DEPARTMENT OF
ELECTRONICS & COMMUNICATION ENGINEERING
EMBEDDED SYSTEMS LAB
MICROPROCESSORS & MICROCONTROLLERS LAB (IT)
III - B. Tech., I - Semester

PRASAD V POTLURI SIDDHARTHA INSTITUTE OF TECHNOLOGY
(Autonomous, Accredited by NBA & NAAC, an ISO 9001:2008 certified institution)
(Sponsored by Siddhartha Academy of General & Technical Education)
VIJAYAWADA – 520 007
ANDHRA PRADESH

MICROPROCESSORS & MICROCONTROLLERS LAB (IT)

LIST OF EXPERIMENTS

1. Introduction to Debugger / XT86 / TASM: 8-bit Arithmetic Operations
2. 16-bit Signed and unsigned Arithmetic operations, ASCII – arithmetic operations.
3. Arithmetic operations – Multi byte Addition and Subtraction, Sum of Squares, Sum of Cubes
4. Logic operations – Shift and rotate – Converting packed BCD to unpacked BCD, BCD to ASCII conversion.
5. 8255 – PPI: Write ALP to generate sinusoidal wave using PPI.
6. Using string operation and Instruction prefix: Move Block, Reverse string, String comparison
7. Write ALP to find smallest, largest number, arrange numbers in Ascending order, Descending order in a given series.
8. Traffic Lights Interface.
9. Stepper Motor Interface
10. 8279 – Keyboard Display: Write a small program to display a string of characters.
11. ADC Interface / DAC Interface.
12. Arithmetic Operations using 8051.
13. Reading and Writing on a parallel port.
14. Timer in Different Modes
15. Serial Communication using 8051.

1. EXPERIMENT
INTRODUCTION TO MASM /TASM

MASM: (Microsoft assembler)

To Create Source File: An editor is a program which allows you to create a file containing the assembly language statements for your program. This file is called a **source file.**
Command to create a source file

C:\MASM\BIN> Edit filename. asm

The next step is to process the source file with an assembler. When you run the assembler, it reads the source file of your program. On the first pass through the source program, the assembler determines the displacement of named data items, the offset labels, etc. and puts this information in a symbol table. On the second pass through the source program the assembler produces the binary code for each instruction and inserts the offsets, etc. that it calculated during first pass.

C:\MASM\BIN > Masm filename. asm X, Y, Z

With this command assembler generates three files.

1. The first file (X) called the object file, is given the extension .OBJ the object file contains the binary codes for the instructions and Information about the addresses of the instructions.
2. The second file (Y) generated by the assembler is called the assembler list file and is given the extension .LST. The list file contains your assembly language statements, the binary codes for each instruction and the offset for each instruction.
3. The third file (Z) generated by this assembler is called the cross-reference file and is given the extension .CRF. The cross-reference file lists all labels and pertinent information required for cross – referencing

NOTE: The Assembler only finds syntax errors: It will not tell you whether program does what it is supposed to do. To determine whether your program works, you have to run the program and test it.
Next step is to process the object file with linker.

C:\MASM\BIN>LINK filename. obj

Run File [Filename1.exe]: "filename1.exe"
Lists file [nul.map]: NUL
Libraries [.lib]: library_name
Definitions File [nul.def] :

Creation of Library: Refer Modular Programming Section

A Linker is a program used to join several object files into one layer object file
NOTE: On IBM PC – type Computers, You must run the LINK program on your .OBJ file even if it contains only one assembly module.
The linker produces a link file with the .EXE extension (an execution file)
Next Run **C:\MASM\BIN> filename**

TASM: (Turbo Assembler)

To Create Source File: An editor is a program which allows you to create a file containing the assembly language statements for your program. This file is called a **source file.**
Command to create a source file
C:\TASM\BIN> Edit filename. Asm

The next step is to process the source file with an assembler. When you run the assembler, it reads the source file of your program. On the first pass through the source program, the assembler determines the displacement of named data items, the offset labels, etc. and puts this information in a symbol table. On the second pass through the source program the assembler produces the binary code for each instruction and inserts the offsets, etc. that it calculated during first pass.
C:\TASM\BIN > TASM filename. asm X, Y, Z
With this command assembler generates three files.

