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. Solⁿ:

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)

Solⁿ:

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.

Solⁿ:

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.

Solⁿ:

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.

Solⁿ:

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)

Solⁿ:

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)

Solⁿ:

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.

Solⁿ: 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 BACK: MVI A,00H STAX D STAX D **NEXT: INX B** MOV A.M MVI C,02H INX H INX D LABEL: RLC LDAX B DCR C ADI 00H JNZ LABEL JNZ BACK ANI 80H JNZ JUMP RST 5 MOV A,M 9) Subtract ten 16-bit data stored in one table from **RRC** the other. Store the result in the third table if the ANI 80H result is positive else store 00. JZ NEXT Solⁿ: MOV A.M LXI B,8050H STAX D LXI H,8070H JUMP: NOP LXI D,8090H **NEXT: INX H BACK: LDAX B** INX D SUB M DCR B STAX D JNZ BACK INX B RST 5 INX H 11) Transfer ten data with even parity from location INX D LDAX B 9270H to 9280H, else transfer the data by clearing bit D7 and setting bit D2. SBB M STAX D Solⁿ: **NEXT: INX B** LXI H,9500H MVI B,0AH INX H INX D MVI C.00H LDAX B MVI D,00H ADI 00H MOV A,M JNZ BACK BACK: INX H LXI D,8091H INX H **MVI B.05** ADD M JNC JUMP LABEL: LDAX D ANI 80H INR C JZ JUMP JUMP: DCR B MVI A,00H JNZ BACK STAX D MOV E,A LXI H,9501H DCX D MVI B,0AH MVI A,00H STAX D MOV A.M BACKA: INX H INX D JUMP: INX D INX H INX D ADD M JNC JUMPA DCR B JNZ LABEL INR D RST 5 JUMPA: DCR B JNZ BACKA 10) Transfer ten data, which has bit D5 and D0, 0 ADD C and 1 respectively from A430H to A440H, else store JNC GO 00 instead of transformation. INR D Solⁿ: GO: INX H

> MOV M,E INX H

> MOV M,A

LXI H,0A430H

LXI D,0A440H MVI B,0AH

RST 5

INX H

RST 5

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

Solⁿ:

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.

Solⁿ:

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

Solⁿ:

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

INT 21H LOOP SHOW MOV AH,4CH INT 21H

END

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<sup>n</sup>:
.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

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<sup>n</sup>:
.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	INT 10H;clear whole screen
	CMP AX,0000	RET
	JA DCE	CLEAR_SCREEN ENDP
	MOV CX,BX	SET_CURSOR PROC
	MOV AH,02	MOV AH,02H
SHO	W:POP DX	MOV DH,12
	INT 21H	MOV DL,20
	LOOP SHOW	MOV BH,00
	CALL EXIT	INT 10H;set cursor at centre
	EXIT PROC	RET
	MOV AH,4CH	SET_CURSOR ENDP
	INT 21H	DISPLAY PROC
	RET	LEA DX,MSG
	EXIT ENDP	
END	EXIT ENDP	ADD DX,02H
END		MOV AH,09H
3 Writ	e an assembly language program to get text	INT 21H; dispay string
	nd display it on the center of a clear screen.	RET
Sol ⁿ :	nd display it on the contor of a clear screen.	DISPLAY ENDP
201.		EXIT_PROGRAM PROC
.MODE	LSMALL	MOV AH,4CH
.STACK	100	INT 21H
.DATA		RET
	MSG DB ' ',10 DUP (?)	EXIT_PROGRAM END
.CODE	11.50 55 710 501 (17	END
	MOV AX,@DATA	4. Write an assembly language program to accept
	MOV DS,AX	string input and convert to upper case if it has lower
	MOV AH,0AH	case letters.
	MOV CX,06H	Sol ⁿ :
	LEA DX,MSG	.MODEL SMALL
	INT 21H;string input	STACK 100
	MOV SI,OFFSET MSG	.DATA
	MOV [SI+0008],'\$'	
	CALL VIDEO_MODE	MSG DB ' ',10 DUP (?) .CODE
	CALL CLEAR_SCREEN	
	CALL SET_CURSOR	MOV AX,@DATA
	CALL DISPLAY	MOV DS,AX
	CALL EXIT PROGRAM	MOV AH,0AH
	-	MOV CX,06H
	VIDEO_MODE PROC	LEA DX,MSG
	MOV AH,00H	INT 21H;string input
	MOV AL,00H	MOV SI,OFFSET MSG
	INT 10H;set video mode	MOV [SI+0008],'\$'
	RET	MOV CX,0010
	VIDEO_MODE ENDP	BACK:MOV AL,[SI]
	CLEAR_SCREEN PROC	CMP AL,61H
	MOV AH,06H	JB OVER
	MOV AL,00H	CMP AL,7AH
	MOV BH,07H	JA OVER
	MOV CX,0000H	AND AL,11011111B;convert to uppercase
	MOV DX,1827H	•

OVER:MOV [SI],AL	
INC SI	.MODEL SMALL
LOOP BACK	.STACK 100
CALL VIDEO_MODE	.DATA
CALL CLEAR_SCREEN	MSG DB ' ',10 DUP (?)
CALL SET_CURSOR	.CODE
CALL DISPLAY	MOV AX,@DATA
CALL EXIT_PROGRAM	MOV DS,AX
VIDEO_MODE PROC	MOV AH,0AH
MOV AH,00H	MOV CX,06H
MOV AL,00H	LEA DX,MSG
INT 10H;set video mode	INT 21H;input string
RET	MOV SI, OFFSET MSG
VIDEO_MODE ENDP	MOV [SI+0008],'\$'
_	CALL VIDEO_MODE
CLEAR_SCREEN PROC	CALL CLEAR_SCREEN
MOV AH,06H	CALL SET_CURSOR
MOV AL,00H	CALL DISPLAY
MOV BH,07H	CALL EXIT_PROGRAM
MOV CX,0000H	VIDEO_MODE PROC
MOV DX,1827H	MOV AH,00H
INT 10H;clear whole screen	MOV AL,00H
RET	INT 10H;set video mode
CLEAR_SCREEN ENDP	RET
SET CURSOR PROC	VIDEO_MODE ENDP
MOV AH,02H	CLEAR_SCREEN PROC
MOV DH,12	MOV AH,06H
MOV DL,20	MOV AL,00H
MOV BH,00	MOV BH,07H
INT 10H;set cursor at centre	MOV CX,0000H
RET	MOV DX,1827H
SET_CURSOR ENDP	INT 10H; clear whole screen
DISPLAY PROC	RET
LEA DX,MSG	CLEAR_SCREEN ENDP
ADD DX,02H	SET_CURSOR PROC
MOV AH,09H	MOV AH,02H
INT 21H;display string	MOV DH,20
RET	MOV DL,10
DISPLAY ENDP	MOV BH,00
EXIT_PROGRAM PROC	INT 10H;set cursor at centre
MOV AH,4CH	RET
INT 21H	SET_CURSOR ENDP
RET	DISPLAY PROC
EXIT_PROGRAM ENDP	LEA DX,MSG
END	ADD DX,02
	MOV AH,09H
	INT 21H; dispay string
5. Write an assembly language program to get input	RET
and display on location 10,20 on the screen Sol ⁿ :	DISPLAY ENDP

EXIT PROGRAM PROC RET **CLEAR SCREEN ENDP** MOV AH,4CH INT 21H SET CURSOR PROC MOV AH.02H **RFT EXIT PROGRAM ENDP** MOV DH,12 **END** MOV DL,20 MOV BH.00 INT 10H;set cursor at centre 6. Write an assembly language program to convert the text stored in memory to upper case only if the SET CURSOR ENDP characters are found in lower case. Display the **DISPLAY PROC** converted text in the screen. LEA DX, DATA Solⁿ: MOV AH,09H .MODEL SMALL INT 21H; dispay string .STACK 64 RET **DISPLAY ENDP** .DATA EXIT_PROGRAM PROC DATA DB 'mY NamE is KiShor', '\$' MOV AH,4CH .CODE **INT 21H** MOV AX,@DATA MOV DS,AX RET **MOV SI, OFFSET DATA EXIT PROGRAM ENDP END** MOV CX,0017 BACK:MOV AL,[SI] CMP AL,61H 7. Write an assembly language program to convert JB OVER the text stored in the memory to lower case if the CMP AL,7AH characters are in upper case. Display the result text JA OVER in the screen AND AL,110111111B; convert to upper case Solⁿ: OVER:MOV [SI],AL INC SI .MODEL SMALL LOOP BACK .STACK 64 CALL VIDEO MODE .DATA CALL CLEAR SCREEN DATA DB 'mY NamE is KiShor', '\$' CALL SET CURSOR .CODE **CALL DISPLAY** MOV AX,@DATA CALL EXIT PROGRAM MOV DS,AX VIDEO_MODE PROC **MOV SI, OFFSET DATA** MOV AH,00H MOV CX,0017 MOV AL,00H BACK:MOV AL,[SI] INT 10H;set video mode CMP AL,41H JB OVER RET VIDEO MODE ENDP CMP AL,5AH CLEAR_SCREEN PROC JA OVER MOV AH,06H OR AL,00100000B; convert to lowercase MOV AL,00H OVER:MOV [SI],AL **INC SI** MOV BH,07H MOV CX,0000H LOOP BACK MOV DX,1827H CALL VIDEO MODE CALL CLEAR_SCREEN INT 10H; clear whole screen

