX
        CMP AX,0000
        JA DCE
        MOV CX,BX
        MOV AH,02H
 SHOW: POP DX
        INT 21H
        LOOP SHOW
        MOV DX,TEMP2
        MOV CX,TEMP3
        MOV BX,TEMP4
        RET
        DECIMAL ENDP
END
```

18. Write a program to find the HCF of two unsigned 16-bit numbers.

Sol^{n}:

```
.MODEL SMALL
.STACK 64
.DATA
        NUM1 DW 0005
        NUM2 DW 0015
        TEMP1 DW ?
        TEMP2 DW ?
        SUM DW ?
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV AX,NUM1
        MOV BX,NUM2
        CMP AX,BX
        JA NEXT
        XCHG AX,BX;find greatest
        MOV SUM,BX
 NEXT:MOV DX,0000H;finding HCF
        MOV TEMP1,AX
        MOV TEMP2,BX
        DIV BX
        CMP DX,0000H
        JE LABEL1
        DEC TEMP2
        MOV AX,TEMP1
```

```
          MOV BX,TEMP2
          JMP NEXT
 LABEL1:MOV AX,SUM
          MOV BX,TEMP2
          MOV DX,0000H
          DIV BX
          CMP DX,0000H
          JE LABEL2
          DEC TEMP2
          MOV AX,TEMP1
          MOV BX,TEMP2
          JMP NEXT
 LABEL2:MOV AX,TEMP2;HCF
          ;BCD display
          MOV BX,0000
          MOV CX,0AH
   DCE:MOV DX,0000H
          DIV CX
          ADD DX,30H
          PUSH DX
          INC BX
          CMP AX,0000
          JA DCE
          MOV CX,BX
          MOV AH,02
  SHOW:POP DX
          INT 21H
          LOOP SHOW
          MOV AH,4CH
          INT 21H
END
```

19. Write a program to find the LCM of two unsigned 16-bit numbers.
Sol^{n}:

```
.MODEL SMALL
.STACK 64
.DATA
          NUM1 DW 0005
          NUM2 DW 0003
          TEMP1 DW ?
          TEMP2 DW ?
          SUM DW ?
.CODE
          MOV AX,@DATA
          MOV DS,AX
          MOV AX,NUM1
          MOV BX,NUM2
          CMP AX,BX
          JA NEXT
          XCHG AX,BX;find greatest
          MOV SUM,BX
 NEXT:MOV DX,0000H
          MOV TEMP1,AX
          MOV TEMP2,BX
          DIV BX
          CMP DX,0000H
          JE LABEL1
          DEC TEMP2
          MOV AX,TEMP1
          MOV BX,TEMP2
          JMP NEXT
 LABEL1:MOV AX,SUM
          MOV BX,TEMP2
          MOV DX,0000H
          DIV BX
          CMP DX,0000H
          JE LABEL2
          DEC TEMP2
          MOV AX,TEMP1
          MOV BX,TEMP2;HCF
          JMP NEXT
  LABEL2:MOV AX,NUM1
          MUL NUM2
          MOV DX,0000H
          DIV TEMP2;LCM
          BCD display
          MOV BX,0000
          MOV CX,0AH
   DCE:MOV DX,0000H
          DIV CX
          ADD DX,30H
          PUSH DX
          INC BX
          CMP AX,0000
          JA DCE
          MOV CX,BX
          MOV AH,02
  SHOW:POP DX
          INT 21H
          LOOP SHOW
          MOV AH,4CH
          INT 21H
END
```

20. Write a program that takes a string from a user and displays each word in a new line diagonally from upper left towards bottom right in a clear screen. If the string is “Programming in Assembly Language is Fun”, it should be displayed as follows:

```
Programming
                  in
                     Assembly
                                   Language
                                              is
                                                   Fun
```

Sol^n:

```
.MODEL SMALL
.STACK 100
.DATA
        MSG DB 42 DUP(?)
        TEMP DB ?
.CODE
        MOV AX,@DATA
        MOV DS,AX
        LEA SI,MSG
        MOV [SI],41
        MOV AH,0AH
        MOV CX,0040
        LEA DX,MSG
        INT 21H;input string
        MOV SI,OFFSET MSG
        ADD SI,02
        MOV [SI+0041],'$'
        CALL VIDEO_MODE
        CALL CLEAR_SCREEN
        MOV DX,0205H
        MOV BL,00H
 LABEL1:CALL SET_CURSOR;set cursor at every
space
        MOV AH,02H
        MOV TEMP,DL
 BACK:MOV DL,[SI]
        CMP DL,20H;check space
        JNE NEXT
        INC DH
        MOV DL,TEMP
        INC DL
        INC SI
        JMP LABEL1
  NEXT:CMP DL,'$';check end point
        JE GO1
        INC SI
        INC TEMP
        INT 21H
        JMP BACK
   GO1:CALL EXIT_PROGRAM
        VIDEO_MODE PROC
        MOV AH,00H
        MOV AL,02H
        INT 10H;set video mode
        RET
        VIDEO_MODE ENDP
        CLEAR_SCREEN PROC
        MOV AH,06H
        MOV AL,00H
        MOV BH,07H
        MOV CX,0000H
        MOV DX,1827H
        INT 10H;clear screen
        RET
        CLEAR_SCREEN ENDP
        SET_CURSOR PROC
        MOV AH,02H
        MOV BH,00
        INT 10H;set cursor as co ordinate
        RET
        SET_CURSOR ENDP
        EXIT_PROGRAM PROC
        MOV AH,4CH
        INT 21H
        RET
        EXIT_PROGRAM ENDP
        END
```

21. Write an assembly language program that calculates the sum of the elements of a 3 by 3 matrix. The 3 by 3 matrix is entered by the user and the sum should be displayed on the PC screen. The program should be able to handle unsigned and signed numbers.

Sol^n:

```
.MODEL SMALL
.STACK 64
.DATA
        NUM1 DB 1,2,3,4,5,6,7,8,9
        NUM2 DB 1,2,3,4,5,6,7,8,9
        NUM3 DB 9 DUP(?)
        TEMP1 DB 3
        TEMP2 DB 3
.CODE
        MOV AX,@DATA
        MOV DS,AX
```

