ASSEMBLING AND EXECUTING THE PROGRAM

Writing an ALP

Assembly level programs generally abbreviated as ALP are written in text editor EDIT.

Type EDIT in front of the command prompt to open an untitled text file.

EDIT<file name>

After typing the program save the file with appropriate file name with an extension .ASM

Ex:

Add.ASM

Assembling an ALP

To assumble an ALP we needed executable file calledMASM.EXE. Only if this file is in

current working directory we can assemble the program. The command is

MASM<filename.ASM>

If the program is free from all syntactical errors, this command will give the OBJECT file.In

case of errors it list out the number of errors, warnings and kind of error.

Note:No object file is created until all errors are rectified.

Linking

After successful assembling of the program we have to link it to get Executable file.

The command is

LINK<File name.OBJ>

This command results in <Filename.exe> which can be executed in front of the command

prompt.

Executing the Program

Open the program in debugger by the command(note only exe files can be open)by the

command.

CV <Filename.exe>

This will open the program in debugger screen where in you can view the assemble code

with the CS and IP values at the left most side and the machine code. Register content

,memory content also be viewed using VIEW option of the debugger.

Execute option in the menu in the menu can be used to execute the program either in

single steps(F7) or burst execution(F5).

- 4 - 1. Program involving Data transfer instructions

i)Byte and word data transfer in different addressing modes

```
DATA SEGMENT
DATA1 DB 23H
DATA2 DW 1234H
DATA3 DB 0H
DATA4 DW 0H
DATA5 DW 2345H,6789H
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:  MOV AX,DATA             ;Initialize  DS to point to start of the memory
    MOV DS,AX                   ;set aside for storing of data
    MOV AL,25X                  ;copy 25H into  8 bit AL register
    MOV AX,2345H                ;copy 2345H into  16 bit AX register
    MOV BX,AX                   ;copy the content of AX into BX register(16 bit)
     MOV  CL,AL                 ;copy the content of AL into CL register
     MOV  AL,DATA1              ;copies the byte contents of data segment memory
                                        ;location   DATA1 into 8 bit AL
   MOV  AX,DATA2                ;copies the word contents of data segment memory
                                        ;location  DATA2 into 16 bit AX
            MOV  DATA3,AL               ;copies the AL content  into the byte contents of
data
```

```
                                        ;segment  memory location DATA3
        MOV  DATA4,AX               ;copies the AX content  into the word contents of
                                        ;data   segment  memory location DATA4
        MOV BX,OFFSET DATA5 ;The 16 bit offset address of DS memeory location
                                         ; DATA5 is copied into BX
        MOV AX,[BX]         ; copies the word content  of  data  segment
;memory location addressed by BX  into
                                       ;AX(register indirect addressing)
        MOV  DI,02H                  ;address element
        MOV AX,[BX+DI}              ; copies the word content  of  data  segment
;memory location addressed by BX+DI  into
;AX(base plus indirect addressing)
        MOV AX,[BX+0002H]        ; copies the word content  of  data  segment
;(16 bit)
  MOV  AL,[DI+2]                ;register relative addressing
        MOV AX,[BX+DI+0002H]  ;copies the word content  of  data  segment
                                        ;memory location addressed by
BX+DI+0002H
                                        ;into   AX(16 bit)
        MOV  AH,4CH                 ; Exit to DOS with function call 4CH
        INT 21H
CODE ENDS                           ; Assembler stop reading
END START
```

- 5 - ii)Block move (with and with out overlapping)

Without overlapping

```
DATA SEGMENT
    X DB 01H,02H,03H,04H,05H    ;Initialize Data Segments Memory Locations
    Y DB 05 DUP(0)
DATA ENDS
    CODE SEGMENT
       ASSUME CS:CODE,DS:DATA
    START:MOV AX,DATA         ; Initialize  DS to point to start of the memory
              MOV DS,AX              ; set aside for storing of data
              MOV CX,05H             ;  Load  counter
              LEA SI,X+04              ;  SI pointer pointed  to  top of the memory block
     LEA DI,X+04+03        ;  03 is displacement of over lapping, DI  pointed   to
                                    ;the   top of the destination block
```

Before execution

00

00

00

00

00

05

04

03

02

01

After execution

05

04

03

02

01

05

04

03

02

01

Y,DI

X, SI

- 6 - With Overlapping

```
DATA SEGMENT
    X DB 01H,02H,03H,04H,05H  ;  Initialize Data Segments Memory Locations
 DATA ENDS
  CODE SEGMENT
       ASSUME CS:CODE,DS:DATA
   START:MOV AX,DATA         ; Initialize  DS to point to start of the memory
            MOV DS,AX              ; set aside for storing of data
            MOV CX,05H             ; Load  counter
            LEA SI,X+04              ;  SI pointer pointed  to  top of the memory block
     LEA DI,X+04+03         ;  03 is displacement of over lapping, DI  pointed   to
                                ;the   top of the destination block
      UP:    MOV BL,[SI]              ;  Move the SI content to BL register
            MOV [DI],BL              ; Move the BL register  to content of DI
   DEC SI                      ; Update SI and DI
            DEC DI
            DEC CX                     ; Decrement the counter till it becomes zero
            JNZ UP
            MOV AH,4CH
            INT 21H
 CODE ENDS
 END START
```

DS Before execution

xx

xx

xx

xx

xx

05

04

03

02

01

DI

SI

X

DS After execution

xx

xx

05

04

03

- 7 - 02

01

03

02

01

- 8 - **iii) Block Interchange**

DATA SEGMENT

```
X DB 01H,02H,03H,04H,05H
Y DB 11H,12H,13H,14H,15H
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:MOV AX,DATA
        MOV DS,AX
 MOV CX,05H                ; Load the counter
        LEA SI,X                   ; SI pointed to the source location x
        LEA DI,Y                   ; DI pointed to the destination location y
   UP:    MOV BL,[SI]                 ; Move the SI content to BL register
        MOV AL,[DI]               ; Move the DI content to AL register
        MOV [SI],AL                ; Move AL register content to content of SI
        MOV [DI],BL                ; Move BL register content to content of DI
      INC SI                       ; Update SI and DI
      INC DI
      DEC CX                       ; Decrement the counter till it becomes zero
      JNZ UP
      MOV AH,4CH
      INT 21H
CODE ENDS
END START
DS Before execution
15
14
13
```

12

11

05

04

03

02

01

Y,DI

X, SI

- 9 - DS After execution

05

04

03

02

01

15

14

13

12

11

- 10 - 2) Program involving Arithmetic and logic operations like addition and subtraction of

multi precision numbers

i) 16 Bit Addition

DATA SEGMENT

```
   NUM DW 1234H, 0F234H
   SUM  DW 2 DUP(0)
DATA ENDS
CODE SEGMENT
     ASSUME CS: CODE, DS:DATA
START: MOV AX,DATA
     MOV DS,AX
        MOV AX,NUM                    ; First number loaded into AX
        MOV BX,0H                      ; For carry BX register is cleared
        ADD AX,NUM+2                  ; Second number added with AX
        JNC DOWN                       ; Check for carry
        INC BX                           ; If carry generated increment the BX
DOWN: MOV SUM,AX                      ; Storing the sum value
        MOV SUM+2,BX                  ; Storing the carry value
        MOV AH,4CH
        INT 21H
CODE ENDS
END START
INPUT    : 1234H, F234H
     OUTPUT :  10468H
```

- 11

- ii) 32 Bit addition

```
DATA SEGMENT
   NUM1 DW 0FFFFH,0FFFFH
   NUM2 DW 1111H,1111H
   SUM  DW 4 DUP(0)
```

```
dATA ENDS
CODE SEGMENT
    ASSUME CS:CODE,DS:DATA
START: MOV AX,DATA
       MOV DS,AX
       MOV AX,NUM1
      ADD AX,NUM2
      MOV SUM,AX
      MOV AX,NUM1+2
      ADC AX,NUM2+2
;Move LSB of  NUM1 to AX
;Add  LSB of NUM2   to AX
;Store the LSB   in SUM
; Move MSB of  NUM1 to AX
; Add  MSB of NUM2   to AX
   JNC DOWN                          ; Check for carry
      MOV SUM+4,01H
DOWN:  MOV SUM+2,AX
          MOV AH,4CH
          INT 21H
CODE ENDS
END START
INPUT: 0FFFFFFFFH, 011111111H
  OUTPUT: 0111111110H
; Store the carry in SUM+4
; Store the MSB in SUM+2
```

 iii) 32 Bit addition using DD directive

```
DATA SEGMENT
   NUM1 DW 12345678H
   NUM2 DW 12345678H
   SUM  DW 3 DUP(0)
DATA ENDS
CODE SEGMENT
```

```
    ASSUME CS:CODE,DS:DATA
START:MOV AX,DATA
      MOV DS,AX
      LEA SI,NUM1            ;  SI pointed to the of LSB of NUM1
      LEA DI,NUM2            ;  DI pointed to the of LSB of NUM2
      MOV AX,[SI]            ;  Move the content of SI to AX
  ADD AX,[DI]
     MOV CX,[SI+2]
        ; Add DI content to AX
        ; Move the SI to point MSB of NUM1 and move that
                                    ;content to CX
      ADC CX,[DI+2]              ; Move the DI to point MSB of NUM2 and  add
                                     ;with carry to CX
      JNC DOWN                   ; Check for carry
      MOV SUM+4,01H              ; Store the carry in SUM+4
DOWN:MOV SUM,AX             ; Store the LSB   in SUM
      MOV SUM+2,CX               ; Store the MSB   in SUM+2
      MOV AH,4CH
      INT 21H
CODE ENDS
END START
```

