MICROPROCESSORS & MICROCONTROLLERS Laboratory (EC-353) Manual

Prepared by

A.M.V.N. MARUTHI, M.Tech

(Lecturer)

&

Y.SRI CHAKRAPANI, M.Tech

(Lecturer)

DEPARTMENT OF ECE

BAPATLA ENGINEERING COLLEGE

BAPATLA

LIST OF EXPERIMENTS

Experiments Based on ALP (8086)

Experiments Based on Interfacing & Microcontroller (8051)

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 assemble 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 **OBJEC**T 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 is 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 (F8) or burst execution (F5).

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
                                       ;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
                                       ;memory location addressed by BX+0002H  into
                                       ;(16 bit)
            MOV  AL,[DI+2]             ;register relative addressing
            MOV AX,[BX+DI+0002H] ;copies the word content  of  data  segment
```

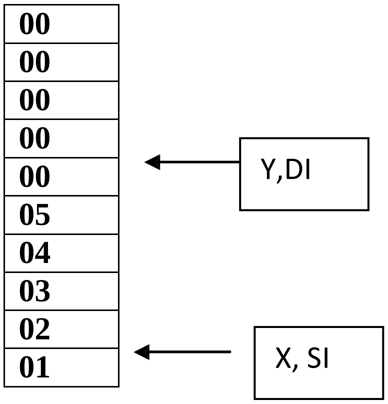
```
                                        ;memory location addressed by BX+DI+0002H
                                         ;into   AX(16 bit)
          MOV  AH,4CH                   ; Exit to DOS with function call 4CH INT
          21H

CODE ENDS                               ; Assembler stop reading
END START
```

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



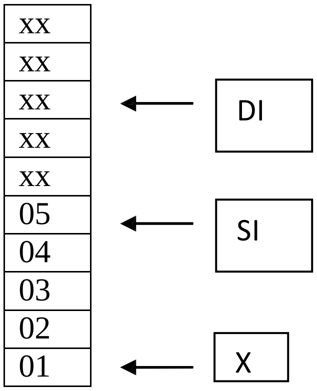
After execution

05
04
03
02
01
05
04
03
02
01

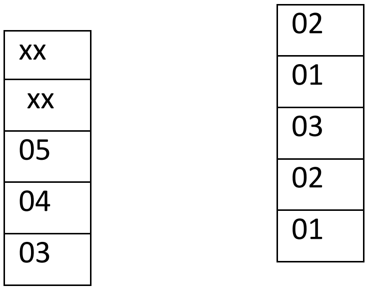
With Overlapping

```
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



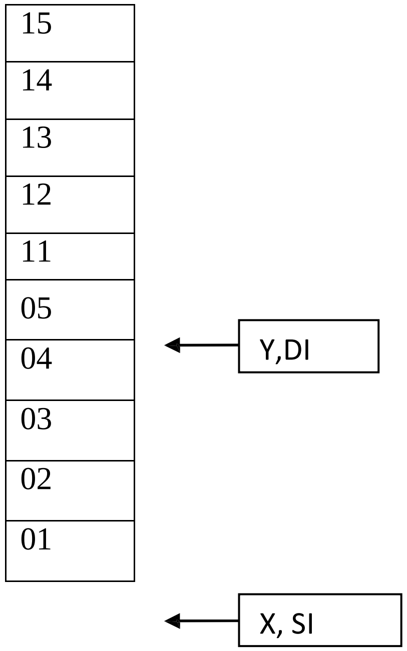
DS After execution



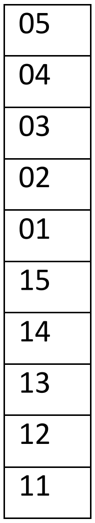
iii) Block Interchange

```
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



DS After execution



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
```

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                      ;Move  LSB  of   NUM1  to  AX
               ADD AX,NUM2                       ;Add   LSB  of  NUM2    to  AX
               MOV SUM,AX                        ;Store the LSB   in SUM
               MOV AX,NUM1+2                     ; Move  MSB  of   NUM1  to  AX
               ADC AX,NUM2+2                     ; Add   MSB  of  NUM2    to  AX
               JNC DOWN                          ; Check for carry
               MOV SUM+4,01H                     ; Store the carry in SUM+4
DOWN:  MOV SUM+2,AX                              ; Store the MSB in SUM+2
              MOV AH,4CH
              INT 21H
   CODE ENDS
   END START
```