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 6X 1011
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

END

11

```
9. Write a program to add the sequence 1+3+4+... up
to the desired steps entered by the user and display
                                                      10. Write an assembly language program to display
the result in decimal format. Assume user enters
                                                      graphical ASCII characters from 32 to 127 on a
numbers from 1 to 9.
                                                      defined window (5, 10 and 20, 70) with white on
Sol<sup>n</sup>:
                                                      blue attribute.
                                                      Sol<sup>n</sup>:
.MODEL SMALL
.STACK 64
                                                       .MODEL SMALL
.DATA
                                                      .STACK 100
       SUM DW 0000
                                                      .DATA
.CODE
                                                       .CODE
       MOV AX,@DATA
                                                              MOV AX.@DATA
       MOV DS,AX
                                                              MOV DS,AX
       MOV AH,01H
                                                              CALL VIDEO MODE
       INT 21H
                                                              CALL CLEAR_SCREEN
       SUB AL,30H
                                                              CALL SET CURSOR
       MOV CH,00H
                                                              CALL DISPLAY
       MOV CL, AL
                                                              CALL EXIT_PROGRAM
       MOV AH,02H
                                                              VIDEO MODE PROC
       MOV DL,20H
                                                              MOV AH,00H
       INT 21H
                                                              MOV AL,00H
       MOV AX,01
                                                              INT 10H;set video mode
       MOV BX,03
       ADD AX,BX
                                                              VIDEO_MODE ENDP
       MOV SUM, AX
                                                              CLEAR SCREEN PROC
 BACK:MOV DX,AX
                                                              MOV AH,06H
       MOV AX, BX
                                                              MOV AL,00H
       MOV BX,DX
                                                              MOV BH, OBFH
       ADD SUM, BX
                                                              MOV CX,0000H
       ADD AX, BX; addition of series
                                                              MOV DX.1827H
       LOOP BACK
                                                              INT 10H; clear whole screen
       MOV AX,SUM; decimal display
  MOV BX,0000
                                                              CLEAR SCREEN ENDP
       MOV CX,0AH
                                                              SET CURSOR PROC
  DCE:MOV DX,0000H
                                                              MOV AH,02H
                                                              MOV DH,10
       DIV CX
       ADD DX,30H
                                                              MOV DL,05
       PUSH DX
                                                              MOV BH,00
       INC BX
                                                              INT 10H;set cursor at centre
       CMP AX,0000
                                                              RET
       JA DCE
                                                              SET CURSOR ENDP
                                                              DISPLAY PROC; display characters
       MOV CX,BX
       MOV AH,02
                                                              MOV CX,5FH
 SHOW:POP DX
                                                              MOV BH,32
       INT 21H
                                                        LABEL1:MOV AH,02H
       LOOP SHOW
                                                              MOV DL,BH
       MOV AH,4C
                                                              INT 21H
```

INT 21H

INC BH

LOOP LABEL1	JB NEXT2
RET	INC TEN
DISPLAY ENDP	SUB AX,0AH
EXIT_PROGRAM PROC	JMP NEXT1
MOV AH,4CH	NEXT2:MOV TEMP1,AX
INT 21H	MOV AX,0001H
RET	MUL HUN
EXIT_PROGRAM ENDP	MOV BX,100H
_ END	MUL BX
	MOV TEMP,AX
	MOV AX,0001H
11. You have an array of data in one table. Change	MUL TEN
each element to decimal ASCII and store it in the	MOV BX,10H
next table. Display the final result in the clear screen.	MUL BX
Sol ⁿ :	ADD AX,TEMP
.MODEL SMALL	ADD AX,TEMP1
STACK 100	MOV [DI],AX;storing BCD value of data
.DATA	INC SI
COUNT DB ?	INC DI
DATA1 DB 'kishor'	INC DI
	DEC COUNT
DATA2 DW 5 DUP(?)	JNZ BACK
SUM DW ?	LEA SI,DATA1
TEMP DW ?	MOV HUN,05
TEMP1 DW ?	BACK2:MOV AL,[SI] ;display BCD value of data
HUN DB 00	MOV AH,00
TEN DB 00	MOV BX,0000
.CODE	MOV CX,0AH
MOV AX,@DATA	DCE:MOV DX,0000H
MOV DS,AX MOV AH,06H;clear screen	DIV CX
	ADD DX,30H
MOV AL,00H	PUSH DX
MOV BH,07H	INC BX
MOV CX,0000H	CMP AX,0000
MOV DX,1827H	JA DCE
INT 10H;dos function	MOV CX,BX
LEA SI,DATA1	MOV AH,02
LEA DI,DATA2;for storing BCD eqvt	SHOW:POP DX
MOV DX,0000H	INT 21H
MOV COUNT,05H	LOOP SHOW
BACK:MOV HUN,00	MOV AH,02
MOV ALL COLL	MOV DL,20H
MOV AH,00H	INT 21H
MOV AL,[SI]	INC SI
L1:CMP AX,64H	DEC HUN
JB NEXT1	JNZ BACK2
INC HUN	MOV AH,4CH
SUB AX,64H	INT 21H
JMP L1	END
NEXT1:CMP AX,0AH	LIND

	JA DCE
	MOV CX,BX
12. Write an assembly language program to count	MOV AH,02
the number of vowels in a string entered by the user.	SHOW:POP DX
Display the result in decimal format.	INT 21H
Sol ⁿ :	LOOP SHOW
.MODEL SMALL	MOV AH,4CH
STACK 64	INT 21H
.DATA	END
MSG DB 10 DUP(?)	
.CODE	13. Write an assembly language program to convert
MOV AX,@DATA	the vowels to uppercase from a string entered by the
MOV DS,AX	user.
MOV AH,0AH	Sol ⁿ :
MOV CX,06H	
LEA DX,MSG	.MODEL SMALL
INT 21H;string input	.STACK 64
MOV SI,OFFSET MSG	.DATA
MOV CX,0006H	MSG DB 10 DUP(?)
MOV BX,0000H	.CODE
ADD SI,02H	MOV AX,@DATA
BACK:MOV AH,[SI]	MOV DS,AX
CMP AH,61H;compairing vowel	MOV AH,0AH
JE NEXT	MOV CX,06H
CMP AH,65H;compairing vowel	LEA DX,MSG
JE NEXT	INT 21H;input string
CMP AH,69H;compairing vowel	MOV SI,OFFSET MSG
JE NEXT	MOV CX,0006H
CMP AH,6FH;compairing vowel	MOV BX,0000H
JE NEXT	ADD SI,02H
CMP AH,75H;compairing vowel	BACK:MOV AH,[SI]
JE NEXT	CMP AH,61H;compairing vowel
JMP GOTO1	JE NEXT
NEXT:INC BX;count vowel	CMP AH,65H;compairing vowel
GOTO1:INC SI	JE NEXT
LOOP BACK	CMP AH,69H;compairing vowel
MOV AH,02	JE NEXT
MOV DL,0AH	
*	CMP AH,6FH;compairing vowel
INT 21H;print space	JE NEXT
;Display count in BCD	CMP AH,75H;compairing vowel
MOV AX,BX	JE NEXT
MOV BX,0000	JMP GOTO1
MOV CX,0AH	NEXT:AND AH,11011111B;convert to uppercase
DCE:MOV DX,0000H	MOV [SI],AH
DIV CX	GOTO1:INC SI
ADD DX,30H	LOOP BACK
PUSH DX	MOV [SI],'\$'
INC BX	CALL VIDEO_MODE
CMP AX,0000	CALL CLEAR_SCREEN