INOUT: 12345678H,12345678H

OUTPUT:2468ACF0H

- 13 **- iv) 16 Bit Subtraction**

```
DATA SEGMENT
   NUM DW  4567H,2345H
   DIF DW 1  DUP(0)
DATA ENDS
CODE SEGMENT
    ASSUME
CS:CODE,DS:DATA
START:   MOV AX,DATA
```

```
        MOV DS,AX
        CLC
; Clearing Carry
        LEA SI,NUM                  ; SI pointed to the NUM
        MOV AX,[SI]                 ; Move NUM1 to AX
        SBB AX,[SI+2]       ; Move the SI to Num2 and subtract with AX(Takes
                                    ;care for both smaller as well  as larger
                                    ;Number subtraction)
        MOV DIF,AX                  ;Store the result
        MOV AH,4CH
        INT 21H
CODE ENDS
END START
INPUT: 4567H,2345H
OUTPUT:2222
```

- 14 **- v) 32 Bit Subtraction**

```
DATA SEGMENT
   NUM1 DW 2345H,6762H
   NUM2 DW 1111H,1111H
   DIF  DW 2 DUP(0)
DATA ENDS
CODE SEGMENT
     ASSUME CS:CODE,DS:DATA
START:    MOV AX,DATA
        MOV DS,AX
```

```
        LEA SI,NUM1
        LEA DI,NUM2
        MOV AX,[SI]
        MOV BX,[DI]
        SUB AX,BX
        MOV DIF,AX
INC SI
; SI pointed to the LSB of NUM1
; DI pointed to the LSB of NUM2
; Move the content of SI to AX
; Move the content of DI to BX
; Subtract from BX to AX
; Store the LSB result in DIF
      ;Update SI to point the MSB of NUM1(if
                                          ;ADD SI,02 instruction its affect carry flag)
        INC SI
        INC DI
        INC DI
        MOV AX,[SI]
        MOV BX,[DI]
            ;Update DI to point the MSB of NUM2
; Move the content of SI to AX
; Move the content of DI to BX
        SBB AX,BX                    ;  Subtract with borrow from BX to AX
        MOV DIF+2,AX
        MOV AH,4CH
```

```
        INT 21H
CODE ENDS
END START
```

INPUT: 23456762,-11111111

OUTPUT:12345651

INPUT:11111111,-23451234

OUTPUT:EDCBFEDD

```
;  Store the MSB result in DIF+2
```

- 15 - Multiplication and Division of signed and unsigned Hexadecimal numbers

vi)16 Bit multiplication for unsigned numbers

```
DATA SEGMENT
   NUM DW  1234H,1234H
   PROD DW 2  DUP(0)
DATA ENDS
CODE SEGMENT
     ASSUME CS:CODE,DS:DATA
START: MOV AX,DATA
           MOV DS,AX
           LEA SI,NUM          ; SI pointed to the Multiplicand
           MOV AX,[SI]  ; Multiplicand has to be in AX register
```

```
   MOV BX,[SI+2]      ; SI+2  pointed to the Multiplier and move it to BX
   MUL BX  ;Perform the multiplication
   MOV PROD,AX  ;32 bit product stored in DX-AX registers
   MOV PROD+2,DX
            MOV AH,4CH
            INT 21H
CODE ENDS
END START
INPUT: Multiplicand- 1234H,
  Multiplier    -  1234H
OUTPUT: DX-01 4B
      AX-54 90
```

- 16 **- vii)16 Bit multiplication for signed numbers**

```
DATA SEGMENT
   NUM DW  -2,1
   PROD DW 2  DUP(0)
DATA ENDS
CODE SEGMENT
     ASSUME CS:CODE,DS:DATA
   START:  MOV AX,DATA
             MOV DS,AX
       LEA SI,NUM        ; SI pointed to the Multiplicand
       MOV AX,[SI]     ; Multiplicand has to be in AX register
       MOV BX,[SI+2]    ; SI+2  pointed to the Multiplier and move it to BX
       IMUL BX              ; Perform the sign multiplication using sign
```

```
                                ;Multiplication operator (IMUL)
        MOV PROD,AX    ; 32 bit product stored in DX-AX registers
        MOV PROD+2,DX
        MOV AH,4CH
                INT 21H
CODE ENDS
END START
INPUT: Multiplicand- -2,
  Multiplier    -  1
OUTPUT: DX – FF FF
                 AX – FF FE       ; Result is in two complement form.
```

- 17 - **viii)8 Bit Division for Unsigned numbers**

```
DATA SEGMENT
   NUM1 DB  72H,
   NUM2 DB  02H
   QUO DB 1  DUP(0)
   REM DB 1  DUP(0)
DATA ENDS
CODE SEGMENT
     ASSUME CS:CODE,DS:DATA
START:   MOV AX,DATA
        MOV DS,AX
        MOV AL,NUM1     ;Move the Dividend  to AL
        MOV AH,0H
        DIV NUM2
```

```
 ; Zero extended for 16 bit/8 bit division
 ; Perform the Division operation
      MOV QUO,AL  ; Store the quotient to AL
      MOV REM,AH       ;Store the reminder to AH
      MOV AH,4CH
      INT 21H
CODE ENDS
END START
```

INPUT: Dividend - 72H,

Divisor - 02 H,

0UTPUT: AL - 39H (quotient);

AX - 00H (reminder);

INPUT: Dividend - 55H,

Divisor - 04 H,

0UTPUT: AL - 15H (quotient);

AX - 01H (reminder);

- 18 - **ix)8 Bit Division for Signed numbers**

```
DATA SEGMENT
   NUM1 DB  -10
   NUM2 DB  02
   QUO DB 1  DUP(0)
   REM DB 1  DUP(0)
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE,DS:DATA
START:   MOV AX,DATA
       MOV DS,AX
       MOV AL,NUM1     ;Move the Dividend  to AL
       CBW
       IDIV NUM2           ; Perform the Sign Division operation using IDIV operator
       MOV QUO,AL  ; Store the quotient to AL
       MOV REM,AH       ;Store the reminder to AH
```

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

INPUT: Dividend - -10

Divisor - 02

0UTPUT: AL – FBH (quotient) ; Result is in two complement form

INPUT: Dividend - -10

Divisor - 03

0UTPUT: AL – FDH (quotient);

AX – FF H (reminder) ; Result is in two complement form

- 19 **- x)16 Bit Division for Unsigned numbers**

```
DATA SEGMENT
   NUM1 DW 4567H,2345H
   NUM2 DW  4111H
   QUO DW 2  DUP(0)
   REM DW 1  DUP(0)
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE,DS:DATA
START:   MOV AX,DATA
       MOV DS,AX
       MOV AX,NUM1                ;Move the lower bit of Dividend  to AX
       MOV DX,NUM1+2
       DIV NUM2
       MOV QUO,AX
       MOV REM,DX
       MOV AH,4CH
       INT 21H
```

```
CODE ENDS

END START

INPUT: Dividend - 23454567,

  Divisor    - 4111,

0UTPUT: AX – 8AC5H (quotient);

                DX – 0952H (reminder);

  ; Move the higher bit of Dividend  to DX

  ; Perform the Division operation

  ; Store the quotient to AX

  ; Store the reminder to DX
```

- 20 **- xi)16 Bit Division for Signed numbers**

```
DATA SEGMENT
   NUM1 DW 4567H,2345H
   NUM2 DW  4111H
   QUO DW 2  DUP(0)
   REM DW 1  DUP(0)
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE,DS:DATA
START:   MOV AX,DATA
       MOV DS,AX
       MOV AX,NUM1              ; Move the lower bit of Dividend  to AX
       MOV DX,NUM1+2
       CWD
       IDIV NUM2
       MOV QUO,AX
       MOV REM,DX
       MOV AH,4CH
       INT 21H
CODE ENDS
```

```
END START
```

INPUT: Dividend - -44444444,

Divisor - 2222,

0UTPUT: AX – FE (quotient);

DX – FF (reminder)

```
  ; Move the higher bit of Dividend  to DX
   ; Perform the sign Division operation using IDIV
   ; Store the quotient to AX
   ; Store the reminder to DX
    ; Result is in two complement form.
```

- 21 - 3.Code Conversion

i)ASCII adjustment instructions

```
CODE SEGMENT
ASSUME CS:CODE
START:   MOV AX,31H       ;Load ASCII 1
                        ADD AX,39H     ;Load ASCII 9
                        AAA              ;ASCII Adjust, AX=0100 UNPACKED BCD
     ADD AX,3030H   ;Answer in ASCII
     MOV BL,9      ;Load divisor
     MOV AX,0702H   ;Load dividend, AAD instruction requires
```

Ax register to contain a two digit unpacked

```
;BCD number before executing
      AAD                  ;AAD appears before division
      DIV BL               ;Contents of adjusted AX register is devised
```

;by an unpacked BCD number to generate

;a single digit result in AL with any

```
;remainder in AH
      MOV AL,5          ;Load multiplicand
      MOV CL,5           ;Load multiplier
      MUL CL             ;AX=0019H
      AAM                ;AX=0205(Unpacked BCD)
                        ADD AX,3030H   ;AX=3235H
     MOV AX,38H     ;Load ASCII 8
                       SUB AX,31H        ;Load ASCII 1
     AAS                ;AX=0007
     AX,3030H           ;AX=3037H
     MOV AH,4CH
     INT 21H
CODE ENDS
END START
```

 ii)Binary to BCD code conversion

```
DATA SEGMENT
        BIN DW 01A9H
        BCD DB 2 DUP(0)
DATA ENDS
CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
 START: MOV AX,DATA
                MOV DS,AX
                MOV AX,BIN
                MOV CL,64H
                DIV CL
                MOV BCD+1,AL
                MOV AL,AH
                MOV AH,00H
                MOV CL,0AH
                DIV CL
                MOV CL,04
                ROR AL,CL
                ADD AL,AH
                MOV AH,4CH
                INT 21H
CODE ENDS
```

END START

Input: binary--------01A9

Output: bcd----------425

;Load the Data to AX.