1. The first file (X) called the object file, is given the extension .OBJ the object file contains the binary codes for the instructions and information about the addresses of the instructions.
2. The second file (Y) generated by the assembler is called the assembler list file and is given the extension .LST. The list file contains your assembly language statements, the binary codes for each instruction and the offset for each instruction.
3. The third file (Z) generated by this assembler is called the cross-reference file and is given the extension .CRF. The cross-reference file lists all labels and pertinent information required for cross – referencing

NOTE: The Assembler only finds syntax errors: It will not tell you whether program does what it is supposed to do. To determine whether your program works, you have to run the program and test it.
Next step is to process the object file with linker.
C:\TASM\BIN>TLINK filename. obj
A Linker is a program used to join several object files into one layer object file
NOTE: On IBM PC – type Computers, You must run the LINK program on your .OBJ file even if it contains only one assembly module.
The linker produces a link file with the .EXE extension (an execution file)
Next Run
C:\TASM\BIN> TD filename.exe

Assembly Language Program Format:
The assembler uses two basic formats for developing S/W
a) One method uses MODELS and
b) Other uses Full-Segment Definitions

* The models are easier to use for simple tasks.
* The full – segment definitions offer better control over the assembly language task and are recommended for complex programs.
a) Format using Models:
; ABSTRACT; 8086 program
; Aim of Program
; REGISTERS; Registers used in your program
; PORTS; PORTS used in your program
. MODEL (type of model i.e. size of memory system)

FOR EXAMPLE

```
. MODEL SMALL
. STACK size of stack; define stack
. DATA; define data segment
------
------Define variables
------
------
. CODE; define code segment s
HERE: MOV AX, @DATA; load ES, DS
MOV ES, AX
MOV DS, AX
---------
---------
---------
. EXIT 0; exit to DOS
END HERE
```

(or)

We can write Code segment as follows.

```
. CODE; Define Code Segment
. STARTUP
EXIT 0
END
```

2. EXPERIMENT

16-bit SIGNED, UNSIGNED AND ASCII ARITHMETIC OPERATIONS

AIM: To perform signed, unsigned and ASCII arithmetic operations using TASM software.

EQUIPMENT REQUIRED: PC loaded with TASM software.

PROCEDURE:

1. Go to start menu and click on run button.
2. Then a command window is opened, then type cmd in the text box and press ok.
3. D: //enter into D drive.
4. CD TASM
5. EDIT //window is opened to write source code in TASM environment.
6. Write program using respective commands and save the program with .ASM extension and quit.
7. D:\TASM>TASM filename.asm //to check errors.
8. tlink filename //to connect to executable files.
9. td filename //to debug the executable file and to see the **RESULT** of the operation.
10. Press F8 to get stepwise execution of the program or F9 to run program.
11. Required outputs are noted down.

UNSIGNED ARITHMETIC OPERATIONS

ADDITION:

```
ASSUME CS: CODE, DS: DATA
DATA   SEGMENT
OPR1 DB 0A2H
OPR2 DB 0A1H
RES DB 1 DUP (0H)
 DATA ENDS
CODE   SEGMENT
START:
MOV    AX, DATA
MOV    DS, AX
MOV    AL, OPR1
MOV     BL, OPR2
ADD      AL, BL
MOV     RES, AL
INT       03H
CODE   ENDS
END    START
END
```

RESULT: **INPUT:**
OUTPUT:

SUBTRACTION:

```
ASSUME CS: CODE, DS: DATA
DATA   SEGMENT
OPR1 DB 0A2H
OPR2 DB 0A1H
RES DB 1 DUP (0H)
 DATA ENDS
CODE   SEGMENT
START:
MOV    AX, DATA
MOV    DS, AX
MOV    AL, OPR1
MOV     BL, OPR2
SUB      AL, BL
MOV     RES, AL
INT       03H
CODE   ENDS
END    START
END
```

RESULT: **INPUT:**
OUTPUT:

MULTIPLICATION:

```
ASSUME CS: CODE, DS: DATA
DATA   SEGMENT
OPR1 DW 0A2H
OPR2 DW 0A1H
RES DW 1 DUP (0H)
 DATA ENDS
CODE   SEGMENT
START:
MOV    AX, DATA
MOV    DS, AX
MOV    AX, OPR1
MOV     BX, OPR2
MUL      BX
MOV     RES, AX
INT       03H
CODE   ENDS
END    START
END
```