CALL SET CURSOR MSG DB 10 DUP(?) **CALL DISPLAY** .CODE CALL EXIT PROGRAM MOV AX,@DATA VIDEO MODE PROC **MOV DS.AX** MOV AH,00H MOV AH, OAH MOV AL,00H MOV CX,06H INT 10H;set video mode LEA DX.MSG RFT INT 21H; string input VIDEO MODE ENDP **MOV SI, OFFSET MSG CLEAR SCREEN PROC** ADD SI.02H MOV AH,06H MOV CX,0006H MOV AL,00H BACK:MOV AL,[SI] MOV BH,07H CMP AL,61H MOV CX,0000H JB OVER MOV DX,1827H CMP AL,7AH INT 10H; clear screen JA OVER **RET** AND AL,110111111B; convert to uppercase **CLEAR SCREEN ENDP** OVER:MOV [SI],AL **INC SI** SET CURSOR PROC MOV AH,02H LOOP BACK MOV DH,12 MOV [SI],'\$' MOV DL,20 CALL VIDEO MODE MOV BH,00 CALL CLEAR_SCREEN INT 10H;set cursor at centre CALL SET CURSOR RET **CALL DISPLAY** CALL EXIT PROGRAM SET CURSOR ENDP **DISPLAY PROC** VIDEO MODE PROC LEA DX,MSG MOV AH,00H ADD DX.02H MOV AL,00H MOV AH,09H INT 10H;set video mode INT 21H; display string RET VIDEO MODE ENDP RET **DISPLAY ENDP CLEAR SCREEN PROC EXIT PROGRAM PROC** MOV AH,06H MOV AH,4CH MOV AL,00H **INT 21H** MOV BH,07H **RET** MOV CX,0000H EXIT_PROGRAM ENDP **MOV DX,1827H END** INT 10H; clear screen RET **CLEAR SCREEN ENDP** 14. Write an assembly language program to get SET CURSOR PROC string input from the user convert it to capital case MOV AH,02H display the attributed string at the center of the MOV DH,12 defined window (2,10 to 22,70). MOV DL,20 MOV BH,00 INT 10H;set cursor at co ordinate .MODEL SMALL .STACK 64

SET_CURSOR ENDP

Solⁿ:

.DATA

MOV AH.00H **DISPLAY PROC** MOV AL,00H LEA DX,MSG ADD DX,02H INT 10H;set video mode RFT MOV AH.09H INT 21H; display string VIDEO MODE ENDP RET **CLEAR SCREEN PROC DISPLAY ENDP** MOV AH.06H **EXIT PROGRAM PROC** MOV AL,00H MOV AH,4CH MOV BH,07H **INT 21H** MOV CX.0000H **MOV DX,1827H** RET **EXIT PROGRAM ENDP** INT 10H; clear screen **END RET** CLEAR_SCREEN ENDP 15. Write an assembly language program to get SET CURSOR PROC string input from the user convert it to lower case MOV AH,02H display the attributed string at the lower left corner MOV DH,10 of the defined window (3, 10 to 21, 10). MOV DL,02 Solⁿ: MOV BH,00 INT 10H;set cursor at co ordinate .MODEL SMALL .STACK 64 SET CURSOR ENDP .DATA **DISPLAY PROC** MSG DB 10 DUP(?) LEA DX,MSG .CODE ADD DX,02H MOV AX,@DATA MOV AH,09H MOV DS,AX INT 21H; display string MOV AH, 0AH **RFT** MOV CX,06H **DISPLAY ENDP** LEA DX,MSG **EXIT PROGRAM PROC** INT 21H; input string MOV AH.4CH MOV SI, OFFSET MSG **INT 21H** ADD SI,02H RET MOV CX,0006H EXIT_PROGRAM ENDP BACK: MOV AL, [SI] **END** CMP AL,41H JB OVER CMP AL,5AH 16. Write an assembly language program that takes a JA OVER string input from user and clear the screen and move OR AL,00100000B; convert to lowercase the string from right edge of the screen to left edge. OVER:MOV [SI],AL The movement should be noticeable. **INC SI** Solⁿ: LOOP BACK MOV [SI],'\$' .MODEL SMALL CALL VIDEO_MODE .STACK 200 CALL CLEAR_SCREEN .DATA CALL SET_CURSOR MSG DB 'KISHOR\$' **CALL DISPLAY** TEMP1 DW 65535; for delay CALL EXIT PROGRAM TEMP2 DW 65535; for delay VIDEO MODE PROC TEMP3 DW 65535; for delay

```
TEMP4 DW 65535; for delay
                                                            INT 21H; display string
      TEMP5 DW 65535; for delay
                                                            RFT
      LOC DW 0021H
                                                            DISPLAY ENDP
.CODF
                                                            EXIT PROGRAM PROC
      MOV AX,@DATA
                                                            MOV AH,4CH
      MOV DS,AX
                                                            INT 21H
      CALL VIDEO MODE
                                                            RFT
      MOV CX,25
                                                            EXIT PROGRAM ENDP
 BACK: CALL CLEAR SCREEN
                                                            END
      MOV DX.LOC
                                                     17. Write an assembly language program to generate
      CALL SET CURSOR
                                                     a multiplication table of any number entered by the
      CALL DISPLAY
                                                     user. Display the table in the screen.
  BACK3:DEC TEMP3; delay loop
                                                     Sol<sup>n</sup>:
      JNZ BACK3
 BACK1:DEC TEMP1;delay loop
                                                     .MODEL SMALL
      JNZ BACK1
                                                     .STACK 64
 BACK2:DEC TEMP2;delay loop
                                                     .DATA
      JNZ BACK2
                                                            NUM DB 2 DUP(?)
 BACK4:DEC TEMP4; delay loop
                                                            TEMP DW?
      JNZ BACK4
                                                            TEMP2 DW?
 BACK5:DEC TEMP5;delay loop
                                                            TEMP3 DW?
      JNZ BACK5
                                                            TEMP4 DW?
      SUB LOC,02
                                                     .CODF
      LOOP BACK
                                                            MOV AX,@DATA
      CALL EXIT PROGRAM
                                                            MOV DS,AX
      VIDEO MODE PROC
                                                            MOV CX,0002
      MOV AH,00H
                                                            LEA DI.NUM
      MOV AL,00H
                                                       KJ:MOV AH,01H;input two digit
      INT 10H:set video mode
                                                            INT 21H
      RET
                                                            MOV [DI],AL
      VIDEO MODE ENDP
                                                            INC DI
      CLEAR SCREEN PROC
                                                            LOOP KJ
      MOV AH,06H
                                                            MOV AH,02H
      MOV AL,00H
                                                            MOV DL,20H
      MOV BH,07H
                                                            INT 21H
      MOV CX,0000H
                                                            MOV SI, OFFSET NUM
      MOV DX,1827H
                                                            MOV DH,[SI]
      INT 10H; clear screen
                                                            INC SI
      RET
                                                            MOV DL,[SI]
      CLEAR_SCREEN ENDP
                                                            SUB DH,30H; converting to eqvt HEX
      SET_CURSOR PROC
                                                            SUB DL,30H
      MOV AH,02H
                                                            MOV CL,04H
      MOV BH,00
                                                            ROL DH,CL
      INT 10H;set cursor
                                                            OR DH,DL
      RET
                                                            MOV CL, DH
      SET CURSOR ENDP
                                                            MOV CH,00H
      DISPLAY PROC
                                                       LABELS:CMP CL,10H
      MOV AH,09H
                                                            JB NEXT
      LEA DX,MSG
```

INC CH

SUB CL,10H	MOV TEMP2,DX
JMP LABELS	DCE: MOV DX,0000H
NEXT:ADD CL,0AH	DIV CX
DEC CH	ADD DX,30H
JNZ NEXT	PUSH DX
MOV CH,00	INC BX
MOV TEMP,CX;HEX eqvt to temp	CMP AX,0000
CALL VIDEO_MODE	JA DCE
CALL CLEAR_SCREEN	MOV CX,BX
MOV CX,000AH	MOV AH,02H
MOV BL,01H	SHOW: POP DX
MOV DH,00H	INT 21H
MOV DL,00H;set cursor at top at first	LOOP SHOW
LABEL1:CALL SET_CURSOR	MOV DX,TEMP2
MOV AX,TEMP	MOV CX,TEMP3
MUL BL;multiplied at ax	MOV BX,TEMP4
CALL DECIMAL	RET
INC BL	DECIMAL ENDP
LOOP LABEL1	END
MOV AH,4CH	LIND
INT 21H	
VIDEO_MODE PROC	18. Write a program to find the HCF of two
MOV AH,00H	unsigned 16-bit numbers.
MOV AL,00H	Sol ⁿ :
·	
INT 10H;set video mode RET	.MODEL SMALL
	.STACK 64
VIDEO_MODE ENDP	.DATA
CLEAR_SCREEN PROC	NUM1 DW 0005
MOV AH,06H	NUM2 DW 0015
MOV AL,00H	TEMP1 DW ?
MOV BH,07H	TEMP2 DW ?
MOV CX,0000H	SUM DW ?
MOV DX,1827H	.CODE
INT 10H;clear whole screen	MOV AX,@DATA
RET	MOV DS,AX
CLEAR_SCREEN ENDP	MOV AX,NUM1
SET_CURSOR PROC	MOV BX,NUM2
MOV AH,02H	CMP AX,BX
MOV BH,00	JA NEXT
INC DH	XCHG AX,BX;find greatest
MOV DL,00	MOV SUM,BX
INT 10H;set cursor	NEXT:MOV DX,0000H;finding HCF
RET	MOV TEMP1,AX
SET_CURSOR ENDP	MOV TEMP2,BX
DECIMAL PROC; display eqvt BCD	DIV BX
MOV TEMP4,BX	CMP DX,0000H
MOV BX,0000H	JE LABEL1
MOV TEMP3,CX	DEC TEMP2
MOV CX,000AH	MOV AY TEMP1