;Move the Data AX to DS.

;Move the Binary Data to AX.

;100 in decimal

;Perform the division by 100.

;Store the quotient in BCD+1.

;Move the Reminder value to AL.

;Initialize the AH.

;10 in decimal.

;Perform the division by 10.

;Perform the Right side rotation 4 times.

;Adding the Reminder in LSB.

- 23 **- iii)BCD to Binary code conversion**

```
DATA SEGMENT
   BCD DW 27H
   BIN DW ?
DATA ENDS
CODE SEGMENT
  ASSUME CS:CODE,DS:DATA
  START:  MOV AX,DATA
          MOV DS,AX
          MOV AX,BCD
          AND AX,07H
          MOV BX,AX
          MOV AX,BCD
          AND AX,0F0H
          MOV CX,0AH
          MUL CX
          ADD AX,BX
          MOV BIN,AX
          MOV AH,4CH
          INT 21H
CODE ENDS
END START
```

Input: BCD----------27

Output:----------1B

;Load the Data to AX.

```
;Move the Data AX to DS.
;Move the BCD Data to AX.
;Perform the AND operation between
;07H and input BCD
;Move data AX to BX
;Move the BCD Data to AX.
;Perform the AND with 0F0H for shifting operation.
;10 in decimal.
;Perform the multiplication by 10.
;Perform the addition operation to get the LSB.
;Move the result to binary.
```

i) Program to find square and cube of a number

```
DATA SEGMENT
   X DW 04H
   SQUARE DW ?
   CUBE DW ?
DATA ENDS
CODE SEGMENT
   ASSUME CS:CODE,DS:DATA
   START: MOV AX,DATA
                MOV DS,AX
                MOV AX,X
                MOV BX,X
                MUL BX
                MOV SQUARE,AX
                MUL BX
                MOV CUBE,AX
                MOV AH,4CH
                INT 21H
CODE ENDS
END START
```

Input: x ---------- 4h

Output: Square ---------10h

Cube ----------40h

;Load the Data to AX.

;Move the Data AX to DS.

;Move the X number Data to AX.

;Move the X number Data to BX.

;Perform the multiplication by BX.

;Store value in SQUARE.

;Perform the multiplication by BX.

;Store value in CUBE.

- 25 **- ii)Program to find LCM of a given number**

```
DATA SEGMENT
    NUM DW 05,04
```

```
    LCM DW 2 DUP(0)
DATA ENDS
CODE SEGMENT
   ASSUME CS:CODE,DS:DATA
   START:  MOV AX,DATA
           MOV DS,AX
           MOV DX,0H
           MOV AX,NUM
           MOV BX,NUM+2
    UP: PUSH AX
           PUSH DX
           DIV BX
           CMP DX,0
           JE EXIT
           POP DX
           POP AX
           ADD AX,NUM
           JNC DOWN
           INC DX
     DOWN: JMP UP
     EXIT: POP LCM+2
           POP LCM
           MOV AH,4CH
           INT 21H
CODE ENDS
END START
```

Input: 0A, 04

Output: 02

;Load the Data to AX.

;Move the Data AX to DS.

;Initialize the DX.

;Move the first number to AX.

;Move the second number to BX.

;Store the quotient/first number in AX.

;Store the remainder value in DX.

;Divide the first number by second number.

;Compare the remainder.

;If remainder is zero, go to EXIT label.

;If remainder is non-zero,

;Retrieve the remainder.

;Retrieve the quotient.

;Add first number with AX.

;If no carry jump to DOWN label.

;Increment DX.

;Jump to Up label.

;If remainder is zero, store the value at LCM+2.

- 26 **- iii)Program to find GCD of two numbers**

```
DATA SEGMENT
    NUM1 DW 000AH
    NUM2 DW 0004H
    GCD DW ?
```

```
DATA ENDS
CODE SEGMENT
   ASSUME CS:CODE,DS:DATA
   START: MOV AX,DATA
          MOV DS,AX
          MOV AX,NUM1
          MOV BX,NUM2
   UP: CMP AX,BX
          JE EXIT
          JB EXCG
  UP1:MOV DX,0H
          DIV BX
          CMP DX,0
          JE EXIT
          MOV AX,DX
          JMP UP
   EXCG:XCHG AX,BX
          JMP UP1
   EXIT:MOV GCD,BX
          MOV AH,4CH
          INT 21H
CODE ENDS
END START
```

Input: 0A,04

Output: 02

;Load the Data to AX.

```
;Move the Data AX to DS.
;Move the first number to AX.
;Move the second number to BX.
;Compare the two numbers.
;If equal, go to EXIT label.
;If first number is below than second,
;go to EXCG label.
;Initialize the DX.
;Divide the first number by second number.
;Compare remainder is zero or not.
;If zero, jump to EXIT label.
;If non-zero, move remainder to AX.
;Jump to UP label.
;Exchange the remainder and quotient.
;Jump to UP1.
;Store the result in GCD.
```

 iv)Program to find factorial of a given number

```
DATA SEGMENT
        X DW 06H
        FACT DW ?
DATA ENDS
CODE SEGMENT
   ASSUME CS:CODE,DS:DATA
```

```
   START: MOV AX,DATA
          MOV DS,AX
          MOV AX,01H
          MOV CX,X
   UP: MUL CX
          LOOP UP
          MOV FACT,AX
          MOV AH,4CH
          INT 21H
CODE ENDS
END START
```

Input: 06

Output: 2D0H

;Set the value of AX as 01H.

;Move the i/p number to CX.

;Perform the Loop multiplication operation.

;Store the FACT value.

- 28 - 5.Program involving bit manipulation instruction

i)If given data is positive or negative

```
DATA SEGMENT
   NUM DB 12H
   MES1 DB 10,13,'DATA IS POSITIVE $'
   MES2 DB 10,13,'DATA IS NEGATIVE $'
DATA ENDS
CODE SEGMENT
```

```
   ASSUME CS:CODE,DS:DATA
   START: MOV AX,DATA
          MOV DS,AX
          MOV AL,NUM
          ROL AL,1
          JC NEGA
;Move the Number to AL.
;Perform the rotate left side for 1 bit position.
;Check for the negative number.
          MOV DX,OFFSET MES1  ;Declare it positive.
          JMP EXIT  ;Exit program.
   NEGA:  MOV DX,OFFSET MES2;Declare it negative.
   EXIT:  MOV AH,09H
          INT 21H
          MOV AH,4CH
          INT 21H
CODE ENDS
END START
```

Output: Data is positive

Positive Numbers: 00-7F

Negative numbers: 80-FF

- 29 **- ii)If given data is odd or even**

```
DATA SEGMENT
    X DW 27H
    MSG1 DB 19,13,'NUMBER IS EVEN$'
    MSG2 DB 10,13,'NUMBER IS ODD$'
DATA ENDS
CODE SEGMENT
  ASSUME CS:CODE,DS:DATA
  START: MOV AX,DATA
        MOV DS,AX
```

```
        MOV AX,X
        TEST AX,01H
        JNZ EXIT
        LEA DX,MSG1
        MOV AH,09H
        INT 21H
        JMP LAST
;Test for Even/Odd number.
;If it is Even go to Exit label.
;(alternate logic)
;MOV BL,2
;DIV BL
;CMP AH,0H
;JNZ EXIT
;Declare it is Even number.
  EXIT:   LEA DX,MSG2  ;Declare it is Odd number.
        MOV AH,09H
        INT 21H
  LAST:   MOV AH,4CH
        INT 21H
CODE ENDS
END START
```

Output: Number is ODD

- 30 **- iii) Logical ones and zeros in a given data**

```
DATA SEGMENT
        X DB 0AAH
        ONE DB ?
        ZERO DB ?
DATA ENDS
CODE SEGMENT
   ASSUME CS: CODE,DS:DATA
   START:  MOV AX,DATA
           MOV DS,AX
           MOV AH,X
           MOV BL,8
```

```
        MOV CL,1
  UP:  ROR AH,CL
        JNC DOWN
        INC ONE
        JMP DOWN1
  DOWN:   INC ZERO
  DOWN1:  DEC BL
        JNZ UP
        MOV AH,4CH
        INT 21H
CODE ENDS
END START
```

Output: Ones--------04

Zeros--------04

;Initialize BL to 8.

;Initialize CL to 1.