RESULT: **INPUT:**
OUTPUT:

DIVISION:

```
ASSUME CS: CODE, DS: DATA
DATA   SEGMENT
OPR1 DW 0A0H
OPR2 DW 0A1H
RES DW 1 DUP (0H)
DATA ENDS
CODE   SEGMENT
START:
MOV    AX, DATA
MOV    DS, AX
MOV    AX, OPR1
MOV     BX, OPR2
DIV      BX
MOV     RES, AX
INT       03H
CODE   ENDS
END    START
END
```

RESULT: **INPUT;**
OUTPUT;

SIGNED ARITHMETIC OPERATIONS

ADDITION:

```
ASSUME CS: CODE, DS:DATA
DATA   SEGMENT
OPR1 DB 25H
OPR2 DB 37H
RES DB 1 DUP (0H)
 DATA ENDS
CODE   SEGMENT
START:
MOV    AX, DATA
MOV    DS, AX
MOV    AL, OPR1
MOV     BL, OPR2
ADC      AL, BL
MOV     RES, AL
INT       03H
CODE   ENDS
END    START
END
```

RESULT: **INPUT;**
OUTPUT

SUBTRACTION:

```
ASSUME CS: CODE, DS: DATA
DATA   SEGMENT
OPR1 DB 25H
OPR2 DB 37H
RES DB 1 DUP (0H)
 DATA ENDS
CODE   SEGMENT
START:
MOV    AX, DATA
MOV    DS, AX
MOV    AL, OPR1
MOV     BL, OPR2
SBB      AL, BL
MOV     RES, AL
INT       03H
CODE   ENDS
END    START
END
```

RESULT: INPUT:
OUTPUT

MULTIPLICATION :(+,+)

```
ASSUME CS: CODE, DS: DATA
DATA   SEGMENT
OPR1 DB 16H
OPR2 DB 5H
RES DB 1 DUP (0H)
 DATA ENDS
CODE   SEGMENT
START:
MOV    AX, DATA
MOV    DS, AX
MOV    AL, OPR1
MOV     BL, OPR2
IMUL     BL
MOV     RES, AL
INT       03H
CODE   ENDS
END    START
END
```

RESULT: INPUT:
OUTPUT:

MULTIPLICATION :(+,-)

```
ASSUME CS: CODE, DS: DATA
DATA   SEGMENT
OPR1 DB 16H
OPR2 DB 0FBH
RES DB 1 DUP (0H)
 DATA ENDS
CODE   SEGMENT
START: MOV    AX, DATA
MOV    DS, AX
MOV    AL, OPR1
MOV     BL, OPR2
IMUL      BL
MOV     RES, AL
INT       03H
CODE   ENDS
END    START
END
```

RESULT:

INPUT;

OUTPUT;

MULTIPLICATION :(-,+)

```
ASSUME CS: CODE, DS: DATA
DATA   SEGMENT
OPR1 DB 0EAH
OPR2 DB 5H
RES DB 1 DUP (0H)
 DATA ENDS
CODE   SEGMENT
START:
MOV    AX, DATA
MOV    DS, AX
MOV    AL, OPR1
MOV     BL, OPR2
IMUL      BL
MOV     RES, AL
INT       03H
CODE   ENDS
END    START
END
```

RESULT:

INPUT;

OUTPUT;

MULTIPLICATION :(-,-)

```
ASSUME CS: CODE, DS: DATA
DATA   SEGMENT
OPR1 DB 0EAH
OPR2 DB 0FBH
RES DB 1 DUP (0H)
DATA ENDS
CODE   SEGMENT
START: MOV    AX, DATA
MOV    DS, AX
MOV    AL, OPR1
MOV     BL, OPR2
IMUL      BL
MOV     RES, AL
INT       03H
CODE   ENDS
END    START
END
```

RESULT:

INPUT;

OUTPUT;

DIVISION :(+,+)

```
ASSUME CS: CODE, DS: DATA
DATA   SEGMENT
OPR1 DB 16H
OPR2 DB 5H
RE DB 1 DUP (0H)
QU DB 1 DUP (0H)
 DATA ENDS
CODE   SEGMENT
START: MOV    AX, DATA
MOV    DS, AX
MOV     AH, 00H
MOV    AL, OPR1
MOV     BL, OPR2
IDIV      BL
MOV     QU, AL
MOV     RE, AH
INT       03H
CODE   ENDS
END    START
END
```