```
          LEA SI,NUM1
          LEA BX,NUM2
          LEA DI,NUM3
          MOV CX,09
 BACK:MOV AL,[SI]
          ADD AL,[BX]
          MOV [DI],AL
          INC SI
          INC DI
          INC BX
          LOOP BACK
          LEA SI,NUM3
 BACK2:MOV BX,0000
          MOV CX,000AH
          MOV AL,[SI]
          MOV AH,00
  DDD: MOV DX,0000
          DIV CX
          ADD DX,30H
          PUSH DX
          INC BX
          CMP AX,0000
          JA DDD
          MOV CX,BX
          MOV AH,02
 SHOW:POP DX
          INT 21H
          LOOP SHOW
          INC SI
          MOV AH,02
          MOV DL,20H
          INT 21H
          DEC TEMP1
          JNZ BACK2
          MOV TEMP1,03
          MOV AH,02
          MOV DL,0AH
          INT 21H
          DEC TEMP2
          JNZ BACK2
          MOV AH,4CH
          INT 21H
END
```

22. Write an assembly language program to find the sum of numbers from (1) to (n). Read (n) from the user and display the sum in decimal format (also try to display the sum in Hexadecimal format)
Sol^n:

```
.MODEL SMALL
.STACK 64
.DATA
          SUM DW ?
          NUM DB 2 DUP(?)
.CODE
          MOV AX,@DATA
          MOV DS,AX
          MOV CX,0002
          LEA DI,NUM
   KJ:MOV AH,01H;input two digit no
          INT 21H
          MOV [DI],AL
          INC DI
          LOOP KJ
          MOV AH,02H
          MOV DL,20H
          INT 21H;print space
          MOV SI,OFFSET NUM
          MOV DH,[SI]
          INC SI
          MOV DL,[SI]
          SUB DH,30H
          SUB DL,30H
          MOV CL,04H
          ROL DH,CL
          OR DH,DL
          MOV AX,0000H
          MOV CL,DH
          MOV CH,00H
  LABELS:CMP CL,10H
          JB NEXT
          INC CH
          SUB CL,10H
          JMP LABELS
  NEXT:ADD CL,0AH;converted to eqvt HEX
          DEC CH
          JNZ NEXT
   BACK:ADD AX,CX
          LOOP BACK
          MOV SUM,AX;adding
          MOV AX,SUM;BCD display
          MOV BX,0000
          MOV CX,0AH
   DCE:MOV DX,0000H
          DIV CX
          ADD DX,30H
          PUSH DX
```

```
        INC BX
        CMP AX,0000
        JA DCE
        MOV CX,BX
        MOV AH,02
  SHOW:POP DX
        INT 21H
        LOOP SHOW
        MOV AH,02
        MOV DL,20H
        INT 21H;print space
        MOV AX,SUM;HEX display
        MOV BX,0000
        MOV CX,10H
  DCE1:MOV DX,0000H
        DIV CX
        CMP DX,000AH
        JA L3
        ADD DX,30H
        JMP L2
  L3:ADD DX,37H
  L2:PUSH DX
        INC BX
        CMP AX,0000
        JA DCE1
        MOV CX,BX
        MOV AH,02
  SHOW1:POP DX
        INT 21H
        LOOP SHOW1
        MOV AH,4C
        INT 21H
END
```

23. Write a program to find the sum of the following series up to the terms specified by the user and display the result in decimal format. (also try to display the sum in HEX format) (2*4) + (3*6) + (4*8) + … to (n) terms

Sol^{n}:

```
.MODEL SMALL
.STACK 64
.DATA
        SUM DW ?
        NUM DB 2 DUP(?)
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV CX,0002
        LEA DI,NUM
  KJ:MOV AH,01H;input 2 digit no
        INT 21H
        MOV [DI],AL
        INC DI
        LOOP KJ
        MOV AH,02H
        MOV DL,20H
        INT 21H
        MOV SI,OFFSET NUM
        MOV DH,[SI]
        INC SI
        MOV DL,[SI]
        SUB DH,30H
        SUB DL,30H
        MOV CL,04H
        ROL DH,CL
        OR DH,DL
        MOV AX,0000H
        MOV CL,DH
        MOV CH,00H
LABELS:CMP CL,10H
        JB NEXT
        INC CH
        SUB CL,10H
        JMP LABELS
NEXT:CMP CH,00H
        JA KJJ
        JMP JK
 KJJ:ADD CL,0AH;converted to eqvt HEX
        DEC CH
        JNZ KJJ;INPUT
  JK:MOV BX,0000H
  BACK:MOV AX,CX
        INC AX
        MOV DL,AL
        MUL DL
        ADD BX,AX;addition of series
        LOOP BACK
        MOV DX,0000H
        MOV AX,BX
        MOV CX,0002
        MUL CX
        MOV SUM,AX
        MOV CH,00H
        MOV AX,SUM;decimal;display;of;sum
        MOV BX,0000
        MOV CX,0AH
 DCE:MOV DX,0000H
         DIV CX
         ADD DX,30H
         PUSH DX
         INC BX
         CMP AX,0000
         JA DCE
         MOV CX,BX
         MOV AH,02
 SHOW:POP DX
         INT 21H
         LOOP SHOW
         MOV AH,02
         MOV DL,20H
         INT 21H
         MOV AX,SUM;HEX;display
         MOV BX,0000
         MOV CX,10H
 DCE1:MOV DX,0000H
         DIV CX
         CMP DX,000AH
         JA L3
         ADD DX,30H
         JMP L2
 L3:ADD DX,37H
 L2:PUSH DX
         INC BX
         CMP AX,0000
         JA DCE1
         MOV CX,BX
         MOV AH,02
 SHOW1:POP DX
         INT 21H
         LOOP SHOW1
         MOV AH,4C
         INT 21H
END
```

24. Writ a program to find out if a number entered by the user is prime or not. If the number is prime, the output on the screen should say "The number is a prime number", else if the number is not prime, the output on the screen should say "The number is not a prime number".

Sol^n:

```
.MODEL SMALL
.STACK 100
.DATA
         TRUE DB 'THE NUMBER IS PRIME$'
         FALSE DB 'THE NUMBER IS NOT PRIME$'
         SUM DW ?
         NUM DB 2 DUP(?)
         TEMP DW ?
.CODE
         MOV AX,@DATA
         MOV DS,AX
         MOV CX,0002
         LEA DI,NUM
 KJ:MOV AH,01H
         INT 21H
         MOV [DI],AL
         INC DI
         LOOP KJ
         MOV AH,02H
         MOV DL,20H
         INT 21H
         MOV SI,OFFSET NUM
         MOV DH,[SI]
         INC SI
         MOV DL,[SI]
         SUB DH,30H
         SUB DL,30H
         MOV CL,04H
         ROL DH,CL
         OR DH,DL
         MOV AX,0000H
         MOV CL,DH
         MOV CH,00H
LABELS:CMP CL,10H
         JB NEXT
         INC CH
         SUB CL,10H
         JMP LABELS
NEXT:CMP CH,00H
         JA KJJ
         JMP JK
 KJJ:ADD CL,0AH
         DEC CH
         JNZ KJJ;INPUT
 JK:MOV AX,CX
         MOV TEMP,AX
         DEC CX
         DEC CX
BACKA:MOV DX,CX
         INC DX
         DIV DL
         CMP AH,00H
         JE YES
```

```
            MOV AX,TEMP
            LOOP BACKA
            JMP NO
   YES:LEA DX,FALSE
            MOV AH,09H
            INT 21H
            JMP EN
            NO:LEA DX,TRUE
            MOV AH,09H
            INT 21H
     EN:MOV AH,4CH
            INT 21H
END
```

25. Write a program that retrieves the system date and time and displays the information on the PC screen.

Sol^n:

```
.MODEL SMALL
.STACK 100H
.DATA
MSG DB "TODAY'S DATE IS ",'$'
MSG1 DB "TODAY'S TIME IS ",'$'
.CODE
MOV AX,@DATA
MOV DS,AX
MOV DX,OFFSET MSG
MOV AH,09H
INT 21H
MOV AH,2AH
INT 21H
PUSH CX
MOV CX,0
MOV CL,DL
PUSH CX
MOV CL,DH
PUSH CX
MOV DH,0
MOV DX,0
POP AX
MOV CX,0
MOV BX,10
DIVIDE1: DIV BX
PUSH DX
ADD CX,1
MOV DX,0
CMP AX,0
JNE DIVIDE1
DIVIDE2: POP DX
ADD DL,30H
MOV AH,02H
INT 21H
LOOP DIVIDE2
MOV DL,'/'
MOV AH,02H
INT 21H
MOV DX,0
POP AX
MOV CX,0
MOV BX,10
DIVIDE3: DIV BX
PUSH DX
ADD CX,1
MOV DX,0
CMP AX,0
JNE DIVIDE3
DIVIDE4: POP DX
ADD DL,30H
MOV AH,02H
INT 21H
LOOP DIVIDE4
MOV DL,'/'
MOV AH,02H
INT 21H
MOV DX,0
POP AX
MOV CX,0
MOV BX,10
DIVIDE5: DIV BX
PUSH DX
ADD CX,1
MOV DX,0
CMP AX,0
JNE DIVIDE5
DIVIDE6: POP DX
ADD DL,30H
MOV AH,02H
INT 21H
LOOP DIVIDE6
MOV DL,0AH
MOV AH,02H
INT 21H
MOV DX,OFFSET MSG1
MOV AH,09H
INT 21H
MOV AH,2CH
```

```
INT 21H
MOV DL,DH
MOV DH,0
PUSH DX
MOV DL,CL
PUSH DX
MOV DL,CH
PUSH DX
MOV DH,0
MOV DX,0
POP AX
MOV CX,0
MOV BX,10
TDIVIDE1: DIV BX
PUSH DX
ADD CX,1
MOV DX,0
CMP AX,0
JNE TDIVIDE1
TDIVIDE2: POP DX
ADD DL,30H
MOV AH,02H
INT 21H
LOOP TDIVIDE2
MOV DL,'-'
MOV AH,02H
INT 21H
MOV DX,0
POP AX
MOV CX,0
MOV BX,10
TDIVIDE3: DIV BX
PUSH DX
ADD CX,1
MOV DX,0
CMP AX,0
JNE TDIVIDE3
TDIVIDE4: POP DX
ADD DL,30H
MOV AH,02H
INT 21H
LOOP TDIVIDE4
MOV DL,'-'
MOV AH,02H
INT 21H
MOV DX,0
POP AX
MOV CX,0
MOV BX,10
TDIVIDE5: DIV BX
PUSH DX
ADD CX,1
MOV DX,0
CMP AX,0
JNE TDIVIDE5
TDIVIDE6: POP DX
ADD DL,30H
MOV AH,02H
INT 21H
LOOP TDIVIDE6
MOV AH,4CH
INT 21H
END
```

EXAM SOLUTION 8086

1. Write an assembly program to read a string from the user and display vowels and consonant separately.
   Sol^{n}:

```
.MODEL SMALL
.STACK 64
.DATA
        VOWEL DB ' VOWELS ARE=$'
        CONST DB '    CONSONENT ARE=$'
        STRING DB 10 DUP(?)
        VO DB 10 DUP(?),'$'
        CO DB 10 DUP(?),'$'
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV AH,0AH
        MOV CX,0AH
        LEA SI,STRING
        MOV [SI],10
        LEA DX,STRING
        INT 21H
        LEA DI,VO
        INC DI
        LEA BX,CO
        INC BX
        LEA SI,STRING
        ADD SI,02
        MOV CX,08H
  BACK:MOV AH,[SI]
        CMP AH,'a'
        JE NEXT
        CMP AH,'e'
        JE NEXT
        CMP AH,'i'
        JE NEXT
        CMP AH,'o'
        JE NEXT
        CMP AH,'u'
        JE NEXT
        MOV AL,[SI]
        MOV [BX],AL
        INC BX
        JMP GO
  NEXT:MOV AL,[SI]
        MOV [DI],AL
        INC DI
    GO:INC SI
        LOOP BACK
        MOV AH,02
        MOV DL,0AH
        MOV AH,09
        LEA DX,VOWEL
        INT 21H
        MOV DL,0AH
        INT 21H
        LEA DX,VO
        MOV AH,09H
        INT 21H
        MOV AH,02
        MOV DL,0AH
        INT 21H
        LEA DX,CONST
        MOV AH,09H
        INT 21H
        LEA DX,CO
        MOV AH,09H
        INT 21H