;Perform the single bit rotate operation

;with respect to right.

;If no carry go to DOWN label.

;Increment one.

;Jump to DOWN1.

;Increment ZERO.

;Decrement the BL.

;If no zero go to UP label.

 iv) 2 out of 5 code

```
DATA SEGMENT
        X DW 82H
        MES DB 10,13,'VALID 2 OUT OF CODE $'
        MES1 DB 10,13,'NOT A VALID 2 OUT OF CODE $'
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE,DS:DATA
    START: MOV AX,DATA
           MOV DS,AX
           MOV AX,X
           MOV BX,0H
           AND AX,0E0H
           JNZ DISP
```

```
        MOV CL,05
        MOV AX,X
  UP:  ROR AX,1
        JNC DOWN
        INC BX
DOWN:DEC C
        JNC UP
        CMP BX,02H
        JNZ DISP
        LEA DX,MES
        MOV AH,09H
        INT 21H
        JMP EXIT
DISP: LEA DX,MES1
        MOV AH,09H
        INT 21H
 EXIT:MOV AH,4CH
        INT 21H
CODE ENDS
END START
```

Output: Not a valid 2 out of 5 code.

;Load the Data to AX.

;Move the Data AX to DS.

;Move the Data word to AX.

;Initialize the BX.

;Perform the AND operation of first 3 bit.

```
;If no zero jump to DISP label.
;If zero, Initialize the counter for check the last 5 bit.
;Move the Data word to AX.
;Rotate right side one time.
;If no carry jump to DOWN label.
;Increment the BX.
;Decrement the counter.
;If no carry jump to UP label.
;Compare the BX with 2.
;If no zero jump to DISP label.
;Declared as 2 out of 5 code .
;Declared as not valid code .
```

- 32 **- v) Bit wise palindrome**

```
DATA SEGMENT
        X DW 0FFFFH
        MSG1 DB 10,13,'NUMBER IS PALINDROME$'
        MSG2 DB 10,13,'NUMBER IS NOT PALINDROME$'
DATA ENDS
CODE SEGMENT
   ASSUME CS:CODE,DS:DATA
   START: MOV AX,DATA
          MOV DS,AX
          MOV AX,X
          MOV CL,10H
   UP: ROR AX,1
          RCL DX,1
          LOOP UP
          CMP AX,DX
          JNZ DOWN
```

```
        LEA DX,MSG1
        MOV AH,09H
        INT 21H
        JMP EXIT
DOWN: LEA DX,MSG2
        MOV AH,09H
        INT 21H
EXIT:MOV AH,4CH
        INT 21H
CODE ENDS
END START
```

Output: Number is Palindrome

```
;Load the Data to AX.
;Move the Data AX to DS.
;Move DW to AX.
;Initialize the counter 10.
;Rotate right one time.
;Rotate left with carry one time.
;Loop the process.
;Compare AX and DX.
;If no zero go to DOWN label.
;Declare as a PALINDROME.
;Jump to EXIT label.
; Declare as not a PALINDROME
```

- 33 - vi) Nibble wise palindrome

```
DATA SEGMENT
        X DW 2662H
        TEMP DW 0H
        MES DB 10,13,'THE WORD IS NIBBLEWISE PALINDROME $'
        MES1 DB 10,13,'THE WORD IS NOT NIBBLEWISE PALINDROME $'
DATA ENDS
CODE SEGMENT
   ASSUME CS:CODE,DS:DATA
   START:  MOV AX,DATA
           MOV DS,AX
           MOV AX,X
           MOV BX,X
           MOV CL,04
           AND AX,0000FH
           AND BX,0F000H
           ROL BX,CL
           CMP AX,BX
           JNZ TER
```

```
        MOV AX,X
        ROR AX,CL
        MOV BX,AX
        AND AX,000FH
        ROR BX,CL
        AND BX,000FH
        CMP AX,BX
        JNZ TER
        MOV AH,09H
        LEA DX,MES
        INT 21H
        JMP LAST
  TER:MOV AH,09H
        LEA DX,MES1
        INT 21H
LAST:MOV AH,4CH
     INT 21H
CODE ENDS
END START
;Initialize counter.
;Perform the and  operation between
;last nibble of AX and 000FH.
;Perform the and  operation between
;last nibble of BX and 000FH.
;Rotate left side 4 times the BX.
;Compare AX with BX.
```

;If no zero go to TER label.

;Move the DW to AX.

;Rotate right side 4 times the AX.

;Move AX to BX.

; Perform the and operation with last nibble.

;Rotate right side 4 times the BX.

;Perform the and operation with last nibble of BX.

;Compare AX with BX.

;If no zero go to TER label.

;Declared as a PALINDROME.

;Declared as a non palindrome.

Output: The word is nibble wise

palindrome

- 34 - 6. PROGRAMS INVOLVING BRANCH/LOOP INSTRUCTIONS / PROGRAMS ON

ARRAYS

i)ADDITION OF n NUMBERS

```
DATA SEGMENT    ;start of data segment
       ARR DW 0010H,0020H,0030H,0040H,0050H
LEN EQU ($-ARR)/2
SUM DW ?
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:  MOV AX,DATA
                    MOV DS,AX
   LEA SI,ARR
                    CLC
                    XOR AX,AX
                    MOV CX,LEN
UP:    ADC AX,[SI]
            INC SI
  INC SI
  DEC CX
  JNZ UP
  MOV SUM,AX
```

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

OUTPUT: 00F0

- 35 -

```
;end of data segment
;start of code segment
;initialize data segment
;SI points to the LSB of data ARR
;clear carry
;clear AX register
;load CX with the number of data words in
ARR
;add
the
numbe
r
pointe
d by SI
to A
registe
r
;point to the next data word
;decrement Cx
;and check if all numbers are added
if no then add
```

;store the addition result in user

defined memory location sum

;terminate the process

;end of code segment

ii)PROGRAM TO SUBTRACT N NUMBERS

```
DATA SEGMENT
        ARR DW 50H,10H,20H,10H,05H
LEN EQU ($-ARR)/2
DIF DW ?
DATA ENDS
CODE SEGMENT
      ASSUME CS:CODE,DS:DATA
      START:    MOV AX,DATA
MOV DS,AX
LEA SI,ARR
CLC
MOV CX,LEN-1
MOV AX,[SI]
UP:  SUB AX,[SI+2]
INC SI
INC SI
DEC CX
JNZ UP
MOV DIF,AX
MOV AH,4CH
INT 21H
CODE ENDS
END START
```

OUTPUT: 0005

;start of data segment

;end of data segment

;start of code segment

;initialize data segment

;SI points to the LSB of data ARR

;clear carry flag

;load CX register with the number of

data words in ARR

;make a copy of the first number

pointed by SI in AX

;subtract the next number from the

contents of AX and store the result in AX

;point to the next number

;decrement CX

;and check if all subtraction of all

numbers is complete if no then subtract

;store the difference in user defined

memory location DIFF

;terminate the process

;end of code segment

PROGRAMS TO FIND LARGEST AND SMALLEST NUMBER

iii)PROGRAM TO FIND LARGEST NUMBER AMONG THE SERIES

```
DATA SEGMENT
        X DW 0010H,52H,30H,40H,50H
LAR DW ?
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE,DS:DATA
    START:   MOV AX,DATA
MOV DS,AX
MOV CX,05H
LEA SI,X
MOV AX,[SI]
DEC CX
UP:  CMP AX,[SI+2]
JA CONTINUE
MOV AX,[SI+2]
  CONTINUE:ADD SI,2
DEC CX
JNZ UP
MOV LAR,AX
MOV AH,4CH
INT 21H
CODE ENDS
END START
;start of data segment
;end of data segment
```

;start of code segment

;initialize data segment

;load CX register with number of datawords

in X

;initialize SI to point to the first number

;make a copy of the number pointed by SI in

AX

;set count value in CX for comparison

;compare two adjacent numbers(one is in

AX and the other is pointed by SI+2)

;if contents of AX is greater than the next

number in array retain the contents of AX

;if not make a copy of the larger number in

AX

;point to the next number

;decrement CX to check if all numbers are

compared

;if no continue to compare

;if yes make a copy of AX(largest number)

in user defined memory location LAR

;terminate the process

;end of code segment

- 37 - iv)PROGRAM TO FIND THE LARGEST NUMBER USING DOS DISPLAY

INTERRUPTS

DATA SEGMENT

```
        X DW 0010H,0052H,0030H,0040H,0050H
;start of data segment
MES DB 10,13,'LARGEST NUMBER AMONG THE SERIES IS $'
DATA ENDS
CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
        START:  MOV AX,DATA
MOV DS,AX
MOV CX,05H
LEA SI,X
MOV AX,[SI]
DEC CX
UP:  CMP AX,[SI+2]
JA CONTINUE
MOV AX,[SI+2]
       CONTINUE:ADD SI,2
DEC CX
JNZ UP
AAM
ADD AX,3030H
MOV BX,AX
                        MOV AX,09H
LEA DX,MES
INT 21H
MOV DL,BH
MOV AH,02H
```

```
INT 21H
MOV DL,BL
INT 21H
MOV AH,4CH
INT 21H
CODE ENDS
END START
;end of data segment
;start of code segment
;initialize data segment
;load CX register with
number of datawords in array X
;SI points to start of dataword
array X
;make a copy of the first
number in AX
;initialize CX with count
value for comparison
;compare the contents of AX
and the number pointed by SI+2
;if AX is greater than the next
number in array then retain the
contents of AX
;else make a copy of the next
number (larger number)in AX
;point to next number in array
```