RESULT:

INPUT;

OUTPUT;

DIVISION :(+,-)

```
ASSUME CS: CODE, DS: DATA
DATA   SEGMENT
OPR1 DB 16H
OPR2 DB 0FBH
RE DB 1 DUP (0H)
QU DB 1 DUP (0H)
DATA ENDS
CODE   SEGMENT
START: MOV    AX, DATA
MOV    DS, AX
MOV     AH, 00H
MOV    AL, OPR1
MOV     BL, OPR2
IDIV      BL
MOV     QU, AL
MOV     RE, AH
INT       03H
CODE   ENDS
END    START
END
```

RESULT:

INPUT;

OUTPUT;

DIVISION :(-,+)

```
ASSUME CS: CODE, DS: DATA
DATA   SEGMENT
OPR1 DB 0EAH
OPR2 DB 5H
RE DB 1 DUP (0H)
QU DB 1 DUP (0H)
DATA ENDS
CODE   SEGMENT
START: MOV    AX, DATA
MOV    DS, AX
MOV     AH, 00H
MOV    AL, OPR1
MOV     BL, OPR2
IDIV      BL
MOV     QU, AL
MOV     RE, AH
INT       03H
CODE   ENDS
END    START
END
```

RESULT:

INPUT;

OUTPUT;

DIVISION :(-,-)

```
ASSUME CS: CODE, DS: DATA
DATA   SEGMENT
OPR1 DB 0EAH
OPR2 DB 0FBH
RE DB 1 DUP (0H)
QU DB 1 DUP (0H)
 DATA ENDS
CODE   SEGMENT
START:
MOV    AX, DATA
MOV    DS, AX
MOV     AH, 00H
MOV    AL, OPR1
MOV     BL, OPR2
IDIV      BL
MOV     QU, AL
MOV     RE, AH
INT       03H
CODE   ENDS
END    START
END
```

RESULT:

INPUT;

OUTPUT;

ASCII ARITHMETIC OPERATIONS

ADDITION:

```
ASSUME CS:CODE
CODE   SEGMENT
START:
MOV   AX, 35H
MOV   BX, 39H
ADD    AX, BX
AAA
INT   03H
CODE   ENDS
END   START
END
```

RESULT: INPUT

OUTPUT-

SUBTRACTION:

```
ASSUME CS: CODE
CODE   SEGMENT
START:
MOV   AX, 37H
MOV   BX, 33H
SUB    AX, BX
AAS
INT   03H
CODE   ENDS
END   START
END
```

RESULT: INPUT-
OUTPUT-

MULTIPLICATION:

```
ASSUME CS:CODE
CODE   SEGMENT
START:
MOV   AL, 3H
MOV   BL, 7H
MUL    BL
AAM
INT   03H
CODE   ENDS
END   START
END
```

RESULT: INPUT-
OUTPUT-

DIVISION:

```
ASSUME CS: CODE
CODE   SEGMENT
START:
MOV   AX, 9H
MOV   BX, 5H
AAD
DIV    BL
INT   03H
CODE   ENDS
END   START
END
```

RESULT: INPUT-
OUTPUT-

RESULT: Hence signed, unsigned and ASCII arithmetic operations are performed using TASM software and required outputs are noted down.

3. EXPERIMENT
Arithmetic operations – Multi byte Addition and Subtraction, Sum of Squares, Sum of Cubes

AIM: To perform multi byte arithmetic operations using TASM software.

EQUIPMENT REQUIRED: PC loaded with TASM software.

PROCEDURE:

1. Go to start menu and click on run button.
2. Then a command window is opened, then type cmd in the text box and press ok.
3. D: //enter into D drive.
4. CD TASM
5. EDIT //window is opened to write source code in TASM environment.
6. Write program using respective commands and save the program with .ASM extension and quit.
7. D:\TASM>TASM filename.asm //to check errors.
8. tlink filename //to connect to executable files.
9. td filename //to debug the executable file and to see the **RESULT** of the operation.
10. Press F8 to get stepwise execution of the program or F9 to run program.
11. Go to view option and click on dump option to verify the output.
12. Required outputs are noted down.