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

2. Write a program in 8086 to read a string and display each word in separate line in centre of screen.
   Sol^n:

```
.MODEL SMALL
.STACK 64
.DATA
        STRING DB 41,41 DUP(?),'$'
.CODE
        MOV AX,@DATA
        MOV DS,AX
        LEA DX,STRING
        MOV AH,0AH
        MOV CX,39
        INT 21H
        LEA SI,STRING
        ADD SI,02
        CALL VIDEO_MODE
        CALL CLEAR_SCREEN
        CALL SET_CURSOR
   BACK:MOV AL,[SI]
        CMP AL,'$'
        JE NEXT
        CMP AL,' '
        JE L1
        MOV AH,02H
        MOV DL,AL
        INT 21H
        INC SI
        JMP BACK
     L1:MOV AH,03H
        INT 10H
        INC DH
        MOV DL,20
        MOV AH,02H
        MOV BH,00
        INT 10H
        INC SI
        JMP BACK

   NEXT:MOV AH,4CH
        INT 21H

        VIDEO_MODE PROC
        MOV AX,0000H
        INT 10H
        RET
        VIDEO_MODE ENDP

        CLEAR_SCREEN PROC
        MOV AH,06H
        MOV AL,00H
        MOV BH,07H
        MOV CX,0000
        MOV DX,1827H
        INT 10H
        RET
        CLEAR_SCREEN ENDP

        SET_CURSOR PROC
        MOV AH,02H
        MOV DX,0C14H
        MOV BH,00
        INT 10H
        RET
        SET_CURSOR ENDP

END
```

3. Write an assembly program to read a text from keyword, convert the text into uppercase and display on the clear screen.
   Sol^n:

```
.MODEL SMALL
.STACK 64
.DATA
        STRING DB 15,15 DUP(?),'$'
.CODE
        MOV AX,@DATA
        MOV DS,AX
        LEA DX,STRING
        MOV CX,14
        MOV AH,0AH
        INT 21H
        CALL VIDEO_MODE
        CALL CLEAR_SCREEN
        CALL SET_CURSOR
        CALL UPPERCASE
        CALL DISPLAY
        MOV AH,4CH
        INT 21H

        VIDEO_MODE PROC
        MOV AX,0000
        INT 10H
        RET
        VIDEO_MODE ENDP

        CLEAR_SCREEN PROC
        MOV AH,06H
        MOV AL,00H
        MOV CX,0000
        MOV DX,1827H
        MOV BH,07H
        INT 10H
        RET
        CLEAR_SCREEN ENDP

        SET_CURSOR PROC
        MOV AH,02H
        MOV DH,12
        MOV DL,15
        MOV BH,00
        INT 10H
        RET
        SET_CURSOR ENDP

        UPPERCASE PROC
        LEA SI,STRING
        ADD SI,02
        MOV CX,0012
  BACK:MOV AL,[SI]
        CMP AL,61H
        JB NEXT
        CMP AL,7AH
        JA NEXT
        AND AL,11011111B
        MOV [SI],AL
  NEXT:INC SI
        LOOP BACK
        RET
        UPPERCASE ENDP

        DISPLAY PROC
        LEA DX,STRING
        ADD DX,02
        MOV AH,09H
        INT 21H
        RET
        DISPLAY ENDP
END
```

4. Write a program to read string and display only the alphabetic characters from the string in clear screen.
   Sol^n:

```
.MODEL SMALL
.STACK 64
.DATA
        STRING DB 15,15 DUP(?),'$'
        ALPHA DB 15 DUP(?),'$'
.CODE
        MOV AX,@DATA
        MOV DS,AX
        LEA DX,STRING
        MOV CX,14
        MOV AH,0AH
        INT 21H
        CALL VIDEO_MODE
        CALL CLEAR_SCREEN
        CALL SET_CURSOR
        CALL ALPHAS
        CALL DISPLAY
        MOV AH,4CH
        INT 21H

        VIDEO_MODE PROC
        MOV AX,0000
        INT 10H
        RET
        VIDEO_MODE ENDP

        CLEAR_SCREEN PROC
        MOV AH,06H
        MOV AL,00H
        MOV CX,0000
        MOV DX,1827H
        MOV BH,07H
        INT 10H
        RET
        CLEAR_SCREEN ENDP

        SET_CURSOR PROC
        MOV AH,02H
        MOV DH,12
        MOV DL,15
        MOV BH,00
        INT 10H
        RET
        SET_CURSOR ENDP

        ALPHAS PROC
        LEA SI,STRING
        ADD SI,02
        LEA DI,ALPHA
        ADD DI,02
        MOV CX,0012
BACK:MOV AL,[SI]
        CMP AL,61H
        JB NEXT
        CMP AL,7AH
        JA NEXT
        MOV [DI],AL
        INC DI
NEXT:INC SI
        LOOP BACK
        RET
        ALPHAS ENDP

        DISPLAY PROC
        LEA DX,ALPHA
        ADD DX,02
        MOV AH,09H
        INT 21H
        RET
        DISPLAY ENDP
END
```