MOV AX,TEMP1

MC	OV BX,TEMP2	CMP AX,BX
JMI	P NEXT	JA NEXT
LABEL1:M	OV AX,SUM	XCHG AX,BX;find greatest
MC	OV BX,TEMP2	MOV SUM,BX
MC	OV DX,0000H	NEXT:MOV DX,0000H
DIV	' BX	MOV TEMP1,AX
CM	P DX,0000H	MOV TEMP2,BX
JE L	ABEL2	DIV BX
DEC	C TEMP2	CMP DX,0000H
MC	OV AX,TEMP1	JE LABEL1
MC	OV BX,TEMP2	DEC TEMP2
JMI	P NEXT	MOV AX,TEMP1
LABEL2:N	IOV AX,TEMP2;HCF	MOV BX,TEMP2
;BC	D display	JMP NEXT
MC	OV BX,0000	LABEL1:MOV AX,SUM
MC	OV CX,0AH	MOV BX,TEMP2
DCE:MO	V DX,0000H	MOV DX,0000H
DIV	CX	DIV BX
ADI	D DX,30H	CMP DX,0000H
PU:	SH DX	JE LABEL2
INC	BX	DEC TEMP2
CM	P AX,0000	MOV AX,TEMP1
JA [DCE	MOV BX,TEMP2;HCF
MC	OV CX,BX	JMP NEXT
MC	OV AH,02	LABEL2:MOV AX,NUM1
SHOW:PO	OP DX	MUL NUM2
INT	[•] 21H	MOV DX,0000H
LOC	OP SHOW	DIV TEMP2;LCM
MC	OV AH,4CH	BCD display
INT	¹ 21H	MOV BX,0000
END		MOV CX,0AH
		DCE:MOV DX,0000H
10 W.:	The second of th	DIV CX
	program to find the LCM of two 6-bit numbers.	ADD DX,30H
Sol ⁿ :	5-oit numbers.	PUSH DX
501.		INC BX
.MODEL SM	1ALL	CMP AX,0000
.STACK 64		JA DCE
.DATA		MOV CX,BX
NU	M1 DW 0005	MOV AH,02
NU	M2 DW 0003	SHOW:POP DX
TEN	MP1 DW ?	INT 21H
TEN	MP2 DW ?	LOOP SHOW
SUI	M DW ?	MOV AH,4CH
.CODE		INT 21H
МС	OV AX,@DATA	END
MC	OV DS,AX	
МС	OV AX,NUM1	
MC	OV BX,NUM2	

20. Write a program that takes a string from a user INT 21H and displays each word in a new line diagonally JMP BACK from upper left towards bottom right in a clear GO1:CALL EXIT PROGRAM screen. If the string is "Programming in Assembly VIDEO MODE PROC Language is Fun", it should be displayed as follows: MOV AH,00H MOV AL,02H **Programming** in INT 10H:set video mode Assembly RFT Language VIDEO MODE ENDP is **CLEAR SCREEN PROC** Fun MOV AH,06H Solⁿ: MOV AL,00H MOV BH,07H .MODEL SMALL MOV CX,0000H .STACK 100 MOV DX,1827H .DATA INT 10H; clear screen MSG DB 42 DUP(?) RET TEMP DB? **CLEAR SCREEN ENDP** .CODE SET CURSOR PROC MOV AX,@DATA MOV AH,02H MOV DS,AX MOV BH,00 LEA SI, MSG INT 10H;set cursor as co ordinate MOV [SI],41 RET MOV AH, 0AH SET CURSOR ENDP MOV CX,0040 **EXIT PROGRAM PROC** LEA DX,MSG MOV AH.4CH INT 21H;input string INT 21H **MOV SI, OFFSET MSG** RFT ADD SI,02 **EXIT PROGRAM ENDP** MOV [SI+0041],'\$' **END** CALL VIDEO MODE CALL CLEAR SCREEN 21. Write an assembly language program that **MOV DX,0205H** calculates the sum of the elements of a 3 by 3 MOV BL,00H matrix. The 3 by 3 matrix is entered by the user and LABEL1:CALL SET_CURSOR;set cursor at every the sum should be displayed on the PC screen. The program should be able to handle unsigned and space signed numbers. MOV AH,02H Solⁿ: MOV TEMP, DL BACK: MOV DL, [SI] .MODEL SMALL CMP DL,20H; check space .STACK 64 JNE NEXT .DATA INC DH NUM1 DB 1,2,3,4,5,6,7,8,9 MOV DL, TEMP NUM2 DB 1,2,3,4,5,6,7,8,9 INC DL NUM3 DB 9 DUP(?) **INC SI** TEMP1 DB 3 JMP LABEL1 TEMP2 DB 3 NEXT:CMP DL,'\$';check end point .CODE JE GO1 MOV AX,@DATA

MOV DS,AX

INC SI

INC TEMP

LEA SI,NUM1	
LEA BX,NUM2	.MODEL SMALL
LEA DI,NUM3	.STACK 64
MOV CX,09	.DATA
BACK:MOV AL,[SI]	SUM DW ?
ADD AL,[BX]	NUM DB 2 DUP(?)
MOV [DI],AL	.CODE
INC SI	MOV AX,@DATA
INC DI	MOV DS,AX
INC BX	MOV CX,0002
LOOP BACK	LEA DI,NUM
LEA SI,NUM3	KJ:MOV AH,01H;input two digit no
BACK2:MOV BX,0000	INT 21H
MOV CX,000AH	MOV [DI],AL
MOV AL,[SI]	INC DI
MOV AH,00	LOOP KJ
DDD: MOV DX,0000	MOV AH,02H
DIV CX	MOV DL,20H
ADD DX,30H	INT 21H;print space
PUSH DX	MOV SI, OFFSET NUM
INC BX	MOV DH,[SI]
CMP AX,0000	INC SI
JA DDD	MOV DL,[SI]
MOV CX,BX	SUB DH,30H
MOV AH,02	SUB DL,30H
SHOW:POP DX	MOV CL,04H
INT 21H	ROL DH,CL
LOOP SHOW	OR DH,DL
INC SI	MOV AX,0000H
MOV AH,02	MOV CL,DH
MOV DL,20H	MOV CH,00H
INT 21H	LABELS:CMP CL,10H
DEC TEMP1	JB NEXT
JNZ BACK2	INC CH
MOV TEMP1,03	SUB CL,10H
MOV AH,02	JMP LABELS
MOV DL,0AH	NEXT:ADD CL,0AH;converted to eqvt HEX
INT 21H	DEC CH
DEC TEMP2	JNZ NEXT
JNZ BACK2	BACK:ADD AX,CX
MOV AH,4CH	LOOP BACK
INT 21H	MOV SUM,AX;adding
END	MOV AX,SUM;BCD display
	MOV BX,0000
	MOV CX,0AH
22. Write an assembly language program to find the	DCE:MOV DX,0000H
sum of numbers from (1) to (n). Read (n) from the	DIV CX
user and display the sum in decimal format (also try to display the sum in Hexadecimal format)	ADD DX,30H
Sol ⁿ :	PUSH DX
teres and the second of the se	

INC BX	MOV CX,0002
CMP AX,0000	LEA DI,NUM
JA DCE	KJ:MOV AH,01H;input 2 digit no
MOV CX,BX	INT 21H
MOV AH,02	MOV [DI],AL
SHOW:POP DX	INC DI
INT 21H	LOOP KJ
LOOP SHOW	MOV AH,02H
MOV AH,02	MOV DL,20H
MOV DL,20H	INT 21H
INT 21H;print space	MOV SI,OFFSET NUM
MOV AX,SUM;HEX display	MOV DH,[SI]
MOV BX,0000	INC SI
MOV CX,10H	MOV DL,[SI]
DCE1:MOV DX,0000H	SUB DH,30H
DIV CX	SUB DL,30H
CMP DX,000AH	MOV CL,04H
JA L3	ROL DH,CL
ADD DX,30H	OR DH,DL
JMP L2	MOV AX,0000H
L3:ADD DX,37H	MOV CL,DH
L2:PUSH DX	MOV CH,00H
INC BX	LABELS:CMP CL,10H
CMP AX,0000	JB NEXT
JA DCE1	INC CH
MOV CX,BX	SUB CL,10H
MOV AH,02	JMP LABELS
SHOW1:POP DX	NEXT:CMP CH,00H
INT 21H	JA KJJ
LOOP SHOW1	JMP JK
MOV AH,4C	KJJ:ADD CL,0AH;converted to eqvt HEX
INT 21H	DEC CH
END	JNZ KJJ;INPUT
	JK:MOV BX,0000H
	BACK:MOV AX,CX
23. Write a program to find the sum of the following	INC AX
series up to the terms specified by the user and	MOV DL,AL
display the result in decimal format. (also try to	MUL DL
display the sum in HEX format) $(2*4) + (3*6) + (4*8) + \dots$ to (n) terms	ADD BX,AX;addition of series
Sol ⁿ :	LOOP BACK
501.	MOV DX,0000H
.MODEL SMALL	
.STACK 64	•
.DATA	·
SUM DW ?	
NUM DB 2 DUP(?)	·
.CODE	·
MOV AX,@DATA	
MOV DS,AX	
STACK 64 .DATA SUM DW ? NUM DB 2 DUP(?) .CODE	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 DS.AX	MOV CX.0AH