;decrement CX

;check if all numbers are

compared if no continue comparison

;if yes convert largest binary

number in AX to unpacked BCD

;convert unpacked BCD to

unpacked ASCII equivalent

;make a copy of it in AX

;display the message stored at

user defined memory location MES

;display the largest number

;terminate the process

;end of code segment

OUTPUT: LARGEST NUMBER AMONG THE SERIES IS 0052

v)PROGRAM TO FIND THE SMALLEST NUMBER AMONG THE SERIES

DATA SEGMENT

X DW 0060H,0020H,0030H,0040H,0050H

;start of data segment

MES DB 10,13,'SMALLEST NUMBER AMONG THE SERIES IS $'

DATA ENDS

```
CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
        START: MOV AX,DATA
MOV DS,AX
MOV CX,05H
LEA SI,X
MOV AX,[SI]
DEC CX
UP:  CMP AX,[SI+2]
JB CONTINUE
MOV AX,[SI+2]
      CONTINUE:ADD SI,2
DEC CX
JNZ UP
AAM
ADD AX,3030H
MOV BX,AX
MOV AH,09H
LEA DX,MES
INT 21H
MOV DL,BH
MOV AH,02H
INT 21H
```



```
;end of data segment
;start of code segment
```

```
;initialize data segment
;load CX with number of
datawords in array X
;SI points to the first number
in array X
;make a copy of the first
number in AX
;intialize CX with count
value for comparison
;compare the contents of AX
with next number in array pointed
by SI+2
;if AX is smaller than the
next number retain the contents of
AX
;else make a copy of the smaller
number in AX
;SI points to the next number
;decrement the count value
;check if all the numbers are
compared if no continue
comparison
;if yes convert the smallest
binary number to unpacked BCD
;convert the unpacked BCD
to unpacked ASCII equivalent
```

;make a copy of the unpacked

ASCII in BX

;display the message stored at

user defined memory location

MES using DOS interrupts

;display the smallest number

in array X using DOS interrupts MOV DL,BL

INT 21H

MOV AH,4CH

INT 21H

CODE ENDS

END START

;terminate the process

;end of code segment

OUTPUT: SMALLEST NUMBER AMONG THE SERIES IS 0020

- 40 - vi)PROGRAM TO SORT THE NUMBERS IN ASCENDING/DESCENDING ORDER

DATA SEGMENT

x DW 42H,34H,26H,17H,09H

LEN EQU 05

ASCD DB 10 DUP(0)

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE,DS:DATA

START: MOV AX,DATA

```
MOV DS,AX
MOV BX,LEN-1
MOV CX,BX
UP1:  MOV BX,CX
LEA SI,X
UP:  MOV AX,[SI]
MOV DX,[SI+2]
CMP AX,DX
                    JB DOWN/JA DOWN
MOV [SI],DX
MOV [SI+2],AX
      DOWN:  INC SI
INC SI
DEC BX
JNZ UP
DEC CX
JNZ UP1
MOV AH,4CH
INT 21H
CODE ENDS
END START
```

OUTPUT: 09 17 26 34 42

```
;start of data segment
;end of data segment
;start of code segment
;initialize data segment
```

```
;load BX(counter1) with count
value(number of datawords in array - 1)
;make a copy of the count value in CX(counter2)
;load the updated CX in BX
;SI points to the first number in the array
;make a copy of the number pointed by SI in
AX
;make a copy of the next number in DX
;compare both the numbers
;if AX < DX/AX > DX retain them as it is
;if not sort the numbers in ascending order
;point to the next number
;decrement the counter1
;compare till the larger number is sorted at
the end of the array
;decrement counter2
;compare till the numbers are sorted in
ascending order
;terminate the process
;end of code segment
```

- 41 - PROGRAMS ON STRING MANIPULATION LIKE STRING TRANSFER, STRING

REVERSING, SEARCHING FOR A CHARACTER IN A STRING AND PALINDROME

vii) PROGRAM FOR STRING TRANSFER

```
DATA SEGMENT
STR1 DB 'HOW ARE YOU'
LEN EQU $-STR1
STR2 DB 20 DUP(0)
DATA ENDS
CODE SEGMENT
;start of data segment
;end of data segment
;start of code segment
```

```
    ASSUME CS:CODE,DS:DATA,ES:DATA
    START:       MOV AX,DATA
MOV DS,AX
MOV ES,AX
LEA SI,STR1
LEA DI,STR2
MOV CX,LEN
CLD
REP MOVSB
MOV AH,4CH
INT 21H
CODE ENDS
END START
;initialize data segment
;initialize extra segment for string operations
;SI points to starting address of string at
;STR1
;DI points to starting address of where the
string has to be transferred
;load CX with length of the string
;clear the direction flag for auto increment SI
;and DI
;the source string is moved to destination
address till CX=0(after every move CX is
;decremented)
;terminate the process
```

```
;end of code segment
```

- 42 - viii)PROGRAM TO REVERSE A STRING

```
DATA SEGMENT
STR1 DB 'HELLO'
LEN EQU $-STR1
STR2 DB 20 DUP(0)
DATA ENDS
CODE SEGMENT
;start of data segment
;end of data segment
;start of code segment
        ASSUME CS:CODE,DS:DATA,ES:DATA
        START: MOV AX,DATA
MOV DS,AX
MOV ES,AX
```

```
LEA SI,STR1
LEA DI,STR2+LEN-1
MOV CX,LEN
UP:  CLD
LODSB
STD
STOSB
LOOP UP
MOV AH,4CH
INT 21H
CODE ENDS
END START
```

OUTPUT: OLLEH

```
;initialize data segment
;initialize extra segment for string operations
;SI points to the starting address of the string
at STR1
;DI points to the address of the last character in
the string(here address of '0')
;load CX with count value equal to number of
characters in the string
;clear the direction flag to autoincrement SI
register
;load AX with the character pointed SI
register
;set the direction flag to autodecrement DI
```

register

;the contents of AX is stored at the address

pointed by DI

;decrement CX and continue the transfer till

CX is zero

;terminate the process

;end of code segment

- 43 - ix)PROGRAM TO SEARCH FOR A CHARACTER IN A STRING

```
DATA SEGMENT
MSG DB 'HELLO'
CNT EQU $-MSG
        SRC EQU 'E'
;start of data segment
MSG1 DB 10,13,'CHARACTER FOUND$'
MSG2 DB 10,13,'CHARACTER NOT FOUND$'
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE,DS:DATA,ES:DATA
    START:    MOV AX,DATA
MOV DS,AX
MOV ES,AX
LEA SI,MSG
MOV AL,SRC
```

```
MOV CL,CNT
MOV CH,00H
CLD
UP:  SCASB
JZ DOWN
LOOP UP
LEA DX,MSG2
MOV AH,09H
INT 21H
JMP EXIT
;end of data segment
;start of code segment
;initialize data segment
;initialize extra segment
;SI points to the starting address of
the string
;the character to be searched in the
string is stored in AL
;CX is loaded with count value equal
to number of characters in the string
;clear the direction flag for
auto increment SI and DI
;check if the character in AL is the
same as that pointed by index register
;if it is same jump to label DOWN
;if not decrement CX and continue
```

checking till CX is zero

;display the message at MSG2 that is

CHARACTER NOT FOUND

;jump to label EXIT

DOWN: LEA DX,MSG1

MOV AH,09H

INT 21H

;if the character is found display the

message CHARACTER FOUND

EXIT: MOV AH,4CH

INT 21H

CODE ENDS

END START

OUTPUT: CHARACTER FOUND

 ;terminate the process

;end of code segment x

PROGRAM TO CHECK FOR PALINDROME

```
DATA SEGMENT
        STR1 DB 'LIRIL'
LEN EQU $-STR1
STR2 DB 20 DUP(0)
;start of data segment
MES1 DB 10,13,'WORD IS PALINDROME$'
MES2 DB 10,13,'WORD IS NOT PALINDROME$'
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE,DS:DATA,ES:DATA
    START: MOV AX,DATA
MOV DS,AX
MOV ES,AX
LEA SI,STR1
LEA DI,STR2+LEN-1
MOV CX,LEN
UP:  CLD
LODSB
```

```
STD
STOSB
LOOP UP
LEA SI,STR1
LEA DI,STR2
CLD
MOV CX,LEN
REPE CMPSB
CMP CX,0H
JNZ NOTPALIN
LEA DX,MES1
MOV AH,09H
```



```
- ;end of data segment
;start of code segment
;initialize data segment
;initialize extra segment for string
operations
;SI points to starting address of string
;DI points the last character in the
string
;load CX with count value equal to
number of characters in the string
;clear the direction flag to
auto increment SI
;get the character in AL from the
```

```
address pointed by SI
;set the direction flag equal to
auto decrement DI
;store the character in AL at address
pointed by DI
;decrement CX and continue with
reversing the string till CX=0
;SI points to the starting address of
original string
;DI points to the starting address of
the string reversed
;set CX as counter for checking if
palindrome
;compare the strings pointed by SI
and DI
;do the comparison till CX=0(if
palindrome)
;if CX is not zero then jump to
display WORD NOT PALINDROME
;display the message at MES1 which
is WORD IS PALINDROME INT 21H
JMP EXIT
  NOTPALIN: LEA DX,MES2
MOV AH,09H
INT 21H
EXIT:  MOV AH,4CH
```

```
INT 21H
CODE ENDS
END START
```