5. Write a program in 8086 to read string . Display each word in separate lines in cleared lines in a cleared screen, count how many words are there and display the count.
   Sol^{n}:

```
.MODEL SMALL
.STACK 64
.DATA
        STRING DB 41,41 DUP(?),'$'
        TEMP DW 0000
.CODE
        MOV AX,@DATA
        MOV DS,AX
        LEA DX,STRING
        MOV AH,0AH
        MOV CX,39
        INT 21H
        LEA SI,STRING
        ADD SI,02
        CALL VIDEO_MODE
        CALL CLEAR_SCREEN
        CALL SET_CURSOR
   BACK:MOV AL,[SI]
        CMP AL,'$'
        JE NEXT
        CMP AL,' '
        JE L1
        MOV AH,02H
        MOV DL,AL
        INT 21H
        INC SI
        JMP BACK
     L1:INC TEMP
        MOV AH,03H
        INT 10H
        INC DH
        MOV DL,20
        MOV AH,02H
        MOV BH,00
        INT 10H
        INC SI
        JMP BACK
```

```
  NEXT:INC TEMP
        MOV AX,TEMP
        CALL VALUES1
        MOV AH,4CH
        INT 21H

        VIDEO_MODE PROC
        MOV AX,0000H
        INT 10H
        RET
        VIDEO_MODE ENDP

        CLEAR_SCREEN PROC
        MOV AH,06H
        MOV AL,00H
        MOV BH,07H
        MOV CX,0000
        MOV DX,1827H
        INT 10H
        RET
        CLEAR_SCREEN ENDP

        SET_CURSOR PROC
        MOV AH,02H
        MOV DX,0C14H
        MOV BH,00
        INT 10H
        RET
        SET_CURSOR ENDP

        VALUES1 PROC
        MOV BX,0000
        MOV CX,0AH
   DCE:MOV DX,0000H
        DIV CX
        ADD DX,30H
        PUSH DX
        INC BX
        CMP AX,0000
        JA DCE
        MOV CX,BX
        MOV AH,02
  SHOW:POP DX
        INT 21H
        LOOP SHOW
        RET
        VALUES1 ENDP
END
```

6. WAP in 8086 to find largest and smallest and display them.
   Sol^{n}:

```
.MODEL SMALL
.STACK 64
.DATA
        DATA1 DW
2214,5231,65535,4532,3219,55555,773
1,8399,9911,1111
        LARGEST DW ?
        SMALLEST DW ?
        LRG DB 'LARGEST IS$'
        SML DB 'SMALLEST IS$'
.CODE
        MOV AX,@DATA
        MOV DS,AX
        LEA SI,DATA1
        MOV CX,000AH
        MOV AX,[SI]
        ADD SI,02
  BACK:MOV BX,[SI]
        CMP AX,BX
        JA NEXT
        MOV AX,BX
  NEXT:ADD SI,02
        LOOP BACK
        MOV LARGEST,AX
        LEA SI,DATA1
        MOV CX,000AH
        MOV AX,[SI]
        ADD SI,02
  BACK1:MOV BX,[SI]
        CMP AX,BX
        JB NEXT1
        MOV AX,BX
  NEXT1:ADD SI,02
        LOOP BACK1
        MOV SMALLEST,AX
        CALL JOKE1
        CALL NEW_LINE
        MOV AX,LARGEST
        CALL DISPLAY
        CALL NEW_LINE
        CALL JOKE2
```

```
          CALL NEW_LINE
          MOV AX,SMALLEST
          CALL DISPLAY

          MOV AH,4CH
          INT 21H

          JOKE1 PROC
          LEA DX,LRG
          MOV AH,09H
          INT 21H
          RET
          JOKE1 ENDP

          JOKE2 PROC
          LEA DX,SML
          MOV AH,09H
          INT 21H
          RET
          JOKE2 ENDP

          NEW_LINE PROC
          MOV AH,03H
          INT 10H
          INC DH
          MOV DL,00
          MOV AH,02H
          INT 10H
          RET
          NEW_LINE ENDP

          DISPLAY PROC
          MOV BX,0000H
          MOV CX,000AH
     DCE: MOV DX,0000H
          DIV CX
          ADD DX,30H
          PUSH DX
          INC BX
          CMP AX,0000
          JA DCE
          MOV CX,BX
          MOV AH,02H
    SHOW: POP DX
          INT 21H
          LOOP SHOW
          RET
          DISPLAY ENDP
END
```

7. WAP in 8086 to convert vowels to uppercase from a string entered by the user and display the converted string in a new line. Also count no of uppercase in converted string and display count.
Sol^{n}:

```
.MODEL SMALL
.STACK 64
.DATA
          STRING DB 17,17 DUP(?),'$'
          COUNT1 DW 0000
.CODE
          MOV AX,@DATA
          MOV DS,AX
          MOV AH,0AH
          LEA DX,STRING
          MOV CX,0015
          INT 21H
          CALL NEW_LINE
          CALL UPPERCASE
          CALL DISPLAY
          CALL COUNTS
          CALL NEW_LINE
          CALL DECIMAL
          MOV AH,4CH
          INT 21H

          NEW_LINE PROC
          MOV AH,03H
          INT 10H
          INC DH
          MOV DL,00
          MOV AH,02H
          INT 10H
          RET
          NEW_LINE ENDP

          UPPERCASE PROC
          LEA SI,STRING
          ADD SI,02
          MOV CX,15
BACK: MOV AL,[SI]
          CMP AL,'a'
          JE OVER
          CMP AL,'e'
          JE OVER
          CMP AL,'i'
          JE OVER
          CMP AL,'o'
          JE OVER
          CMP AL,'u'
          JE OVER
          JMP GO
OVER: AND AL,11011111B
  GO: MOV [SI],AL
          INC SI
          LOOP BACK
          RET
          UPPERCASE ENDP

          DISPLAY PROC
          LEA DX,STRING
          ADD DX,01H
          MOV AH,09H
          INT 21H
          RET
          DISPLAY ENDP

          COUNTS PROC
          LEA SI,STRING
          ADD SI,02
          MOV CX,15
 LA1: MOV AL,[SI]
          CMP AL,41H
          JB NEXT
          CMP AL,5AH
          JA NEXT
          INC COUNT1
NEXT: INC SI
          LOOP LA1
          RET
          COUNTS ENDP

          DECIMAL PROC
          MOV AX,COUNT1
          MOV BX,0000H
          MOV CX,000AH
   DCE: MOV DX,0000H
          DIV CX
          ADD DX,30H
          PUSH DX
          INC BX
          CMP AX,0000
          JA DCE
          MOV CX,BX
          MOV AH,02H
  SHOW: POP DX
          INT 21H
          LOOP SHOW
          RET
          DECIMAL ENDP
END
```