PROGRAMS:

ADDITION

```
ASSUME CS: CODE, DS: DATA
DATA   SEGMENT
OPR1   DB   12H, 34H, 56H, 29H
OPR2   DB   32H, 04H, 76H, 21H
RES   DW   1   DUP (0H)
DATA    ENDS
CODE   SEGMENT
START:
MOV    AX, DATA
MOV    DS, AX
MOV    SI, OFFSET   OPR1
MOV    DI, OFFSET   OPR2
MOV    BX, OFFSET   RES
MOV    CX, 04H
BACK:
MOV     AL, [SI]
MOV     DL, [DI]
MOV      AH, 00H
ADC       AL, DL
MOV      [BX], AX
INC        SI
INC        DI
INC        BX
```

```
LOOP    BACK
INT    03H
CODE   ENDS
END   START
END
```

RESULT: INPUT-
OUTPUT-

SUBTRACTION

```
ASSUME CS: CODE, DS: DATA
DATA  SEGMENT
OPR1  DB  12H, 34H, 56H, 29H
OPR2  DB  32H, 04H, 76H, 21H
RES  DW  1  DUP (0H)
DATA   ENDS
CODE  SEGMENT
START:
MOV   AX, DATA
MOV   DS, AX
MOV   SI, OFFSET  OPR1
MOV   DI, OFFSET  OPR2
MOV   BX, OFFSET  RES
MOV   CX, 04H
BACK:
MOV    AL, [SI]
MOV    DL, [DI]
MOV    AH, 00H
ADC     AL, DL
MOV    [BX], AX
INC     SI
INC     DI
INC     BX
LOOP    BACK
INT    03H
CODE   ENDS
END   START
END
```


RESULT:
INPUT-
OUTPUT-

SUM OF SQUARES:

```
MOV CL, NUM
MOV SUM, 00
L1:     MOV AL, CL
MUL AL
ADD AL, SUM
MOV SUM, AL
LOOP L1
END
```

RESULT:
INPUT:
OUTPUT:

SUM OF CUBES

```
MOV CL, NUM
MOV SUM, 00
L1:     MOV AL, CL
MUL AL
MUL CL
ADD AL, SUM
MOV SUM, AL
LOOP L1
END
```

4. EXPERIMENT
Logic operations – Shift and rotate – Converting packed BCD to unpacked BCD, BCD to ASCII conversion

AIM: To perform

1. logical shift and rotate operations,
2. Conversion of packed BCD to unpacked BCD and BCD to ASCII using TASM.

EQUIPMENT REQUIRED loaded with TASM software.

PROCEDURE:

1. Go to start menu and click on run button.
2. Then a command window is opened, then type cmd in the text box and press ok.
3. D: //enter into D drive.
4. CD TASM
5. EDIT //window is opened to write source code in TASM environment.
6. Write program using respective commands and save the program with .ASM extension and quit.
7. D:\TASM>TASM filename.asm //to check errors.
8. tlink filename //to connect to executable files.
9. td filename //to debug the executable file and to see the **RESULT** of the operation.
10. Press F8 to get stepwise execution of the program or F9 to run program.
11. Required outputs are noted down.

PROGRAMS:

LOGICAL OPERATIONS: AND

```
ASSUME CS: CODE
CODE    SEGMENT
START:
MOV    AX, 3355H
MOV    BX, 5355H
AND    AX, BX
INT    03H
CODE    ENDS
END    START
END
```

RESULT: INPUT

OUTPUT-

OR:

```
ASSUME CS: CODE
CODE    SEGMENT
START:
MOV    AX, 3355H
MOV    BX, 5355H
OR       AX, BX
INT    03H
CODE    ENDS
END    START
END
```

RESULT: INPUT

OUTPUT-

NOT:

```
ASSUME CS: CODE
CODE    SEGMENT
START:
MOV    AX, 3355H
NOT    AX
INT    03H
CODE    ENDS
END    START
END
```

RESULT: INPUT
OUTPUT-

XOR:

```
ASSUME CS: CODE
CODE    SEGMENT
START:
MOV    AX, 3355H
MOV    BX, 5355H
XOR    AX, BX
INT    03H
CODE    ENDS
END    START
END
```

RESULT: INPUT

OUTPUT-

SHIFT OPERATIONS: RIGHT

```
ASSUME CS: CODE
CODE    SEGMENT
START:
MOV    AX, 5352H
MOV    CL, 01H
SHR    AX, CL
 INT    03H
CODE    ENDS
END    START
END
```