OUTPUT: WORD IS PALINDROME


```
;jump to end of the program
;display the message WORD NOT
PALINDROME using DOS
interrupts
;terminate the process
;end of code
```

segment 7.1.Program to use DOS interrupt INT 21H function called for reading a character from

keyboard, buffered keyboard input, display of character and string on console.

```
    DATA SEGMENT
INKEY DB ?
BUF DB 20 DUP(0)
MES DB 10,13, 'DAYANANDA SAGAR COLLEGE OF ENGINEERING $'
    DATA ENDS
    CODE SEGMENT
    ASSUME CS:CODE , DS:DATA
    START:   MOV AX,DATA
MOV DS,AX
MOV AH,01H
INT 21H
;DOS function to read a character from keyboard ;with
echo. [AL = 8bit character]
MOV INKEY,AL  ;Returns ASCII value of the pressed key.
MOV BUF,10
MOV AH,0AH
LEA DX,BUF
INT 21H
MOV AH,06H
MOV DL,'A'
```

```
INT 21H
 MOV AH,09H
LEA DX,MES
INT 21H
MOV AH,4CH
INT 21H
CODE ENDS
END START
;Load how many characters to enter.
;Dos function to read string of characters from
;keyboard.
;Dos function to display a character.
;Load the character to be displayed.
;Dos function to read string of characters from
;keyboard.
;DX = offset address of the message
```

- 47 - 2.Creation of a new file

```
DATA SEGMENT
FILENAME DB'HELLO.NEW'
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:   MOV AX,DATA
MOV DS,AX
MOV AH,3CH
MOV CX,0
;initialise data segment
;dos function call to create
;new file
;CX = file attribute
CODE
ENDS
END START
MOV DX,OFFSET FILENAME   ; dx has offset address of
;filename
INT 21H
```

- 48 - 3.Writing to a file

```
DATA SEGMENT
MES DB 10,13,'ENTER SOME DATA IN THE FILE$'
FILENAME DB 'HELLO.NEW'
BUFFER DB 22 DUP(0)
MES1 DB 10,13, 'ERROR IN FILE HANDLING$'
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:
ERROR:
EXIT:
CODE ENDS
END START
MOV AX,DATA
MOV DS,AX
MOV AH,09H
LEA DX,MES
INT 21H
MOV BUFFER,20
MOV AH,0AH
MOV DX,OFFSET BUFFER
INT 21H
MOV AH,3CH
MOV CX,0
```

```
MOV DX,OFFSET FILENAME
INT 21H
MOV BX,AX
MOV AH,40H  ;function to write in a file
MOV CX,20
MOV DX, OFFSET BUFFER
INT 21H
JC ERROR
JMP EXIT
MOV DX,OFFSET MES1
MOV AH,09H
INT 21H
MOV AH,4CH
INT 21H
```

```
DATA SEGMENT
YY DB
MM DB
D DB
TDAY DW UN,MON,TUE,WED,THU,FRI,SAT
SUN DB'SUNDAY,$'
MON DB'MONDAY,$'
TUE DB'TUESDAY,$'
WED DB'WEDNESDAY,$'
THU DB'THURSDAY,$'
FRI DB FRIDAY,$'
SAT DB'SATURDAY,$'
TMON DW JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
JAN DB'JANUARY,$'
FEB DB'FEBRUARY,$'
MAR DB'MARCH,$'
APR DB'APRIL,$'
MAY DB'MAY,$'
JUN DB'JUNE,$'
JUL DB'JULY,$'
AUG DB'AUGUST,$'
SEP DB'SEPTEMBER,$'
OCT DB'OCTOBER,$'
NOV DB'NOVEMBER,$'
DEC DB'DECEMBER,$'
DATA ENDS
```

```
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
DISCHAR MACRO CHAR
PUSH AX
PUSH DX
MOV DL,CHAR
MOV AH,02
INT 21H
POP DX
POP AX
ENDM
START:   MOV AX,DATA
MOV DX,AX
CALL PDATE
- 50 - MOV AH,4CHH
INT 21H
PDATE PROC NEAR
MOV AH,2AH
INT21H
MOV YY,CX
MOV MM,DH
MOV D,DL
MOV AH,0
ROL AX,1
MOV SI,OFFSET TDAY
ADD SI,AX
```

```
MOV DX,[SI]
MOV AH,09H
INT21H
MOV AL,D
MOV AH,00H
AAM
OR AH,AH
JZ DIGIT0
ADD AH,30H
DISCHAR AH
DIGIT0:
DIS19:
ADD AL,30H
DISCHAR AL
DISCHAR ''
MOV AL,MM
SUB AL,1
MOV AH,0
ROL AX,1
MOV SI,OFFSET TMON
ADD SI,AX
MOV DX,[SI]
MOV AH,09H
INT21H
MOV AX,YY
CMP AX,2000
```

JB DIS19

SUB AX,2000

DISCHAR'2'

DISCHAR'0'

JMP SKIP

SUB AX,1900

DICHAR'1'

DISCHAR'9'

- 51 - SKIP: AAM

ADD AX,3030H

DISCHAR AH

DISCHAR AL

RET

PDATE ENDP

CODE ENDS

END START

Output: sunday,september 25,2005

- 52 - 5.Set System Date

DATA SEGMENT

MES DB 10,13,'ENTER THE DATE WITH FORMAT: DD:MM:YY $'

MES1 DB 10,13,"DATE:$"

BUFF DB 10 DUP(0)

```
DB 0
DB 10 DUP(0)
YY DB ?
MM DB ?
D DB ?
DATA ENDS
CODE SEGMENT
        ASSUME CS:CODE, DS:DATA
        START: MOV AX,DATA
MOV DS,AX
CALL DATEP
MOV AH,4CH
INT 21H
DATEP PROC NEAR
MOV AH,09H
LEA DX,MES
INT 21H
MOV AH,09H
LEA DX,MES1
INT 21H
MOV AH,0AH
LEA DX,BUFF
INT 21H
MOV CL,04
MOV DL,0H
LEA SI,BUFF
```

```
ADD SI,02
BACK:  MOV AL,[SI]
CMP AL,':'
JZ TER
ROL DL,CL
SUB AL,30H
```



```
TER:
BACK1:
TER1:
BACK2:
TER2:
ADD DL,AL
INC SI
JMP BACK
MOV DH,DL
ADD DL,0F0H
ROR DL,CL
MOV AL,10
MUL DL
AND DH,0FH
ADD AL,DH
MOV DH,AL
MOV DL,0
INC SI
MOV AL,[SI]
CMP AL,':'
```

```
JZ TER1
ROL DL,CL
SUB AL,30H
ADD DL,AL
INC SI
JMP BACK1
MOV DH,DL
AND DL,0F0H
ROR DL,CL
MOV AL,10
MUL DL
AND DH,0FH
ADD AL,DH
MOV MM,AL
MOV DL,0
INC SI
MOV AL,[SI]
CMP AL,13
JZ TER2
ROL DL,CL
SUB AL,30H
ADD DL,AL
INC SI
JMP BACK2
MOV DH,DL
AND DL,0F0H
```

- 54 - ROR DL,CL

```
MOV AL,10
MUL DL
AND DH,0FH
ADD AL,0DH
MOV YY,AL
MOV AH,2BH
MOV CL,YY
MOV CH,00
ADD CX,2000
MOV DH,MM
MOV DL,0DH
INT 21H
RET
DATEP ENDP
CODE ENDS
END START
```

- 55 - 6.READ SYSTEM TIME

```
DATA SEGMENT
  HOUR DB ?
  MIN DB ?
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
DISCHAR MACRO CHAR
                PUSH AX
```

```
PUSH DX
                MOV DL,CHAR
  MOV AH,02
                INT 21H
                POP DX
                POP AX
                ENDM
START:   MOV AX,DATA
    MOV DS,AX
                CALL TIME
                MOV AH,4CH
                INT21H
TIME PROC NEAR
MOV AH,2CH                  ;function to read system time
INT21H
MOV HOUR,CH
MOV MIN,CL
CMP CH,12
JB DOWN
SUB CH,12
DOWN:  MOV AL,CH
MOV AH,00H
AAM
MOV AX,3030H
DISCHAR AH
DISCHAR AL
```

```
DISCHAR':'
MOV AL,CL
MOV AH,00H
AAM
- 56 - ADD AX,3030H
DISCHAR AH
          DISCHAR AL
                    DISCHAR' '
                   CMP HOUR,12
                    JB AM
                    DISCHAR 'P'
                  JMP DOWN1
AM:               DISCHAR'A'
DOWN1:       DISCHAR'M'
                    RET
TIME ENDP
CODE ENDS
END START
```