8. WAP in 8086 to read string and count no of vowels and display string & count in clear screen.
   Sol^{n}:

```
.MODEL SMALL
.STACK 64
.DATA
          MSG DB 10,10 DUP(?),'$'
          COUNT1 DW 0000
.CODE
          MOV AX,@DATA
          MOV DS,AX
          MOV AH,0AH
          MOV CX,09H
          LEA DX,MSG
          INT 21H

          MOV AH,03H
          INT 10H
          INC DH
          MOV AH,02H
          MOV DL,00
          INT 10H

          MOV SI,OFFSET MSG
          MOV CX,0006H
          MOV BX,0000H
```

```
        ADD SI,02H
BACK:MOV AH,[SI]
        CMP AH,61H
        JE NEXT
        CMP AH,65H
        JE NEXT
        CMP AH,69H
        JE NEXT
        CMP AH,6FH
        JE NEXT
        CMP AH,75H
        JE NEXT
        JMP GOTO1
 NEXT:INC BX
GOTO1:INC SI
        LOOP BACK
        MOV COUNT1,BX
        CALL VIDEO_MODE
        CALL CLEAR_SCREEN
        CALL SET_CURSOR

        LEA DX,MSG
        ADD DX,02
        MOV AH,09H
        INT 21H

        MOV AH,03H
        INT 10H
        INC DH
        MOV DL,00
        MOV AH,02H
        INT 10H

        MOV AX,COUNT1
        CALL DECIMAL

        MOV AH,4CH
        INT 21H

        VIDEO_MODE PROC
        MOV AX,0000H
        INT 10H
        RET
        VIDEO_MODE ENDP

        CLEAR_SCREEN PROC
        MOV AH,06H
        MOV AL,00H
        MOV BH,07H
        MOV CX,0000
        MOV DX,1827H
        INT 10H
        RET
        CLEAR_SCREEN ENDP

        SET_CURSOR PROC
        MOV AH,02H
        MOV DX,0C14H
        MOV BH,00
        INT 10H
        RET
        SET_CURSOR ENDP

        DECIMAL PROC
        MOV BX,0000H
        MOV CX,000AH
  DCE: MOV DX,0000H
        DIV CX
        ADD DX,30H
        PUSH DX
        INC BX
        CMP AX,0000
        JA DCE
        MOV CX,BX
        MOV AH,02H
 SHOW: POP DX
        INT 21H
        LOOP SHOW
        RET
        DECIMAL ENDP
END
```

9. WAP in 8086 to find multiplication table of two digit no.
   Sol^n:

```
.MODEL SMALL
.STACK 64
.DATA
        NUM DB 2 DUP(?)
        TEMP DW ?
        TEMP2 DW ?
        TEMP3 DW ?
        TEMP4 DW ?
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV CX,0002
        LEA DI,NUM
   KJ:MOV AH,01H
        INT 21H
        MOV [DI],AL
        INC DI
        LOOP KJ
        MOV AH,02H
        MOV DL,20H
        INT 21H
        MOV SI,OFFSET NUM
        MOV DH,[SI]
        INC SI
        MOV DL,[SI]
        SUB DH,30H
        SUB DL,30H
        MOV CL,04H
        ROL DH,CL
        OR DH,DL
        MOV CL,DH
        MOV CH,00H
  LABELS:CMP CL,10H
        JB NEXT
        INC CH
        SUB CL,10H
        JMP LABELS
  NEXT:ADD CL,0AH
        DEC CH
        JNZ NEXT
        MOV CH,00
        MOV TEMP,CX

        CALL VIDEO_MODE
        CALL CLEAR_SCREEN

        MOV CX,000AH
        MOV BL,01H
        MOV DH,00H
        MOV DL,00H
  LABEL1:CALL SET_CURSOR
        MOV AX,TEMP
        MUL BL
        CALL DECIMAL
        INC BL
        LOOP LABEL1


        MOV AH,4CH
        INT 21H

        VIDEO_MODE PROC
        MOV AH,00H
        MOV AL,00H
        INT 10H
        RET
        VIDEO_MODE ENDP

        CLEAR_SCREEN PROC
        MOV AH,06H
        MOV AL,00H
        MOV BH,07H
        MOV CX,0000H
        MOV DX,1827H
        INT 10H
        RET
        CLEAR_SCREEN ENDP

        SET_CURSOR PROC
        MOV AH,02H
        MOV BH,00
        INC DH
        MOV DL,00
        INT 10H
        RET
        SET_CURSOR ENDP

        DECIMAL PROC
        MOV TEMP4,BX
        MOV BX,0000H
        MOV TEMP3,CX
        MOV CX,000AH
        MOV TEMP2,DX
  DCE: MOV DX,0000H
```

```
            DIV CX
            ADD DX,30H
            PUSH DX
            INC BX
            CMP AX,0000
            JA DCE
            MOV CX,BX
            MOV AH,02H
  SHOW: POP DX
            INT 21H
            LOOP SHOW
            MOV DX,TEMP2
            MOV CX,TEMP3
            MOV BX,TEMP4
            RET
            DECIMAL ENDP
END
```