DCE:MOV DX,0000H	FALSE DB 'THE NUMBER IS NOT PRIME\$'
DIV CX	SUM DW ?
ADD DX,30H	NUM DB 2 DUP(?)
PUSH DX	TEMP DW ?
INC BX	.CODE
CMP AX,0000	MOV AX,@DATA
JA DCE	MOV DS,AX
MOV CX,BX	MOV CX,0002
MOV AH,02	LEA DI,NUM
SHOW:POP DX	KJ:MOV AH,01H
INT 21H	INT 21H
LOOP SHOW	MOV [DI],AL
MOV AH,02	INC DI
MOV DL,20H	LOOP KJ
INT 21H	MOV AH,02H
MOV AX,SUM;HEX;display	MOV DL,20H
MOV BX,0000	INT 21H
MOV CX,10H	MOV SI,OFFSET NUM
DCE1:MOV DX,0000H	MOV DH,[SI]
DIV CX	INC SI
CMP DX,000AH	MOV DL,[SI]
JA L3	SUB DH,30H
ADD DX,30H	SUB DL,30H
JMP L2	MOV CL,04H
L3:ADD DX,37H	ROL DH,CL
L2:PUSH DX	OR DH,DL
INC BX	MOV AX,0000H
CMP AX,0000	MOV CL,DH
JA DCE1	MOV CH,00H
MOV CX,BX	LABELS:CMP CL,10H
MOV AH,02	JB NEXT
SHOW1:POP DX	INC CH
INT 21H	SUB CL,10H
LOOP SHOW1	JMP LABELS
	NEXT:CMP CH,00H
MOV AH,4C INT 21H	·
	JA KJJ
END	JMP JK
24. Writ a program to find out if a number entered	KJJ:ADD CL,0AH
by the user is prime or not. If the number is prime,	DEC CH
the output on the screen should say "The number is a	JNZ KJJ;INPUT
prime number", else if the number is not prime, the	JK:MOV AX,CX
output on the screen should say "The number is not a	MOV TEMP,AX
prime number".	DEC CX
Sol ⁿ :	DEC CX
MODEL SMALL	BACKA:MOV DX,CX
.MODEL SMALL	INC DX
STACK 100	DIV DL
.DATA	CMP AH,00H
TRUE DB 'THE NUMBER IS PRIME\$'	JE YES

JE YES

MOV AX, TEMP JNE DIVIDE1 LOOP BACKA **DIVIDE2: POP DX** JMP NO ADD DL,30H YES:LEA DX.FALSE MOV AH,02H MOV AH,09H **INT 21H INT 21H** LOOP DIVIDE2 JMP EN MOV DL.'/' NO:LEA DX,TRUE MOV AH,02H MOV AH,09H **INT 21H INT 21H** MOV DX.0 **EN:MOV AH,4CH** POP AX **INT 21H** MOV CX,0 **END** MOV BX,10 **DIVIDE3: DIV BX PUSH DX** 25. Write a program that retrieves the system date ADD CX,1 and time and displays the information on the PC MOV DX,0 screen. CMP AX,0 Solⁿ: JNE DIVIDE3 **DIVIDE4: POP DX** .MODEL SMALL ADD DL,30H .STACK 100H MOV AH,02H .DATA INT 21H MSG DB "TODAY'S DATE IS ",'\$' LOOP DIVIDE4 MSG1 DB "TODAY'S TIME IS ",'\$' MOV DL,'/' .CODE MOV AH.02H MOV AX,@DATA **INT 21H** MOV DS,AX MOV DX.0 MOV DX, OFFSET MSG POP AX MOV AH,09H MOV CX,0 **INT 21H** MOV BX.10 MOV AH,2AH **DIVIDE5: DIV BX** INT 21H **PUSH DX PUSH CX** ADD CX,1 MOV CX,0 MOV DX,0 MOV CL, DL CMP AX,0 **PUSH CX** JNE DIVIDE5 MOV CL, DH **DIVIDE6: POP DX PUSH CX** ADD DL,30H MOV DH,0 MOV AH,02H MOV DX,0 INT 21H POP AX **LOOP DIVIDE6** MOV CX.0 MOV DL,0AH MOV BX,10 MOV AH,02H **DIVIDE1: DIV BX INT 21H PUSH DX** MOV DX, OFFSET MSG1 ADD CX,1 MOV AH,09H MOV DX,0 **INT 21H** CMP AX,0 MOV AH,2CH

INT 21H
MOV DL,DH
MOV DH,O
PUSH DX
MOV DL,CL
PUSH DX
MOV DL,CH
PUSH DX
MOV DH,O
MOV DH,O
MOV DX,O
POP AX
MOV CX,O
MOV BX,10

PUSH DX
ADD CX,1
MOV DX,0
CMP AX,0
JNE TDIVIDE1
TDIVIDE2: POP DX
ADD DL,30H
MOV AH,02H

TDIVIDE1: DIV BX

LOOP TDIVIDE2 MOV DL,'-'

INT 21H

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

CONSTIDB ' 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

EXAM SOLUTION 8086

 Write an assembly program to read a string from the user and display vowels and consonant separately.

Solⁿ:

.MODEL SMALL

.STACK 64

.DATA

VOWEL DB ' VOWELS ARE=\$'

```
MOV DL,AL
          INT 21H
                                                               INT 21H
          MOV DL,0AH
                                                               INC SI
          INT 21H
                                                               JMP BACK
          LEA DX,VO
                                                           L1:MOV AH,03H
          MOV AH,09H
                                                               INT 10H
          INT 21H
                                                               INC DH
          MOV AH,02
                                                               MOV DL.20
          MOV DL,0AH
                                                               MOV AH,02H
          INT 21H
                                                               MOV BH,00
          LEA DX, CONST
                                                               INT 10H
          MOV AH,09H
                                                               INC SI
          INT 21H
                                                               JMP BACK
          LEA DX,CO
                                                          NEXT:MOV AH,4CH
          MOV AH,09H
                                                               INT 21H
          INT 21H
                                                               VIDEO_MODE PROC
          MOV AH,4CH
                                                               MOV AX,0000H
          INT 21H
                                                               INT 10H
   END
                                                               RET
                                                               VIDEO MODE ENDP
2. Write a program in 8086 to read a string
   and display each word in separate line in
                                                               CLEAR SCREEN PROC
   centre of screen.
                                                               MOV AH,06H
   Sol<sup>n</sup>:
                                                               MOV AL,00H
                                                               MOV BH,07H
                                                               MOV CX,0000
   .MODEL SMALL
                                                               MOV DX,1827H
   .STACK 64
                                                               INT 10H
   .DATA
                                                               RET
          STRING DB 41,41 DUP(?),'$'
                                                               CLEAR_SCREEN ENDP
   .CODE
          MOV AX,@DATA
                                                               SET CURSOR PROC
          MOV DS,AX
                                                               MOV AH,02H
          LEA DX,STRING
                                                               MOV DX,0C14H
          MOV AH,0AH
                                                               MOV BH,00
          MOV CX,39
                                                               INT 10H
          INT 21H
                                                               RET
          LEA SI, STRING
                                                               SET_CURSOR ENDP
          ADD SI,02
          CALL VIDEO MODE
                                                        END
          CALL CLEAR SCREEN
          CALL SET_CURSOR
                                                     3. Write an assembly program to read a
      BACK:MOV AL,[SI]
                                                        text from keyword, convert the text into
          CMP AL,'$'
          JE NEXT
                                                        uppercase and display on the clear
          CMP AL,' '
                                                        screen.
          JE L1
                                                        Sol<sup>n</sup>:
```