RESULT: INPUT

OUTPUT-

LEFT:

```
ASSUME    CS: CODE
CODE    SEGMENT
START:
MOV    AX, 5352H
MOV    CL, 02H
SHL    AX, CL
INT    03H
CODE    ENDS
END    START
END
```


RESULT: INPUT

OUTPUT-

ROTATE RIGHT:

```
ASSUME    CS: CODE
CODE    SEGMENT
START:
MOV    AX, 8351H
MOV    CL, 01H
ROR    AX, CL
INT    03H
CODE    ENDS
END    START
END
```

RESULT: INPUT

OUTPUT-

LEFT:

```
ASSUME   CS: CODE
CODE   SEGMENT
START:
MOV   AX, 8351H
MOV   CL, 03H
ROL   AX, CL
INT   03H
CODE   ENDS
END   START
END
```

RESULT: INPUT
OUTPUT-

PACKED BCD TO UNPACKED BCD

```
ASSUME   CS:  CODE
CODE   SEGMENT
START:
MOV   AL, 56H
MOV   AH, AL
SHR   AH, 04H
AND   AX, 0F0FH
INT   03H
CODE   ENDS
END   START
END
```


RESULT: INPUT
OUTPUT-

BCD TO ASCII

```
ASSUME   CS: CODE
CODE   SEGMENT
START:
MOV   AL, 56H
MOV   AH, AL
SHR   AH, 04H
AND   AX, 0F0FH
OR   AX, 3030H
INT   03H
CODE   ENDS
END   START
END
```

RESULT: INPUT
OUTPUT-

5. EXPERIMENT
SINE WAVE GENERATION USING 8255

AIM: To generate a sine wave using 8255.

EQUIPMENT REQUIRED: PC loaded with TASM software.

PROCEDURE:

1. First click on start and then select run and type cmd then ok.
2. Type D:// Enter into D Drive.
3. Type cd esa// Enter into ESA
4. Type XT86.
5. In the processor kit, reset the processor by pressing the reset button.
6. Now enter. A, next. DA address, Enter the program in the window.
7. The program is to be entered a space before every instruction.
8. For the execution shift+1 (or)! is entered.
9. Then type Ex and Enter Key.
10. After that type G.
11. Then enter the starting address.
12. Then the sinewave is displayed on the CRO.
13. Then observe the wave form and take readings like amplitude and frequency.

PROGRAM:

```
MOVB AL, #80
MOVW DX, #0FFE7
OUTB DX
L7 MOVB AL, #00
L1 MOVW DX, #0FFE1
    OUTB DX
    INCB AL
   CMPB AL, #0F
   JB L1
L2 MOVW DX, #0FFE1
    OUTB DX
    INCB AL
    INCB AL
   INCB AL
   CMPB AL, #0EF
   JB L2
L3 MOVW DX, #0FFE1
    OUTB DX
    INCB AL
   CMPB AL, #0FF
   JB L3
L4 MOVW DX, #0FFE1
    OUTB DX
    DECB AL
   CMPB AL, #0EF
```

```
    JA L4
L5 MOVW DX, #0FFE1
     OUTB DX
     DECB AL
     DECB AL
    DECB AL
    CMPB AL, #0F
    JA L5
L6 MOVW DX, #0FFE1
     OUTB DX
     DECB AL
    CMPB AL, #00
    JA L6
JMP L7
INT 03!
```

RESULT:

Sine Wave.

6. EXPERIMENT
STRING OPERATIONS-1

AIM: To perform string operations like move block, reverse string, sorting using TASM software.

EQUIPMENT REQUIRED: PC loaded with TASM software.

PROCEDURE:

1. Go to start menu and click on run button.
2. Then a command window is opened, then type cmd in the text box and press ok.
3. D: //enter into D drive.
4. CD TASM
5. EDIT //window is opened to write source code in TASM environment.
6. Write program using respective commands and save the program with .ASM extension and quit.
7. D:\TASM>TASM filename.asm //to check errors.
8. tlink filename //to connect to executable files.
9. td filename //to debug the executable file and to see the **RESULT** of the operation.
10. Press F8 to get stepwise execution of the program or F9 to run program.
11. Go to view option and click on dump option to verify the output.
12. Required outputs are noted down.