```
   INPUT: 0FFFFFFFFH, 011111111H
  OUTPUT: 0111111110H
```

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

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                     ; SI pointed to the LSB of NUM1
           LEA DI,NUM2                     ; DI pointed to the LSB of NUM2
           MOV AX,[SI]                     ; Move the content of SI to AX
           MOV BX,[DI]                     ; Move the content of DI to BX
           SUB AX,BX                       ; Subtract from BX to AX
           MOV DIF,AX                      ; Store the LSB result in DIF
             INC SI                           ;Update SI to point the MSB of NUM1(if
                                            ;ADD SI,02 instruction its affect carry flag)
           INC SI
           INC DI                          ;Update DI to point the MSB of NUM2
           INC DI
           MOV AX,[SI]                     ; Move the content of SI to AX
           MOV BX,[DI]                     ; Move the content of DI to BX
           SBB AX,BX                       ;  Subtract with borrow from BX to AX
           MOV DIF+2,AX                    ;  Store the MSB result in DIF+2
           MOV AH,4CH
           INT 21H
CODE ENDS
END START
```

```
INPUT: 23456762,-11111111
OUTPUT:12345651

INPUT:11111111,-23451234
OUTPUT:EDCBFEDD
```

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

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.
```

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              ; Move the higher bit of Dividend  to DX
           DIV NUM2                   ; Perform the Division operation
           MOV QUO,AX                 ; Store the quotient to AX
           MOV REM,DX                 ; Store the reminder to DX
           MOV AH,4CH
           INT 21H
CODE ENDS
END START
```

INPUT: Dividend - 23454567,
Divisor - 4111,

0UTPUT: AX – 8AC5H (quotient); DX – 0952H (reminder);

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                ; Move the higher bit of Dividend  to DX
           CWD
           IDIV NUM2                     ; Perform the sign Division operation using IDIV
           MOV QUO,AX                    ; Store the quotient to AX
           MOV REM,DX                    ; Store the reminder to DX
           MOV AH,4CH
           INT 21H
CODE ENDS
END START
```

```
INPUT: Dividend - -44444444,
           Divisor     - 2222,
0UTPUT: AX – FE (quotient);
             DX – FF (reminder)                ; Result is in two complement form.
```

3)PROGRAMS ON BRANCH INSTRUCTIONS

i)To find weather is even or odd

```
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               ;Test for Even/Odd number.
          JNZ EXIT                  ; if it is Even go to Exit label.
          MOV BL,2
          DIV BL
          CMP AH,0H
          JNZ EXIT
          LEA DX,MSG1               ;Declare it is Even number.
          MOV AH,09H
          INT 21H
          JMP LAST
  EXIT: LEA DX,MSG2                 ;Declare it is Odd number.
          MOV AH,09H
          INT 21H
  LAST:   MOV AH,4CH INT
          21H

CODE ENDS
END START
```

Result: Output: Number is ODD

ii)To find number of 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                    ;Initialize BL to 8.
       MOV CL,1                    ;Initialize CL to 1.
  UP:  ROR AH,CL                   ;Perform the single bit rotate operation
                                   ;with respect to right.
       JNC DOWN                    ;If no carry go to DOWN label.
       INC ONE                     ;Increment one.
       JMP DOWN1                   ;Jump  to  DOWN1.
  DOWN:   INC ZERO                 ;Increment    ZERO.
  DOWN1:  DEC BL                   ;Decrement the BL.
       JNZ UP                      ;If no zero go to UP label.

       MOV AH,4CH
       INT 21H

CODE ENDS
END START
```

```
Output: Ones--------04
        Zeros--------04
```

iii)Program to find largest number among the given data

```
DATA SEGMENT                                  ;start of data segment
        X DW 0010H,52H,30H,40H,50H
        LAR DW ?
DATA ENDS                                     ;end of data segment

CODE SEGMENT                                  ;start of code segment
     ASSUME CS:CODE,DS:DATA
     START:   MOV AX,DATA                     ;initialize data segment
                MOV DS,AX
                MOV CX,05H                    ;load CX register with number of datawords
                                               in X
                LEA SI,X                      ;initialize SI to point to the first number
                MOV AX,[SI]                   ;make a copy of the number pointed by SI in
                                               AX
                DEC CX                        ;set count value in CX for comparison
        UP:     CMP AX,[SI+2]                 ;compare two adjacent numbers(one is in
                                              AX and the other is pointed by SI+2)
                JA CONTINUE                   ;if contents of AX is greater than the next
                                              number in array retain the contents of AX
                MOV AX,[SI+2]                 ;if not make a copy of the larger number in
                                               AX
  CONTINUE:ADD SI,2                           ;point to the next number
                DEC CX                        ;decrement CX to check if all numbers are
                                              compared
                JNZ UP                        ;if no continue to compare
                MOV LAR,AX                    ;if yes make a copy of AX(largest number)
                                              in user defined memory location LAR
                MOV AH,4CH                    ;terminate the process
                INT 21H
CODE ENDS                                     ;end of code segment
END START
```