MOV AH,02H

```
.MODEL SMALL
                                                          JB NEXT
.STACK 64
                                                          CMP AL,7AH
.DATA
                                                          JA NEXT
      STRING DB 15,15 DUP(?),'$'
                                                          AND AL,11011111B
.CODE
                                                          MOV [SI],AL
      MOV AX,@DATA
                                                     NEXT:INC SI
      MOV DS,AX
                                                          LOOP BACK
      LEA DX.STRING
                                                          RET
      MOV CX,14
                                                          UPPERCASE ENDP
      MOV AH,0AH
                                                          DISPLAY PROC
      INT 21H
      CALL VIDEO_MODE
                                                          LEA DX,STRING
      CALL CLEAR SCREEN
                                                          ADD DX,02
      CALL SET_CURSOR
                                                          MOV AH,09H
      CALL UPPERCASE
                                                          INT 21H
      CALL DISPLAY
                                                          RET
      MOV AH,4CH
                                                          DISPLAY ENDP
      INT 21H
                                                   END
      VIDEO MODE PROC
                                                4. Write a program to read string and
       MOV AX,0000
                                                   display only the alphabetic characters
      INT 10H
                                                   from the string in clear screen.
      RET
                                                   Sol<sup>n</sup>:
      VIDEO_MODE ENDP
                                                   .MODEL SMALL
      CLEAR SCREEN PROC
                                                   .STACK 64
       MOV AH,06H
                                                   .DATA
      MOV AL,00H
                                                          STRING DB 15,15 DUP(?),'$'
      MOV CX,0000
                                                          ALPHA DB 15 DUP(?),'$'
      MOV DX,1827H
                                                   .CODE
       MOV BH,07H
      INT 10H
                                                          MOV AX,@DATA
      RET
                                                          MOV DS,AX
      CLEAR SCREEN ENDP
                                                          LEA DX,STRING
                                                          MOV CX,14
      SET CURSOR PROC
                                                          MOV AH,0AH
       MOV AH,02H
                                                          INT 21H
      MOV DH,12
                                                          CALL VIDEO MODE
      MOV DL,15
                                                          CALL CLEAR SCREEN
      MOV BH,00
                                                          CALL SET CURSOR
      INT 10H
                                                          CALL ALPHAS
      RET
                                                          CALL DISPLAY
      SET_CURSOR ENDP
                                                          MOV AH,4CH
       UPPERCASE PROC
                                                          INT 21H
      LEA SI, STRING
      ADD SI,02
                                                          VIDEO MODE PROC
      MOV CX,0012
                                                          MOV AX,0000
 BACK:MOV AL,[SI]
                                                          INT 10H
```

CMP AL,61H

RET RET VIDEO MODE ENDP **DISPLAY ENDP** END **CLEAR SCREEN PROC** 5. Write a program in 8086 to read string. MOV AH,06H Display each word in separate lines in MOV AL,00H cleared lines in a cleared screen, count MOV CX,0000 how many words are there and display MOV DX,1827H the count. MOV BH,07H Solⁿ: INT 10H RET .MODEL SMALL CLEAR_SCREEN ENDP .STACK 64 .DATA SET CURSOR PROC STRING DB 41,41 DUP(?),'\$' MOV AH,02H **TEMP DW 0000** MOV DH,12 .CODE MOV DL,15 MOV AX,@DATA MOV BH,00 MOV DS,AX **INT 10H** LEA DX,STRING RET MOV AH,0AH SET CURSOR ENDP MOV CX,39 **INT 21H** LEA SI, STRING **ALPHAS PROC** ADD SI,02 LEA SI, STRING CALL VIDEO_MODE ADD SI,02 CALL CLEAR_SCREEN LEA DI, ALPHA CALL SET_CURSOR ADD DI,02 BACK:MOV AL,[SI] MOV CX,0012 CMP AL,'\$' BACK:MOV AL,[SI] JE NEXT CMP AL,61H CMP AL,' ' JB NEXT JE L1 CMP AL,7AH MOV AH,02H MOV DL,AL JA NEXT **INT 21H** MOV [DI],AL **INC SI** INC DI JMP BACK **NEXT:INC SI** L1:INC TEMP **LOOP BACK** MOV AH,03H **RET INT 10H ALPHAS ENDP** INC DH MOV DL,20 **DISPLAY PROC** MOV AH,02H LEA DX,ALPHA MOV BH,00 ADD DX,02 INT 10H **INC SI** MOV AH,09H

INT 21H

JMP BACK

NEXT:INC TEMP	6.	WAP in 8086 to find largest and smalles
MOV AX,TEMP		and display them.
CALL VALUES1		Sol ⁿ :
MOV AH,4CH		
INT 21H		.MODEL SMALL
		.STACK 64
VIDEO_MODE PROC		.DATA
MOV AX,0000H		
INT 10H		DATA1 DW
RET		2214,5231,65535,4532,3219,55555,773
VIDEO_MODE ENDP		1,8399,9911,1111
		LARGEST DW ?
CLEAR_SCREEN PROC		SMALLEST DW ?
MOV AH,06H		LRG DB 'LARGEST IS\$'
MOV AL,00H		SML DB 'SMALLEST IS\$'
MOV BH,07H		.CODE
MOV CX,0000		MOV AX,@DATA
MOV DX,1827H		MOV DS,AX
INT 10H RET		LEA SI,DATA1
CLEAR_SCREEN ENDP		MOV CX,000AH
CLEAN_SCREEN ENDP		MOV AX,[SI]
SET_CURSOR PROC		ADD SI,02
MOV AH,02H		BACK:MOV BX,[SI]
MOV DX,0C14H		CMP AX,BX
MOV BH,00		JA NEXT
INT 10H		MOV AX,BX
RET		NEXT:ADD SI,02
SET_CURSOR ENDP		LOOP BACK
VALUES1 PROC		MOV LARGEST,AX
MOV BX,0000		LEA SI,DATA1
MOV CX,0AH		MOV CX,000AH
DCE:MOV DX,0000H		MOV AX,[SI]
DIV CX		ADD SI,02
ADD DX,30H		BACK1:MOV BX,[SI]
PUSH DX		CMP AX,BX
INC BX		JB NEXT1
CMP AX,0000		MOV AX,BX
JA DCE		NEXT1:ADD SI,02
MOV CX,BX		LOOP BACK1
MOV AH,02		MOV SMALLEST,AX
SHOW:POP DX		CALL JOKE1
INT 21H		CALL NEW LINE
LOOP SHOW		MOV AX,LARGEST
RET		CALL DISPLAY
VALUES1 ENDP		CALL NEW LINE
END		CALL JOKE2
		CALL JUNEZ

CALL NEW LINE **LOOP SHOW MOV AX, SMALLEST** RET **CALL DISPLAY DISPLAY ENDP** END MOV AH,4CH 7. WAP in 8086 to convert vowels to INT 21H uppercase from a string entered by the user and display the converted string in **JOKE1 PROC** a new line. Also count no of uppercase LEA DX,LRG in converted string and display count. MOV AH,09H Solⁿ: INT 21H RET .MODEL SMALL **JOKE1 ENDP** .STACK 64 JOKE2 PROC .DATA STRING DB 17,17 DUP(?),'\$' LEA DX,SML **COUNT1 DW 0000** MOV AH,09H .CODF **INT 21H** MOV AX,@DATA **RET MOV DS,AX JOKE2 ENDP** MOV AH,0AH LEA DX,STRING **NEW LINE PROC** MOV CX,0015 MOV AH,03H **INT 21H** INT 10H CALL NEW LINE INC DH **CALL UPPERCASE** MOV DL,00 MOV AH,02H **CALL DISPLAY CALL COUNTS** INT 10H CALL NEW LINE **RET CALL DECIMAL NEW LINE ENDP** MOV AH,4CH **INT 21H DISPLAY PROC** MOV BX,0000H **NEW LINE PROC** MOV CX,000AH MOV AH,03H DCE: MOV DX,0000H INT 10H DIV CX INC DH ADD DX,30H MOV DL,00 **PUSH DX** MOV AH,02H INC BX **INT 10H** CMP AX,0000 **RET** JA DCE NEW_LINE ENDP MOV CX,BX MOV AH,02H **UPPERCASE PROC** SHOW: POP DX

INT 21H

LEA SI, STRING

ADD SI,02

```
MOV CX,15
                                                       MOV BX,0000H
BACK: MOV AL,[SI]
                                                       MOV CX,000AH
     CMP AL,'a'
                                                  DCE: MOV DX,0000H
                                                       DIV CX
     JE OVER
     CMP AL, 'e'
                                                       ADD DX,30H
     JE OVER
                                                       PUSH DX
     CMP AL,'i'
                                                       INC BX
     JE OVER
                                                       CMP AX,0000
     CMP AL,'o'
                                                       JA DCE
     JE OVER
                                                       MOV CX,BX
     CMP AL,'u'
                                                       MOV AH,02H
     JE OVER
                                                 SHOW: POP DX
     JMP GO
                                                       INT 21H
OVER: AND AL,11011111B
                                                       LOOP SHOW
GO: MOV [SI],AL
                                                       RET
     INC SI
                                                       DECIMAL ENDP
     LOOP BACK
                                                END
     RET
                                             8. WAP in 8086 to read string and count no
     UPPERCASE ENDP
                                                of vowels and display string & count in
                                                clear screen.
     DISPLAY PROC
                                                Sol<sup>n</sup>:
     LEA DX,STRING
     ADD DX,01H
                                                .MODEL SMALL
     MOV AH,09H
                                                .STACK 64
     INT 21H
                                                .DATA
     RET
     DISPLAY ENDP
                                                       MSG DB 10,10 DUP(?),'$'
                                                       COUNT1 DW 0000
                                                .CODE
     COUNTS PROC
                                                       MOV AX,@DATA
     LEA SI, STRING
                                                       MOV DS,AX
     ADD SI,02
                                                       MOV AH, OAH
     MOV CX,15
                                                       MOV CX,09H
LA1: MOV AL,[SI]
                                                       LEA DX,MSG
     CMP AL,41H
                                                       INT 21H
     JB NEXT
     CMP AL,5AH
                                                       MOV AH,03H
     JA NEXT
                                                       INT 10H
     INC COUNT1
NEXT: INC SI
                                                       INC DH
                                                       MOV AH,02H
     LOOP LA1
                                                       MOV DL,00
     RET
                                                       INT 10H
     COUNTS ENDP
                                                       MOV SI, OFFSET MSG
     DECIMAL PROC
     MOV AX,COUNT1
                                                       MOV CX,0006H
```