PROGRAMS:

MOVING BLOCK OF STRING:

```
ASSUME    CS: CODE, DS: DATA, ES: EXTRA
DATA    SEGMENT
ORG    1000H
STR1    DB    'HIFRIEND'
DATA    ENDS
EXTRA    SEGMENT
ORG    2000H
STR2    DB    1    DUP (0H)
EXTRA    ENDS
CODE    SEGMENT:
START:
MOV    AX, DATA
MOV    DS, AX
MOV    AX, EXTRA
MOV    ES, AX
MOV    SI, OFFSET    STR1
MOV    DI, OFFSET    STR2
 MOV    CL, 0AH
CLD
REP    MOVSB
INT    03H
CODE    ENDS
```

```
END   START
END
```

RESULT: INPUT
OUTPUT

REVERSE OF A STRING:

```
ASSUME   CS: CODE, DS: DATA
DATA   SEGMENT
ORG    1000H
STR1   DB   'HI FRIEND'
DATA ENDS
CODE   SEGMENT
START:
MOV   AX, DATA
MOV   DS, AX
MOV   SI, OFFSET   STR1
MOV   DI, 00BH
MOV   CL, 04H
BACK:
MOV   AL, [SI]
XCHG   [DI], AL
XCHG   [SI], AL
INC      SI
DEC      DI
LOOP    BACK
INT       03H
CODE    ENDS
END     START
END
```


RESULT: INPUT
OUTPUT

COMPARISON OF STRING:

```
ASSUME   CS: CODE, DS: DATA, ES: EXTRA
DATA   SEGMENT
ORG   1000H
STR1   DB   'HI FRIEND'
DATA   ENDS
EXTRA   SEGMENT
ORG   2000H
STR2   DB   'HIFRIEND'
EXTRA   ENDS
CODE   SEGMENT:
START:
MOV   AX, DATA
MOV   DS, AX
MOV   AX, EXTRA
MOV   ES, AX
MOV   SI, OFFSET   STR1
MOV   DI, 0AH
 MOV   CL, 0AH
REP   CMPSB
INT   03H
CODE   ENDS
END   START
END
```

RESULT: INPUT

OUTPUT

7. EXPERIMENT
Smallest, largest number, arrange numbers in Ascending order, Descending order

AIM: Using string operations to perform smallest, largest number, arrange numbers in ascending order, descending order in a given series.

EQUIPMENT REQUIRED: PC loaded with TASM software.

PROCEDURE:

1. Go to start menu and click on run button.
2. Then a command window is opened, then type cmd in the text box and press ok.
3. D: //enter into D drive.
4. CD TASM
5. EDIT //window is opened to write source code in TASM environment.
6. Write program using respective commands and save the program with .ASM extension and quit.
7. D:\TASM>TASM filename.asm //to check errors.
8. tlink filename //to connect to executable files.
9. td filename //to debug the executable file and to see the **RESULT** of the operation.
10. Press F8 to get stepwise execution of the program or F9 to run program.
11. Go to view option and click on dump option to verify the output.
12. Required outputs are noted down.

PROGRAMS:

SMALLEST NUMBER FROM GIVEN LIST:

```
ASSUME   CS: CODE, DS: DATA
DATA   SEGMENT
LIST   DB   05H, 19H, 26H, 56H, 44H
RES   DB   1   DUP (0H)
DATA   ENDS
CODE   SEGMENT
START:
MOV   AX, DATA
MOV    DS, AX
MOV    CX, 0004H
MOV   BL, [SI]
MOV   SI, OFFSET   LIST
L2:
MOV   AL, [SI+1]
CMP    BL, AL
JB   L1
MOV   BL, AL
L1:
INC    SI
LOOP    L2
MOV   RES, BL
```

```
INT      03H
CODE   ENDS
END    START
END
```

RESULT: INPUT
OUTPUT

LARGEST NUMBER FROM GIVEN LIST:

```
ASSUME   CS: CODE, DS: DATA
DATA   SEGMENT
LIST   DB   05H, 19H, 26H, 56H, 44H
RES   DB   1   DUP(0H)
DATA   ENDS
CODE   SEGMENT
START:
MOV   AX, DATA
MOV    DS, AX
MOV    CX, 0004H
MOV   BL, [SI]
MOV   SI, OFFSET   LIST
L2:
MOV   AL, [SI+1]
CMP    BL, AL
JA   L1
MOV   BL, AL
L1:
INC    SI
LOOP    L2
MOV   RES, BL
INT      03H
CODE   ENDS
END    START
END
```