4) PROGRAM USING SUBROUTINES:

PROGRAM TO FIND FACTORIAL OF A NUMBER USING PROCEDURE

```
NUM EQU 3
MSG DB 'FACTORIAL OF ',NUM+'0',' IS:'
ASCRES DB 4 DUP(?),'H',0DH,0AH,'$'
RES DW ?
HEXCODE DB '0123456789ABCDEF'
     .CODE
HEX_ASC PROC
     MOV DL,10H
     MOV AH,0
     MOV BX,0
     DIV DL                            ;DIV AL/DL WHERE AL=CHAR & DL=10H
     MOV BL,AL                         ;AL=QUOTIENT
     MOV DH,HEXCODE[BX]
     MOV BL,AH                 ;AH=REMAINDER
     MOV DL,HEXCODE[BX]
     RET
HEX_ASC ENDP

FACT PROC
     CMP AX,01                         ;IF N=1, FACT=1  ELSE FACT=N*FACT(N-1)
     JE EXIT
     PUSH AX
     DEC AX                            ;N-1
     CALL FACT                         ;FACT(N-1)
     POP AX
     MUL RES                           ;N*FACT(N-1)
     MOV RES,A       X                 ;RES=FACTORIAL
     RET
EXIT:
     MOV RES,01
     RET
FACT ENDP

MAIN:
     MOV AX,@DATA
     MOV DS,AX
     MOV AX,NUM                ;AX=N
     CALL FACT
     MOV AL,BYTE PTR RES+1            ;CONVERT MSB OF RESULT TO ASCII
     CALL HEX_ASC
```

```
        MOV ASCRES,DH
        MOV ASCRES+1,DL
        MOV AL,BYTE PTR RES      ;CONVERT LSB OF RESULT TO ASCII
        CALL HEX_ASC
        MOV ASCRES+2,DH
        MOV ASCRES+3,DL
        MOV AH,09H
        MOV DX,OFFSET MSG        ;DISPLAY MSG
        INT 21H
        MOV AH,4CH                       ;EXIT
        INT 21H
        ALIGN 16
END MAIN
```

Output:

```
Factorial of the number is 06
```

5)PROGRAM TO SORT THE ARRAYS

```
DATA SEGMENT                                    ;start of data segment
      x DW 42H,34H,26H,17H,09H
      LEN EQU 05
      ASCD DB 10 DUP(0)
DATA ENDS                                       ;end of data segment

CODE SEGMENT                                    ;start of code segment
    ASSUME CS:CODE,DS:DATA
    START:  MOV AX,DATA                         ;initialize data segment
            MOV DS,AX
            MOV BX,LEN-1                        ;load BX(counter1) with count
                                                value(number of datawords in array - 1)
            MOV CX,BX                           ;make a copy of the count value in CX(counter2)
      UP1:  MOV BX,CX                           ;load the updated CX in BX
            LEA SI,X                            ;SI points to the first number in the array
      UP:   MOV AX,[SI]                         ;make a copy of the number pointed by SI in
                                                AX
            MOV DX,[SI+2]                       ;make a copy of the next number in DX
            CMP AX,DX                           ;compare both the numbers
            JB DOWN/JA DOWN                     ;if AX < DX/AX > DX retain them as it is
            MOV [SI],DX                         ;if not sort the numbers in ascending order
            MOV [SI+2],AX
   DOWN:    INC SI                              ;point to the next number
            INC SI
            DEC BX                              ;decrement the counter1
            JNZ UP                              ;compare till the larger number is sorted at
                                                the end of the array
            DEC CX                              ;decrement counter2
            JNZ UP1                             ;compare till the numbers are sorted in
                                                ascending order
            MOV AH,4CH                          ;terminate the process
            INT 21H
CODE ENDS                                       ;end of code segment
END START
```