MOV BX,0000H

```
ADD SI,02H
                                                    CLEAR SCREEN PROC
BACK:MOV AH,[SI]
                                                    MOV AH,06H
   CMP AH,61H
                                                    MOV AL,00H
   JE NEXT
                                                    MOV BH,07H
   CMP AH,65H
                                                    MOV CX,0000
   JE NEXT
                                                    MOV DX,1827H
   CMP AH,69H
                                                    INT 10H
   JE NEXT
                                                    RET
   CMP AH,6FH
                                                    CLEAR SCREEN ENDP
   JE NEXT
                                                    SET CURSOR PROC
   CMP AH,75H
   JE NEXT
                                                    MOV AH,02H
   JMP GOTO1
                                                    MOV DX,0C14H
NEXT:INC BX
                                                    MOV BH,00
GOTO1:INC SI
                                                    INT 10H
    LOOP BACK
                                                    RET
    MOV COUNT1,BX
                                                    SET CURSOR ENDP
    CALL VIDEO MODE
    CALL CLEAR SCREEN
                                                    DECIMAL PROC
    CALL SET_CURSOR
                                                    MOV BX,0000H
                                                    MOV CX,000AH
                                               DCE: MOV DX,0000H
    LEA DX,MSG
   ADD DX,02
                                                    DIV CX
    MOV AH,09H
                                                    ADD DX,30H
    INT 21H
                                                    PUSH DX
                                                    INC BX
    MOV AH,03H
                                                    CMP AX,0000
    INT 10H
                                                    JA DCE
    INC DH
                                                    MOV CX,BX
    MOV DL,00
                                                    MOV AH,02H
                                              SHOW: POP DX
    MOV AH,02H
    INT 10H
                                                    INT 21H
                                                    LOOP SHOW
    MOV AX, COUNT1
                                                    RET
    CALL DECIMAL
                                                    DECIMAL ENDP
                                             END
    MOV AH,4CH
    INT 21H
                                          9. WAP in 8086 to find multiplication table
                                             of two digit no.
    VIDEO_MODE PROC
                                             Sol<sup>n</sup>:
    MOV AX,0000H
   INT 10H
                                             .MODEL SMALL
    RET
                                             .STACK 64
    VIDEO_MODE ENDP
                                             .DATA
```

NUM DB 2 DUP(?)

TEMP DW 2	MOVELOOF
TEMP DW ? TEMP2 DW ?	MOV DL,00H
	LABEL1:CALL SET_CURSOR
TEMP4 DW 2	MOV AX,TEMP
TEMP4 DW ?	MUL BL
.CODE	CALL DECIMAL
MOV AX,@DATA	INC BL
MOV DS,AX	LOOP LABEL1
MOV CX,0002	
LEA DI,NUM	
KJ:MOV AH,01H	MOV AH,4CH
INT 21H	INT 21H
MOV [DI],AL	
INC DI	VIDEO_MODE PROC
LOOP KJ	MOV AH,00H
MOV AH,02H	MOV AL,00H
MOV DL,20H	INT 10H
INT 21H	RET
MOV SI,OFFSET NUM	VIDEO_MODE ENDP
MOV DH,[SI]	
INC SI	CLEAR_SCREEN PROC
MOV DL,[SI]	MOV AH,06H
SUB DH,30H	MOV AL,00H
SUB DL,30H	MOV BH,07H
MOV CL,04H	MOV CX,0000H
ROL DH,CL	MOV DX,1827H
OR DH,DL	INT 10H
MOV CL,DH	RET
MOV CH,00H	CLEAR_SCREEN ENDP
LABELS:CMP CL,10H	
JB NEXT	SET_CURSOR PROC
INC CH	MOV AH,02H
SUB CL,10H	MOV BH,00
JMP LABELS	INC DH
NEXT:ADD CL,0AH	MOV DL,00
DEC CH	INT 10H
JNZ NEXT	RET
MOV CH,00	SET_CURSOR ENDP
MOV TEMP,CX	
	DECIMAL PROC
CALL VIDEO_MODE	MOV TEMP4,BX
CALL CLEAR_SCREEN	MOV BX,0000H
_	MOV TEMP3,CX
MOV CX,000AH	MOV CX,000AH
MOV BL,01H	MOV TEMP2,DX
MOV DH,00H	DCE: MOV DX,0000H
•	,

DIVICY	
DIV CX	BACK:MOV AH,[SI]
ADD DX,30H	CMP AH,'a'
PUSH DX	JE NEXT
INC BX	CMP AH,'e'
CMP AX,0000	JE NEXT
JA DCE	CMP AH,'i'
MOV CX,BX	JE NEXT
MOV AH,02H	CMP AH,'o'
SHOW: POP DX	JE NEXT
INT 21H	CMP AH,'u'
LOOP SHOW	JE NEXT
MOV DX,TEMP2	CMP AH,30H
MOV CX,TEMP3	JB L1
MOV BX,TEMP4	CMP AH,39H
RET	JA L1
DECIMAL ENDP	INC NUM
END	JMP GO
	L1:CMP AH,61H
	JB L2
10. WAP in 8086 to read string and count	CMP AH,7AH
no of vowels, consonents, numericals and	JA L2
other character and display them.	INC CON
Sol ⁿ :	JMP GO
	L2: INC OTH
.MODEL SMALL	JMP GO
.STACK 64	NEXT:INC VOW
.DATA	GO:INC SI
STRING DB 15,15 DUP (?)	LOOP BACK
VOW DW ?	MOV AH,02
CON DW ?	MOV DL,0AH
NUM DW ?	INT 21H
OTH DW ?	MOV AH,09H
MSG1 DB 'VOWEL=\$'	LEA DX,MSG1
MSG2 DB 'CONSONENT=\$'	INT 21H
MSG3 DB 'NUMERICAL=\$'	MOV AX,VOW
MSG4 DB 'OTHER=\$'	CALL DISPLAY
.CODE	MOV AH,02
MOV AX,@DATA	MOV DL,0AH
MOV DS,AX	INT 21H
LEA DX,STRING	MOV AH,09H
MOV AH,0AH	LEA DX,MSG2
MOV CX,14	INT 21H
INT 21H	MOV AX,CON
LEA SI,STRING	CALL DISPLAY
ADD SI,02	MOV AH,02
MOV CX,14	

MOV DL,0AH	.DATA
INT 21H	SUM DW 0000
MOV AH,09H	LO DW 01
LEA DX,MSG3	HI DW 05
INT 21H	TEMP DW ?
MOV AX,NUM	.CODE
CALL DISPLAY	MOV AX,@DATA
MOV AH,02	MOV DS,AX
MOV DL,0AH	MOV AH,01
INT 21H	INT 21H
	MOV AH,00
MOV AH,09H	SUB AL,30H
LEA DX,MSG4	MOV CY OOOAH
INT 21H	MOV CX,000AH BACK:MOV DX,0000H
MOV AX,OTH	MOV BX,0000H
CALL DISPLAY	ADD BX,LO
	ADD BX,TEMP
MOV AH,4CH	MOV AX,BX
INT 21H	MOV BX,HI
	MUL BX
DISPLAY PROC	ADD SUM,AX
MOV BX,0000	ADD LO,02
MOV CX,0AH	INC HI
DCE:MOV DX,0000H	LOOP BACK
DIV CX	MOV AH,02
ADD DX,30H	MOV DL,0AH
PUSH DX	INT 21H
INC BX	MOV AX,SUM
CMP AX,0000	MOV BX,0000
JA DCE	MOV CX,0AH
MOV CX,BX	DCE:MOV DX,0000H
MOV AH,02	DIV CX
SHOW:POP DX	ADD DX,30H
INT 21H	PUSH DX
	INC BX
LOOP SHOW	CMP AX,0000 JA DCE
RET	MOV CX,BX
DISPLAY ENDP	MOV AH,02
END	SHOW:POP DX
	INT 21H
	LOOP SHOW
11. WAP in 8086 of display	MOV AH,4CH
sum=(1+x)*5+(3+x)*6upto 10 terms	INT 21H
,where x is no from 0 to 9.	END
Sol ⁿ :	