- 57 - 7.Set system time

```
DATA SEGMENT
MES DB 10,13,’ENTER TIME WITH THE FORMAT :HOUR FOLLOWED BY MIN
FOLLOWED BY AM OR PM$’
MES1 DB 10,13 ,’TIME:$’
BUF DB 10
DB 0
DB 10 DUP(0)
HOUR DB?
MIN DB?
DATA ENDS
CODE SEGMENT
```

```
ASSUME CS:CODE, DS:DATA
START:  MOV AX,DATA
MOV DS,AX
CALL TIME
MOV AX,4CH
INT 21H
TIME PROC NEAR
MOV AH,09H
LEA DX,MES
INT  21H
MOV AH,09H
LEA DX,MES1
INT 21H
MOV AH,0AH
LEA DX,BUF
INT 21H
MOV CL,4
MOV DL,00H
LEA SI,BUF
ADD SI,2
UP:
DOWN:
MOV AL,[SI]
CMP AL,':'
JZ DOWN
ROL DL,CL
```

```
SUB AL,30H
ADD DL,AL
INC SI
JMP UP
MOV DH,DL
- 58 - UP1:
AND DL,0F0H
ROR DL,CL
MOV AL,10
MUL DL
AND DH,0FH
ADD AL,DH
MOV HOUR,AL
MOV DL,0
INC SI
MOV AL,[SI]
CMP AL,’ ’
JZ DOWN1
ROL DL,CL
SUB AL,30H
ADD DL,AL
INC SI
JMP UP1
DOWN1:   MOV DH,DL
AND DL,0F0H
ROR DL,CL
```

```
MOV AL,10
MUL DL
AND DH,0FH
ADD AL,DH
MOV MIN,AL
INC SI
MOV CH,[SI]
CMP CH,'P'
JNZ SKIP
ADD HOUR,0CH
SKIP:
CODE ENDS
END START
MOV AH,2DH
MOV CH,HOUR
MOV CL,MINUTE
MOV CX,0000H
INT 21H
RET
TIME ENDP
```

1)MATRIX KEYBOARD INTERFACING

```
DATA SEGMENT
PORTA EQU 120H
PORTC EQU 122H
CWRD EQU 123H
ARRAY DB '0123456789.+-*/%ACK=MMMM'
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE,DS:DATA
START:  MOV AX,DATA
MOV DS,AX              ;initialise data segment
MOV AL,90H             ;initialise 8255 porta as i/p and portc as o/p
MOV DX,CWRD
OUT DX,AL
REPEAT:   MOV DX,PORTC     ;make first row of the keyboard high through pc0
```

```
MOV AL,01
OUT DX,AL
MOV DX,PORTA
IN AL,DX              ; input contents of porta and check if key is pressed-
CMP AL,00             ; in first row.
JZ NEXT
JMP FIRSTROW
NEXT:   MOV DX,PORTC    ;if key not found in first row, check if key is in
                                                ;second row
MOV AL,02
OUT DX,AL
MOV DX,PORTA
IN AL,DX
CMP AL,00
JNZ SECONDROW
MOV AL,04             ; if key not found then check for key closure in
        ;third row
MOV DX,PORTC
OUT DX,AL
MOV DX,PORTA
IN AL,DX
CMP AL,00H
JNZ THIRDROW
JMP REPEAT
FIRSTROW:   CALL DELAY          ;check all the keys one by onein first row
LEA SI,ARRAY
```

```
- 60 - UP:  SHR AL,1
JC DISPLAY                    ;if key found jump to the display subroutine
INC SI
JMP UP
JMP DISPLAY
SECONDROW:CALL DELAY
LEA SI,ARRAY+08H      ;second row keys from array +08
UP1:SHR AL,1
JC DISPLAY                    ;if key found jump to the display subroutine
INC SI
JMP UP1
THIRDROW: CALL DELAY
LEA SI,ARRAY+10H      ;third row keys from array +16(dec)
UP2:    SHR AL,1
JC DISPLAY                    ;if key found jump to the display subroutine
INC SI
JMP UP2
JMP DISPLAY
DISPLAY:  MOV DL,[SI]
CMP DL,97                     ;24 in decimal. 8x3rows = 24keys
JZ EXIT
MOV AH,02H                    ; display key no in ascii
INT 21H
JMP REPEAT
DELAY:
L1:
```

```
L2:
CODE ENDS
END START
MOV BX,0FFFFH
MOV CX,0FFFH
DEC CX
JNZ L2
DEC BX
JNZ L1
RET
EXIT:MOV AH,4CH
INT 21H
```

2)SEVEN SEGMENT DISPLAY INTERFACE

```
- 61 - DATA SEGMENT
PORTA EQU 120H
PORTB EQU 121H
PORTC EQU 122H
CWRD EQU 123H
TABLE DB 8CH,0C7H,86H,89H
DATA ENDS
CODE SEGMENT
   ASSUME CS:CODE, DS:DATA
   START:  MOV AX,DATA       ;intialise data segment
MOV DS,AX
MOV AL,80H            ;initialise 8255 portb and portc as o/p
MOV DX,CWRD
OUT DX,AL
MOV BH,04              ; BH = no of digitsto be displayed
LEA SI,TABLE        ; SI = starting address of lookup table
  NEXTDIGIT:MOV CL,08          ; CL = no of segments = 08
MOV AL,[SI]
  NEXTBIT:  ROL AL,01
```

```
MOV CH,AL              ;save al
MOV DX,PORTB      ;one bit is sent out on portb
OUT DX,AL
MOV AL,01
MOV DX,PORTC     ;one clock pulse sent on pc0
OUT DX,AL
DEC AL
MOV DX,PORTC
OUT DX,AL
MOV AL,CH              ; get the sevensegment code back in al
DEC CL                 ;send all 8 bits,thus one digit is displayed
JNZ NEXTBIT
DEC BH
INC SI
JNZ NEXTDIGIT
;display all the four digits
MOV AH,4CH          ;exit to dos
INT 21H
        CODE ENDS
        END START
```

- 62 - 3) LOGICAL CONTROLLER INTERFACE

```
DATA SEGMENT
PA EQU 120H       ;INITIALIZE THE ADDRESS OF PORT A OF 8255
PB EQU 121H       ;INITIALIZE THE ADDRESS OF PORT B OF 8255
PC EQU 122H       ;INITIALIZE THE ADDRESS OF PORT C OF 8255
CR EQU 123H       ;INITIALIZE THE ADDRESS OF  CONTROL WORD
REGISTER
 DATA ENDS
 CODE SEGMENT
 ASSUME CS:CODE, DS:DATA
START:    MOV AX, DATA
MOV DS, AX
MOV AX, 082H      ;load the control word
MOV DX, CR
OUT DX,AX
REPEAT:    MOV DX, PB
IN AL,DX
AND AL, 03H
CMP AL,00H
JZ DISPLAY
;input the data (from dip switch)from port b
CMP AL,03H            ;check if input is 11
```

```
JZ DISPLAY
MOV AL,0FFH        ;display 11 if input is 01 else 10
MOV DX, PA         ;output to porta
OUT DX,AL
JMP REPEAT
DISPLAY:  MOV AL,00H     ;display oo if input is 00 else 11
CODE ENDS
END START
MOV DX,PA        ;output to porta
OUT DX, AL
JMP REPEAT
```

- 63 -4)STEPPER MOTOR INTERFACE

```
DATA SEGMENT
PORTA EQU 120H
PORTB EQU 121H
PORTC EQU 122H
CWRD EQU 123H
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:
UP:
DELAY:
UP2:
UP1:
CODE ENDS
MOV AX,DATA
MOV DS,AX
MOV AL,80H          ;initialise 8255 ,porta as o/p port
MOV DX,CWRD
OUT DX,AL
MOV DX,PORTA
MOV AL,88H          ;load initial bit pattern
OUT DX,AL           ;output on porta
CALL DELAY
```

```
ROL AL,01H          ;rotate left to get exitation sequence of 11,22,44,88
OUT DX,AL
JMP UP
MOV CX,0FFFFH  ;delay can be adjusted to get different speeds
MOV BX,0FFH
DEC BX
JNZ UP1
DEC CX
JNZ UP2
RET
MOV AH,4CH
INT 21H
 END START
```

VISVESWARAIAH TECHNOLOGICAL UNIVERSITY,BELGAUM

- 64 - Branch: ELECTRONICS AND COMMUNICATION Semester: V

Subject Code: ECL57 Duration Of Exam:3hrs

Subject Title: Advanced Microprocessor Lab Max Exam Marks: 50

QUESTION BANK

PART – A

1A) Write an ALP to show the byte and word transfers in different addressing modes.

1B) Write an ALP to transfer a given block of data word from source memory to Destination memory without overlap.

1C) Write an ALP to transfer a given block of data word from source memory to destination memory with overlap.

1D) Write an ALP to interchange two blocks of data.

2A) Write an ALP to add / subtract two16 bit numbers.

2B) Write an ALP to add / subtract two32 bit numbers.

2C) Write an ALP to add / subtract two32 bit numbers using DD Directive.

2D) Write an ALP to multiply two 16 bit unsigned / signed numbers and display.

2E) Write an ALP to divide two 8 bit numbers (signed and unsigned)

2F). Write an ALP to divide two 16 bit numbers(signed and unsigned)

3A)Write an ALP to to add/substract/ multiply/divide two ASCII numbers.

3B) Write an ALP to convert 16 bit binary No to BCD.

3C) Write an ALP to convert BCD No to binary.

3D) Write an ALP to find square and cube of an 8 bit number .

3E) Write an ALP to find LCM of a 16 bit No.

3F) Write an ALP to find the GCD of two 16 bit unsigned integers.

3G) Write an ALP to find the factorial of an 8 bit number.