10. WAP in 8086 to read string and count no of vowels,consonents,numericals and other character and display them.
Sol^{n}:

```
.MODEL SMALL
.STACK 64
.DATA
            STRING DB 15,15 DUP (?)
            VOW DW ?
            CON DW ?
            NUM DW ?
            OTH DW ?
            MSG1 DB 'VOWEL=$'
            MSG2 DB 'CONSONENT=$'
            MSG3 DB 'NUMERICAL=$'
            MSG4 DB 'OTHER=$'
.CODE
            MOV AX,@DATA
            MOV DS,AX
            LEA DX,STRING
            MOV AH,0AH
            MOV CX,14
            INT 21H
            LEA SI,STRING
            ADD SI,02
            MOV CX,14
 BACK:MOV AH,[SI]
            CMP AH,'a'
            JE NEXT
            CMP AH,'e'
            JE NEXT
            CMP AH,'i'
            JE NEXT
            CMP AH,'o'
            JE NEXT
            CMP AH,'u'
            JE NEXT
            CMP AH,30H
            JB L1
            CMP AH,39H
            JA L1
            INC NUM
            JMP GO
  L1:CMP AH,61H
            JB L2
            CMP AH,7AH
            JA L2
            INC CON
            JMP GO
 L2: INC OTH
            JMP GO
NEXT:INC VOW
  GO:INC SI
            LOOP BACK
            MOV AH,02
            MOV DL,0AH
            INT 21H
            MOV AH,09H
            LEA DX,MSG1
            INT 21H
            MOV AX,VOW
            CALL DISPLAY
            MOV AH,02
            MOV DL,0AH
            INT 21H
            MOV AH,09H
            LEA DX,MSG2
            INT 21H
            MOV AX,CON
            CALL DISPLAY
            MOV AH,02
```

```
        MOV DL,0AH
        INT 21H
        MOV AH,09H
        LEA DX,MSG3
        INT 21H
        MOV AX,NUM
        CALL DISPLAY
        MOV AH,02
        MOV DL,0AH
        INT 21H
        MOV AH,09H
        LEA DX,MSG4
        INT 21H
        MOV AX,OTH
        CALL DISPLAY

        MOV AH,4CH
        INT 21H

        DISPLAY PROC
        MOV BX,0000
        MOV CX,0AH
    DCE:MOV DX,0000H
        DIV CX
        ADD DX,30H
        PUSH DX
        INC BX
        CMP AX,0000
        JA DCE
        MOV CX,BX
        MOV AH,02
   SHOW:POP DX
        INT 21H
        LOOP SHOW
        RET
        DISPLAY ENDP
END
```

11. WAP in 8086 of display sum=(1+x)*5+(3+x)*6…upto 10 terms ,where x is no from 0 to 9.
    Sol^{n}:

```
.MODEL SMALL
.STACK 64
.DATA
        SUM DW 0000
        LO DW 01
        HI DW 05
        TEMP DW ?
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV AH,01
        INT 21H
        MOV AH,00
        SUB AL,30H
        MOV TEMP,AX
        MOV CX,000AH
   BACK:MOV DX,0000H
        MOV BX,0000H
        ADD BX,LO
        ADD BX,TEMP
        MOV AX,BX
        MOV BX,HI
        MUL BX
        ADD SUM,AX
        ADD LO,02
        INC HI
        LOOP BACK
        MOV AH,02
        MOV DL,0AH
        INT 21H
        MOV AX,SUM
        MOV BX,0000
        MOV CX,0AH
    DCE:MOV DX,0000H
        DIV CX
        ADD DX,30H
        PUSH DX
        INC BX
        CMP AX,0000
        JA DCE
        MOV CX,BX
        MOV AH,02
   SHOW:POP DX
        INT 21H
        LOOP SHOW
        MOV AH,4CH
        INT 21H
END
```

12. WAP in 8086 to sort no stored in array.

Sol^{n}:

```
.MODEL SMALL
.STACK 64
.DATA
        DATA1 DB 125,235,197,91,48
        COUNT DB 05
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV CX,05
 BACK2:LEA SI,DATA1
        MOV DI,SI
        INC DI
        MOV BX,CX
 BACK1:MOV AL,[SI]
        CMP AL,[DI]
        JA NEXT
        MOV AL,[DI]
        MOV AH,[SI]
        MOV [SI],AL
        MOV [DI],AH
 NEXT: INC SI
        INC DI
        DEC BX
        JNZ BACK1
        LOOP BACK2

        LEA SI,DATA1
  L1: MOV AL,[SI]
        MOV AH,00
        MOV BX,0000
        MOV CX,0AH
   DCE:MOV DX,0000H
        DIV CX
        ADD DX,30H
        PUSH DX
        INC BX
        CMP AX,0000
        JA DCE
        MOV CX,BX
        MOV AH,02
  SHOW:POP DX
        INT 21H
        LOOP SHOW
        MOV AH,02
        MOV DL,20H
        INT 21H
        INC SI
        DEC COUNT
        JNZ L1

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

13. WAP in 8086 of display sum=1^2+2^2+….upto 10 terms and display result.

Sol^{n}:

```
.MODEL SMALL
.STACK 64
.DATA
        SUM DW 0000
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV CX,000AH
        MOV DH,01
  BACK:MOV AH,00
        MOV AL,DH
        MUL DH
        ADD SUM,AX
        INC DH
        LOOP BACK
        MOV AX,SUM

        MOV BX,0000
        MOV CX,0AH
    DCE:MOV DX,0000H
        DIV CX
        ADD DX,30H
        PUSH DX
        INC BX
        CMP AX,0000
        JA DCE
        MOV CX,BX
        MOV AH,02
   SHOW:POP DX
        INT 21H
        LOOP SHOW

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

14. WAP in 8086 to read string and count no of vowels and display no of vowel even or odd on screen.
    Sol^{n}:

```
.MODEL SMALL
.STACK 64
.DATA
        MSG DB 15,15 DUP(?),'$'
        MSG1 DB 'EVEN VOWELS$'
        MSG2 DB 'ODD VOWELS$'
        COUNT1 DW 0000
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV AH,0AH
        MOV CX,12
        LEA DX,MSG
        INT 21H
        MOV AH,02
        MOV DL,0AH
        INT 21H
        MOV SI,OFFSET MSG
        MOV CX,000DH
        MOV BX,0000H
        ADD SI,02H
   BACK:MOV AH,[SI]
        CMP AH,61H
        JE NEXT
        CMP AH,65H
        JE NEXT
        CMP AH,69H
        JE NEXT
        CMP AH,6FH
        JE NEXT
        CMP AH,75H
        JE NEXT
        JMP GOTO1
    NEXT:INC BX
   GOTO1:INC SI
        LOOP BACK
        MOV COUNT1,BX
        MOV AX,BX
        AND AL,00000001
        CMP AX,00H
        JE LL1
        MOV AH,09
        LEA DX,MSG2
        INT 21H
        JMP GO
   LL1:MOV AH,09
        LEA DX,MSG1
        INT 21H
    GO:MOV AH,4CH
        INT 21H
END
```