.MODEL SMALL .STACK 64

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

Sol ⁿ :	DEC COUNT
	JNZ L1
.MODEL SMALL	-
.STACK 64	MOV AH,4CH
.DATA	INT 21H
DATA1 DB 125,235,197,91,48	END
COUNT DB 05	
.CODE	13. WAP in 8086 of display
MOV AX,@DATA	sum=1 ² +2 ² +upto 10 terms and display
MOV DS,AX	result.
MOV CX,05	
BACK2:LEA SI,DATA1	Sol ⁿ :
MOV DI,SI	
INC DI	MODEL SMALL
MOV BX,CX	.MODEL SMALL
BACK1:MOV AL,[SI]	.STACK 64
CMP AL,[DI]	.DATA
JA NEXT	SUM DW 0000 .CODE
MOV AL,[DI]	
MOV AH,[SI]	MOV DS AV
MOV [SI],AL	MOV CY COOAL
MOV [DI],AH	MOV CX,000AH MOV DH,01
NEXT: INC SI	
INC DI	BACK:MOV AL DH
DEC BX	MOV AL,DH MUL DH
JNZ BACK1	
LOOP BACK2	ADD SUM,AX INC DH
	LOOP BACK
LEA SI,DATA1	MOV AX,SUM
L1: MOV AL,[SI]	INIOV AA,SOIVI
MOV AH,00	MOV BX,0000
MOV BX,0000	MOV CX,0AH
MOV CX,0AH	DCE:MOV DX,0000H
DCE:MOV DX,0000H	DIV CX
DIV CX	ADD DX,30H
ADD DX,30H	PUSH DX
PUSH DX	INC BX
INC BX	CMP AX,0000
CMP AX,0000	JA DCE
JA DCE	MOV CX,BX
MOV CX,BX	MOV AH,02
MOV AH,02	SHOW:POP DX
SHOW:POP DX	INT 21H
INT 21H	LOOP SHOW
LOOP SHOW	2001 3110 **
MOV AH,02	MOV AH,4CH
MOV DL,20H	INT 21H
INT 21H	END
INC SI	LIND

```
14. WAP in 8086 to read string and count no
                                                            JE LL1
   of vowels and display no of vowel even
                                                            MOV AH,09
   or odd on screen.
                                                            LEA DX,MSG2
   Sol<sup>n</sup>:
                                                            INT 21H
                                                            JMP GO
   .MODEL SMALL
                                                       LL1:MOV AH,09
   .STACK 64
                                                            LEA DX,MSG1
   .DATA
                                                            INT 21H
          MSG DB 15,15 DUP(?),'$'
                                                       GO:MOV AH,4CH
          MSG1 DB 'EVEN VOWELSS'
                                                            INT 21H
          MSG2 DB 'ODD VOWELS$'
                                                     END
          COUNT1 DW 0000
   .CODE
                                                  15. WAP in 8086 of display
          MOV AX,@DATA
          MOV DS,AX
                                                     sum=x+2x+3x+...10 terms ,where x is no
                                                     from 0 to 99.
          MOV AH, 0AH
                                                     Sol<sup>n</sup>:
          MOV CX,12
          LEA DX,MSG
                                                     .MODEL SMALL
          INT 21H
                                                     .STACK 64
          MOV AH,02
                                                     .DATA
          MOV DL,0AH
                                                            NUM DB 2 DUP(?)
          INT 21H
                                                            SUM DW 0000
          MOV SI, OFFSET MSG
                                                     .CODE
          MOV CX,000DH
                                                            MOV AX,@DATA
          MOV BX,0000H
                                                            MOV DS,AX
          ADD SI,02H
      BACK:MOV AH,[SI]
                                                            MOV CX,0002
                                                            LEA DI, NUM
          CMP AH,61H
                                                       KJ:MOV AH,01H
          JE NEXT
                                                            INT 21H
          CMP AH,65H
                                                            MOV [DI],AL
          JE NEXT
                                                            INC DI
          CMP AH,69H
                                                            LOOP KJ
          JE NEXT
                                                            MOV AH,02H
          CMP AH,6FH
                                                            MOV DL,20H
          JE NEXT
                                                            INT 21H
          CMP AH,75H
                                                            MOV SI, OFFSET NUM
          JE NEXT
                                                            MOV DH,[SI]
          JMP GOTO1
      NEXT:INC BX
                                                            INC SI
                                                            MOV DL,[SI]
      GOTO1:INC SI
                                                            SUB DH,30H
          LOOP BACK
          MOV COUNT1,BX
                                                            SUB DL,30H
                                                            MOV CL,04H
          MOV AX,BX
                                                            ROL DH,CL
          AND AL,00000001
                                                            OR DH, DL
          CMP AX,00H
```

MOV CL, DH

MOV CH,00H	TEMP DW ?
LABELS:CMP CL,10H	TEMP2 DW ?
JB NEXT	TEMP3 DW ?
INC CH	TEMP4 DW ?
SUB CL,10H	COUNT DB 05
JMP LABELS	.CODE
NEXT:ADD CL,0AH	MOV AX,@DATA
	MOV DS,AX
DEC CH	CALL VIDEO_MODE
JNZ NEXT	CALL CLEAR_SCREEN
MOV CH,00	
MOV BH,CL	LEA SI,NUM
MOV CX,0AH	MOV DH,00H
B1:MOV AL,BH	MOV DL,00H
MUL CL	KJ:MOV CX,000AH
ADD SUM,AX	MOV BL,01H
LOOP B1	LABEL1:MOV AL,[SI] ;CALL SET_CURSOR
	MOV AH,00
MOV AX,SUM	MUL BL
MOV BX,0000	CALL DECIMAL
MOV CX,0AH	INC BL LOOP LABEL1
DCE:MOV DX,0000H	
DIV CX	MOV AH,03H INT 10H
	ADD DH,02
ADD DX,30H	MOV AH,02H
PUSH DX	MOV DL,00
INC BX	MOV BH,00
CMP AX,0000	INT 10H
JA DCE	INC SI
MOV CX,BX	DEC COUNT
MOV AH,02	JNZ KJ
SHOW:POP DX	
INT 21H	MOV AH,4CH
LOOP SHOW	INT 21H
MOV AH,4CH	
INT 21H	VIDEO_MODE PROC
END	MOV AH,00H
	MOV AL,00H
	INT 10H
16. WAP in 8086 to find multiplication table	RET
of 5 nos stored in array.	VIDEO_MODE ENDP
Sol ⁿ :	
JUI.	CLEAR_SCREEN PROC
	MOV AH,06H
MODEL CMALL	MOV AL,00H
.MODEL SMALL	MOV BH,07H
.STACK 64	MOV CX,0000H
.DATA	MOV DX,1827H
NUM DB 2,3,4,5,6	INT 10H

RET	COUNT1 DW 0000
CLEAR_SCREEN ENDP	.CODE
_	MOV AX,@DATA
;SET_CURSOR PROC	MOV DS,AX
;MOV AH,02H	MOV AH,0AH
;MOV BH,00	MOV CX,12H
;INC DL	LEA DX,MSG
;MOV DH,00	INT 21H
;INT 10H	
;RET	CALL VIDEO_MODE
;SET_CURSOR ENDP	CALL CLEAR_SCREEN
DECIMAL PROC	
MOV TEMP4,BX	MOV SI,OFFSET MSG
MOV BX,0000H	MOV CX,000DH
MOV TEMP3,CX	MOV BX,0000H
MOV CX,000AH	ADD SI,02H
MOV TEMP2,DX	BACK:MOV AL,[SI]
DCE: MOV DX,0000H	CMP AL,61H
DIV CX	JE NEXT
ADD DX,30H	CMP AL,65H
PUSH DX	JE NEXT
INC BX	CMP AL,69H
CMP AX,0000	JE NEXT
JA DCE	CMP AL,6FH
MOV CX,BX	JE NEXT
MOV AH,02H	CMP AL,75H
SHOW: POP DX	JE NEXT
INT 21H	MOV AH,02
LOOP SHOW	MOV DL,AL
MOV DL,''	INT 21H
INT 21H	JMP GOTO1
MOV DX,TEMP2	NEXT:INC BX
MOV CX,TEMP3	GOTO1:INC SI
MOV BX,TEMP4	LOOP BACK
RET	MOV COUNT1,BX
DECIMAL ENDP	·
END	MOV AH,02
	MOV DL,0AH
17. WAP in 8086 to read string and count no	INT 21H
of vowels and display string without	
vowel in clear screen & count in clear	MOV AX,COUNT1
screen.	CALL DECIMAL
Sol ⁿ :	
301.	MOV AH,4CH
.MODEL SMALL	INT 21H
STACK 64	
.DATA	VIDEO_MODE PROC
MSG DB 15,15 DUP(?),'\$'	MOV AX,0000H
אסס כדיכד מס ספואו (; ל' ל')	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