4A) Write an ALP to check if the number is positive or negative.

4B) Write an ALP to check if the given number is even or odd.

4C) Write an ALP to check number of ones and zeroes n the given data.

4D) Write an ALP to check if the given byte is 2 out of 5 code or not (i.e., the code is first 3 MSB must be 0 and the last 5 LSB should have two 1s).

4E) Write and Alp to check if the given 16 bit data is a palindrome (bitwise).

4F. Write and Alp to check if the given 16 bit data is a palindrome (nibble-wise).

5A) Write an Alp to add / substract ‘N’ 16 bit numbers and display the result.

5B)Write an ALP to find the largest of ‘N’ 16 bit numbers and display the result.

5C) Write an ALP to find the smallest of ‘N’ 16 bit numbers and display the result.

5D) Write an ALP to sort the given set of 16 bit unsigned integers in ascending order using bubble sort algorithm.

- 65 - 6A) Write an ALP to transfer a given source string to destination using string instructions.

6B) Write an ALP to reverse a string.

6C) Write an ALP to search for a character in a string.

6D) Write an ALP to check if the given string is a palindrome or not.

7A) Write an ALP to read a character from a keyboard with and without echo.

7B) Write an ALP to read a string of characters from the keyboard and display.

7C) Write an ALP to create a new file.

7D) Write an ALP to read the contents of a file.

7E) Write an ALP to write a new file.

8A) Write an ALP to read the system date.

8B)Write an ALP to set the system date.

8C) Write an ALP to read the system time.

8D) Write an ALP to set the system time.

PART - B

1A) Write an ALP to scan the keyboard for key closure and store the code of the key pressed in memory location.

1B) Write an ALP to implement a rolling display of set characters using a display interface.

1C) Interface a logic controller via 8255 using I/O cards and perform the following Operations: Read all the 8 inputs from the logic controller, Complement /XOR/AND/OR/NAND and display at the output.

1D) Write an ALP to control the speed of a stepper motor & to drive the stepper motor interface to rotate the motor in clockwise and anticlockwise directions.

VIVA QUESTIONS IN ADVANCED MICROPROCESSOR

- 66 - 1. List all the modern microprocessor

2. Name some 16 bit Processor (8086, 80286, 80386L, EX)

3. Name some 32 bit processors (80386DX, 80486, PENTIUM OVERDRIVE)

4. Name some 64 bit processor (Pentium, Pentium pro, Pentium II, Xeon, Pentium III, and

Pentium IV)

5. List the address bus width and the memory size of all the processor.

Processor

8086

8088

80186

80286

80386

80386DX

80386EX

80486

address bus

20

20

20

24

24

32

26

32

memory size

1M

1M

1M

16M

16M

4G

64M

4G

PENTIUM 64

PENTIUM O 32

4G

4G

PENTIUM P 32 4G

PENTIUM 2,3,4 36 64G

6. The memory map of any IBM COMPATIBLE PC consists of three main parts, name them

[transient memory area, system area, Extended memory system]

7. The first I MB of the memory area is called as ..... (Real memory area)

8. What does the TPA hold (interrupt vectors, bios, DOS, IO.SYS, MSDOS, DEVICE

DRIVERS, command.com)

9. The system area contain programs in .....memory(ROM)

10. What are the main two parts of 8086 internal architecture.(BIU,EU)

11. Name the registers in BIU (CS, DS, ES, SS, IP)

12. Name the registers in EU.(AX, BX, CX, DX, SP, BP, SI, DI)

13. Name the flag registers in 8086. (O, D, I, T, S, Z, A, P, C)

14. How is the real memory semented?

15. What is the advantage of segmentation.

16. Name the default segment and offset register combinations.

17. What is the relocatable program.

18. Name the three main addressing modes in 8086.

19. Name the data addressing modes. And the program addressing modes. Give examples

20. Explain MOV AL, ‘A’, MOV AX, NUMBER, MOV [BP], DL, MOV CH,[1000],

MOV[BX+SI],SP, MOV ARRAY[SI],BL, MOV DH,[BX+DI+10H]

21. Name the programme memory addressing modes. (Direct, relative, indirect)

22. What is an intersegment and intra segment jump.

23. Differentiate near and short jumps (+_32k and +127to_128 bytes)

24. Differentiate near and far jumps.

25. Differentiate push and pop instructions.

26. Explain PUSH word ptr [BX], POP F.

27. JMP TABLE[BX]

28. Explain the following : ASSUME,DB,DD,DW,DQ,END - 67 - 29. Give the opcode format for 8086 instructions.

(op(1-2b),(mode,reg,rem),(displacement-0-2b))

30. Explain LES BX, LEA AX, DATA, LDS DI,LIST

31. Explain how the string instructions are executed.

32. List some string instructions

33. Explain the significance of REP Prefix.

34. Explain XCHG, LAHF, SAHF, XLAT

35. What are the two types of I/O addressing modes. (fixed port ,variable port)

36. What do you mean by segment override prefix.

37. Explain the following directives. NEAR ,FAR,BYTE PTR,ORG,OFFSET,ORG

38. Differentiate END, ENDP, ENDM

39.Differntiare PROC AND

40. What are the two basic formats used by assemblers. Where are they used.

(Models, full segment definition)

41. Explain ADD BYTE PTR (.model tiny (64kb), .model small(128 kb), .model huge.

42. Explain ADD BYTE PTR [DI], 3, SBB BYTE PTR [DI],5, CMP[DI], CH

IMUL BYTE PTR [BX], IDIV SI, CWD, CBW.

43. DAA, (ONLY ON AL), AAA, AAD, AAM, AAS.

44. Name the logical instructions. How can we invert number .(XOR WITH 1s)

45. Differentiate TEST and CMP, and NOT& NEG, SAR & SHR, RCL & ROL, SCAS &

CMPS, REPE SCASB &REPNE &SCASB

46. Which are the flags affected. JA(Z=0 C=0), JB(C=0), JG (Z=0 S=0), JLE(Z=1 S<>0)

47. LOOP, LOOPNE, LOOPE LOOPZ

48. Differentiate NEAR & FAR CALL, NEAR RET & FAR RET

49. Explain, maskable, non maskable, vectored, non vectored, software & Hardware Interrupts.

50. What are interrupt vectors. (4 byte no. stored in the first 1024 bytes of memory. There are

256 interrupt vectors. Each vector contains value of CS & IP, 32 vectors are reserved for

present and future. 32 to 255 are available for users.

51. Name the interrupt instructions. (INT, INT0, INT3)

52. Give significance of INT0, INT3.

53. Give the significance of IRET instruction how is it different from RET.

(Like far RET retrieves 6 bytes from stack, two for IP, two for CS and two for flags.)

54. Explain the operation of real mode interrupt.

55. Explain the protected mode interrupt.

56. Explain how the interrupt flag bit IF and TF are used during an interrupt

57. Name the hardware and soft ware interrupt of 8086, explain about them. (NMI, INTR are

hardware interrupts. INT, INT0, INT3, BOYND, are the software interrupts)

58. How can you expand the interrupt structure. (using 74LS 244 7 more interrupts can

accommodated. Daisy chained interrupt is better as it requires only one interrupt vector.)

59. Give a general description of 8259 interrupt controller.

61. Explain the above pins of 8086 TEST, READY, RESET, BHE/S7, MN/MX, ALE, DT/R,

DEN, HOLD, HLDA, SO, RO/GT1, LOCK, QS1-QS0.

62. Name the maximum mode pins.

63. Name the minimum mode pins.

64. Name the function of 8284

65 How does the RESET function.

- 68 - 66. What is the clock frequency of the 8086.

67. How are the address and data buses are separated.

68. What is the function of the following 74LS373, 245, 244

69. Differentiate between minimum mode and maximum mode of operation.

70. What are the two methods of interfacing memory. (linear and absolute decoding)

71. What do you understand by linear and absolute decoding.

72. What is the maximum memory capacity of 8086

73. Name the difference between 8086,8088.

74, Name the difference between 8085 and 8086.

75. Name the types of memory used in microprocessor based system.

76. What is the function of the 8288 controller

77. What are the various signals in a RAM and ROM memories.

78. Name the following. 8255, 8155, 8259, 8253, 8257, 8251

79. Give the format of control word register.

80. Explain the PPI you know.

81. Explain the modes of 8255.

82. Explain the basic function of 8279.

83. How are the delays obtained in a microprocessor based system.

84. What is an arithmetic coprocessor, What are its functions. (multiply, devide, ad, subtract,

square root, calculate partial tangent, partial arctangent and logarithms)

85. What are the data types used. (16,32, 64 bit signed integers, 18 bit BCD data, 32, 64 and

80 bit floating point nos.)

86. What are the advantages of the 8087 coprocessor. (many times faster than the

microprocessor)

87. How can we use the DOS function calls.

88. What is the function of int21 interrupts.

89. Explain PUBLIC and EXTERN statements.

90. What do you mean by modular programming, how is it accomplished in 8086.

91. what are libraries.

92. Differentiate between MACRO and PROCEDURE.

93. What are the conditional statements used in a MACRO. (REPEAT, WHILE)

94. What are the different methods of reading the keyboard using DOS function calls.

95. How can we use XLAT instruction for look up tables.

96. What are the two methods of interfacing I/O (memory mapped I/O and I/O mapped I/O)