OUTPUT: 09 17 26 34 42

6)PROGRAM TO USE SOFTWARE AND HARDWARE INTERRUPTS FOR RECEIVING A INPUT FROM KEY BOARD AND DISPLAY IT ON SCREEN.

```
DATA SEGMENT
     INKEY DB ?
     BUF DB 20 DUP(0)
     MES DB 10,13, BAPATLA EINGINEERING COLLEGE $' DATA ENDS

CODE SEGMENT
ASSUME CS:CODE , DS:DATA

START:     MOV AX,DATA
           MOV DS,AX
           MOV AH,01H             ;DOS function to read a character from keyboard ;with
                                  echo. [AL = 8bit character]
           INT 21H
           MOV INKEY,AL           ;Returns ASCII value of the pressed key.
           MOV BUF,10             ;Load how many characters to enter.
           MOV AH,0AH             ;Dos function to read string of characters from
                                  ;keyboard.
           LEA DX,BUF
           INT 21H
           MOV AH,06H             ;Dos    function    to    display    a
           character. MOV DL,'A';Load the character to be displayed.
           INT 21H

            MOV AH,09H            ;Dos function to read string of characters from
                                  ;keyboard.
           LEA DX,MES             ;DX = offset address of the message
           INT 21H
           MOV AH,4CH
           INT 21H
     CODE ENDS
     END START
```

7)PROGRAM TO FIND THE LARGEST NUMBER USING DOS DISPLAY INTERRUPTS

```
DATA SEGMENT                                              ;start of data segment
    X DW 0010H,0052H,0030H,0040H,0050H
      MES DB 10,13,'LARGEST NUMBER AMONG THE SERIES IS $'
DATA ENDS                                                 ;end of data segment
CODE SEGMENT                                              ;start of code segment
    ASSUME CS:CODE,DS:DATA
    START:  MOV AX,DATA                              ;initialize data segment
            MOV DS,AX
            MOV CX,05H                                    ;load CX register with
                                                          number of datawords in array
            X LEA SI,X                                    ;SI points to start of dataword
                                                          array X
            MOV AX,[SI]                                   ;make a copy of the
                                                          first number in AX
            DEC CX                                        ;initialize CX with count
                                                          value for comparison
      UP:   CMP AX,[SI+2]                                 ;compare the contents of AX
                                                          and the number pointed by SI+2
            JA CONTINUE                              ;if AX is greater than the next
                                                          number in array then retain
                                                          the contents of AX
            MOV AX,[SI+2]                                 ;else make a copy of the next
                                                          number (larger number)in
                                                          AX
   CONTINUE:ADD SI,2                                      ;point to next number in array
            DEC CX                                        ;decrement CX
            JNZ UP                                   ;check if all numbers
            are
                                                          compared if no continue
                                                          comparison
            AAM                                           ;if yes convert largest binary
                                                          number in AX to unpacked BCD
            ADD AX,3030H                                  ;convert unpacked BCD to
                                                          unpacked ASCII equivalent
            MOV BX,AX                                     ;make a copy of it in AX
           MOV AX,09H                                ;display the message stored at
                                                          user defined memory location
                                                          MES
            LEA DX,MES
            INT 21H
            MOV DL,BH                                     ;display the largest number
            MOV
            AH,02H INT
            21H
            MOV DL,BL
            INT 21H
            MOV AH,4CH                               ;terminate the process
            INT 21H
CODE ENDS                                                 ;end of code segment
```

```
END START
```

OUTPUT: LARGEST NUMBER AMONG THE SERIES IS 0052

8)PROGRAM ON DAC WAVEFORM GENERATIONS:

ALP TO GENERATE A RECTANGULAR FREQUENCY OF 2KHz FREQUENCY

ADDRESS	INSTRUCTION	OPCODE	comment
2900	MOV AL,80	C6C080	move the control word for port a under mode0 operation into al register
2903	OUT 26,AL	E626	copy the contents into cr register port of address 26h
2905	MOV AL,OFFH	C6C0FF	copy 0 into al register i.e. low signal.
2908	OUT50,AL	E620	send this low signal to port a. i.e. address 20h
290a	CALL 3800	E8F30E	call a procedure to introduce some delay so that the Signal stays low for some time.
290D	MOV AL,00	F600	Now the signal is made high and kept in AL register.
290F	OUT 20,AL	E620	Send this high signal to port A. i.e. address 20H
2911	CALL 8500	E8EC5B	Calling the procedure to introduce some delay so that the signal stays high for some time
2914	JMP 2905	E9EEFF	The loop is infinite and rectangular wave is generated of Required frequency
2917	HLT	F4	Terminates the program
3800	MOV CX,002AH	C7C12A00	Move the number into CX register so that a rectangular wave of 2KHz frequency is generated
3804	NOP	90	Introduces some delay
3805	NOP	90	Introduces some delay
3806	LOOP 3804	E2FC	Loop executes and introduces delay
3808	RET	C3	Returns to the calling program
8500	MOV CX,002AH	C7C1200	Moves the number into CX register so that a rectangular wave is generated

8504	NOP	90	Introduces a delay
8505	NOP	90	Introduces a delay
8506	NOP	90	Introduces a delay
8507	NOP	90	Introduces a delay
8508	NOP	90	Introduces a delay
8509	LOOP 8504	E2F9	Loop executes and introduces delay
850B	RET	C3	Returns to the calling procedure

9)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:          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
UP:             CALL DELAY
                ROL AL,01H          ;rotate left to get exitation sequence of 11,22,44,88
                OUT DX,AL
                JMP UP
DELAY:          MOV CX,0FFFFH  ;delay can be adjusted to get different speeds
UP2:            MOV BX,0FFH
UP1:            DEC BX
                JNZ UP1
                DEC CX
                JNZ UP2
                RET
                MOV AH,4CH
                INT 21H
CODE ENDS
 END START
```

10)i)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
```

```
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:          MOV      BX,0FFFFH
L1:             MOV  CX,0FFFH  L2:
                DEC CX
                JNZ L2
                DEC BX
                JNZ L1
                RET

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

ii)SEVEN SEGMENT DISPLAY INTERFACE

```
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                ;display all the four digits
                JNZ NEXTDIGIT
                MOV AH,4CH            ;exit to dos
                INT 21H
      CODE ENDS END
      START
```

12) Programs on Data Transfer Instructions for 8051 Microcontroller:

Aim:

Write a 8051 ALP to copy a block of 10 bytes from RAM location starting at 37h to RAM location starting at 59h.

Program:

```
ORG 00H
MOV   R0,#37h             ; source pointer
MOV   R1,#59h             ; dest pointer
MOV   R2,#10              ; counter
L1: MOV   A,@R0
MOV   @R1,A
INC    R0
INC    R1
DJNZ  R2,L1
END
```

Output:

Before execution

R0 – 37H

05
04
03
02
01
05
04
03
02
01

After execution

R1 – 59H

05
04
03
02
00
05
04
03
02
01

13) Programs on Arithmetic and Logical Operations:

a) **ADDITION OF FIRST 10 NATURAL NUMBERS**

Aim: Write an 8051 ALP for addition of first 10 natural numbers

Program:

```
ORG 00H
MOV R0,#0AH
LOOP:ADDC A,R0
DJNZ R0,LOOP
MOV R1,A
END
```

Output:

```
R1: 37h
```

b) ADDITION OF TWO 16-BIT NUMBERS:

Aim: Write an 8051 ALP for addition of two 16-bit numbers

Program:

```
MOV A,R7        ;Move the low-byte into the accumulator
ADD A,R5         ;Add the second low-byte to the accumulator
MOV R3,A         ;Move the answer to the low-byte of the result
MOV A,R6           ;Move the high-byte into the accumulator
ADDC A,R4          ;Add the second high-byte to the accumulator, plus carry.
MOV R2,A            ;Move the answer to the high-byte of the result
MOV A,#00h          ;By default, the highest byte will be zero.
ADDC A,#00h          ;Add zero, plus carry from step 2.
MOV MOV R1,A     ;Move the answer to the highest byte of the result
```

Output:

answer now resides in R1, R2, and R3. RET

14) Programs on 8051 Applications:

Aim:

Write a 8051 ALP using Timer0 to create a 10khz square wave on P1.0

Program:

```
ORG 00H
MOV TMOD,#02H        ;8-bit auto-reload mode
MOV TH0,#-50         ;-50 reload value in TH0
SETB TR0             ;start timer0
LOOP:   JNB TF0, LOOP   ;wait for overflow
CLR TF0              ;clear timer0 overflow flag
CPL P1.0             ;toggle port bit
SJMP LOOP            ;repeat
END
```