15. WAP in 8086 of display sum=x+2x+3x+…10 terms ,where x is no from 0 to 99.
    Sol^{n}:

```
.MODEL SMALL
.STACK 64
.DATA
        NUM DB 2 DUP(?)
        SUM DW 0000
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV CX,0002
        LEA DI,NUM
    KJ:MOV AH,01H
        INT 21H
        MOV [DI],AL
        INC DI
        LOOP KJ
        MOV AH,02H
        MOV DL,20H
        INT 21H
        MOV SI,OFFSET NUM
        MOV DH,[SI]
        INC SI
        MOV DL,[SI]
        SUB DH,30H
        SUB DL,30H
        MOV CL,04H
        ROL DH,CL
        OR DH,DL
        MOV CL,DH
```

```
          MOV CH,00H
    LABELS:CMP CL,10H
          JB NEXT
          INC CH
          SUB CL,10H
          JMP LABELS
    NEXT:ADD CL,0AH
          DEC CH
          JNZ NEXT
          MOV CH,00
          MOV BH,CL
          MOV CX,0AH
      B1:MOV AL,BH
          MUL CL
          ADD SUM,AX
          LOOP B1

          MOV AX,SUM
          MOV BX,0000
          MOV CX,0AH
      DCE:MOV DX,0000H
          DIV CX
          ADD DX,30H
          PUSH DX
          INC BX
          CMP AX,0000
          JA DCE
          MOV CX,BX
          MOV AH,02
     SHOW:POP DX
          INT 21H
          LOOP SHOW
          MOV AH,4CH
          INT 21H
  END
```

16. WAP in 8086 to find multiplication table of 5 nos stored in array.
Sol^{n}:

```
.MODEL SMALL
.STACK 64
.DATA
        NUM DB 2,3,4,5,6
        TEMP DW ?
        TEMP2 DW ?
        TEMP3 DW ?
        TEMP4 DW ?
        COUNT DB 05
.CODE
        MOV AX,@DATA
        MOV DS,AX
        CALL VIDEO_MODE
        CALL CLEAR_SCREEN

        LEA SI,NUM
        MOV DH,00H
        MOV DL,00H
   KJ:MOV CX,000AH
        MOV BL,01H
  LABEL1:MOV AL,[SI] ;CALL SET_CURSOR
        MOV AH,00
        MUL BL
        CALL DECIMAL
        INC BL
        LOOP LABEL1
        MOV AH,03H
        INT 10H
        ADD DH,02
        MOV AH,02H
        MOV DL,00
        MOV BH,00
        INT 10H
        INC SI
        DEC COUNT
        JNZ KJ

        MOV AH,4CH
        INT 21H

        VIDEO_MODE PROC
        MOV AH,00H
        MOV AL,00H
        INT 10H
        RET
        VIDEO_MODE ENDP

        CLEAR_SCREEN PROC
        MOV AH,06H
        MOV AL,00H
        MOV BH,07H
        MOV CX,0000H
        MOV DX,1827H
        INT 10H
        RET
        CLEAR_SCREEN ENDP

        ;SET_CURSOR PROC
        ;MOV AH,02H
        ;MOV BH,00
        ;INC DL
        ;MOV DH,00
        ;INT 10H
        ;RET
        ;SET_CURSOR ENDP

        DECIMAL PROC
        MOV TEMP4,BX
        MOV BX,0000H
        MOV TEMP3,CX
        MOV CX,000AH
        MOV TEMP2,DX
  DCE:  MOV DX,0000H
        DIV CX
        ADD DX,30H
        PUSH DX
        INC BX
        CMP AX,0000
        JA DCE
        MOV CX,BX
        MOV AH,02H
  SHOW: POP DX
        INT 21H
        LOOP SHOW
        MOV DL,' '
        INT 21H
        MOV DX,TEMP2
        MOV CX,TEMP3
        MOV BX,TEMP4
        RET
        DECIMAL ENDP
END
```

17. WAP in 8086 to read string and count no of vowels and display string without vowel in clear screen & count in clear screen.
Sol^{n}:

```
.MODEL SMALL
.STACK 64
.DATA
        MSG DB 15,15 DUP(?),'$'
        COUNT1 DW 0000
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV AH,0AH
        MOV CX,12H
        LEA DX,MSG
        INT 21H

        CALL VIDEO_MODE
        CALL CLEAR_SCREEN


        MOV SI,OFFSET MSG
        MOV CX,000DH
        MOV BX,0000H
        ADD SI,02H
   BACK:MOV AL,[SI]
        CMP AL,61H
        JE NEXT
        CMP AL,65H
        JE NEXT
        CMP AL,69H
        JE NEXT
        CMP AL,6FH
        JE NEXT
        CMP AL,75H
        JE NEXT
        MOV AH,02
        MOV DL,AL
        INT 21H
        JMP GOTO1
   NEXT:INC BX
  GOTO1:INC SI
        LOOP BACK
        MOV COUNT1,BX

        MOV AH,02
        MOV DL,0AH
        INT 21H

        MOV AX,COUNT1
        CALL DECIMAL

        MOV AH,4CH
        INT 21H

        VIDEO_MODE PROC
        MOV AX,0000H
        INT 10H
```

```
            RET
            VIDEO_MODE ENDP

            CLEAR_SCREEN PROC
            MOV AH,06H
            MOV AL,00H
            MOV BH,70H
            MOV CX,0000
            MOV DX,1827H
            INT 10H
            RET
            CLEAR_SCREEN ENDP


            DECIMAL PROC
            MOV BX,0000H
            MOV CX,000AH
  DCE: MOV DX,0000H
            DIV CX
            ADD DX,30H
            PUSH DX
            INC BX
            CMP AX,0000
            JA DCE
            MOV CX,BX
            MOV AH,02H
 SHOW: POP DX
            INT 21H
            LOOP SHOW
            RET
            DECIMAL ENDP
END
```