RESULT: INPUT
OUTPUT

ASCENDING ORDER:

```
ASSUME   CS: CODE, DS: DATA
DATA   SEGMENT
STR   DB   'BINDHU'
DATA   ENDS
CODE   SEGMENT
START: MOV   AX, DATA
MOV    DS, AX
MOV    DX, 0005H
L3:
MOV   CX, DX
MOV   SI, OFFSET   STR
```

```
L2:
MOV   AL, [SI]
CMP   AL, [SI+1]
JB   L1
XCHG   AL, [SI+1]
XCHG   AL, [SI]
L1:
INC   SI
LOOP   L2
DEC   DX
JNZ   L3
INT   03H
CODE   ENDS
END   START
END
```

RESULT: INPUT

OUTPUT

DESCENDING ORDER:

```
ASSUME   CS: CODE, DS: DATA
DATA   SEGMENT
STR   DB   'BINDHU'
DATA   ENDS
CODE   SEGMENT
START: MOV   AX, DATA
MOV    DS, AX
MOV    DX, 0005H
L3:
MOV   CX, DX
MOV   SI, OFFSET   STR
L2:
MOV   AL, [SI]
CMP   AL, [SI+1]
JA   L1
XCHG   AL, [SI+1]
XCHG   AL, [SI]
L1:
INC   SI
LOOP   L2
DEC   DX
JNZ   L3
INT   03H
CODE   ENDS
END   START
END
```

RESULT: INPUT

OUTPUT

8. EXPERIMENT
TRAFFIC LIGHTS INTERFACING

AIM:

To study interfacing technique of Traffic Lights Interface with microprocessor 8086and write an 8086 ALP.

Apparatus: 8086 kit-1No, Traffic Light Interface Module.

Procedure:

Program:

```
ORG 2000H
MOV B AL,#08H
MOVW DX,#0FFE7H
OUTB DX,AL
MOVW CX,#0005
MOVW SI,#2006F
MOVB AL,[SI]
MOVW DX,#0FFE1H
OUTB DX,AL
INC SI
ADDW DX,#0002
MOVB AL,[SI]
OUTB DX,AL
INCW SI
ADDW DX,#0002
MOVB AL,[SI]
OUTB DX,AL
INCW SI
PUSH SI
PUSH CX
MOVW DX,#OFFEDH
INB AL,DX
TESTB AL,#08H
JZ
NOP
DB 9AH,70H,1BH
DB 00H,0FEH
DW AL,2C3C
JNZ 202B
JZ 2042
NOP
DB 9AH,1CH,0BH
DB 00H,0FFH
CMPB AL,#11H
JNZ 2038
POP CX
POP SI
```

```
MOVB AL,[SI]
MOVW DX,#0FFE1H
OUTB DX,#AL
INCW SI
ADDW DX,#0002
MOVB AL,[SI]
OUTB DX,AL
INCW SI
ADDW DX,#0002H
MOVB AL,[SI]
OUTB DX,AL
INCW SI
CALL 205F
LOOP 200C
JMP 2006
MOVB BL,#0FH
PUSH CX
MOVW CX,#1FFFH
NOP
LOOP 2066
DECB BL
JNZ 2063
POP CX
RET
DB 88H,83H,0F2H
DB 88H,87H,0F2H
DB 38H,88H,0F4H
DB 78H,88H,0F4H
DB 83H,88H,0F4H
DB 87H,88H,0F8H
DB 88H,38H,0F1H
DB 88H,38H,0F1H
DB 88H,88H,00H
DB 88H,88H,00H
```


RESULT:

9. EXPERIMENT
STEPPER MOTOR INTERFACE

AIM: Write an assembly language program for stepper motor interface with 8086.

Apparatus: 8086 Microprocessor with power supply
Stepper motor inter face

Procedure:

1. Go to start menu and click on RUN and it will be opened.
2. Enter address D:/ESA/XT86.EXE
3. Press enter to continue.
4. Reset microprocessor kit.
5. To enter into assemble mode type A and press "ENTER"
6. Initialize segment register to 0000 i.e sb 00
7. Clear label using command "LC"
8. Direct address (DA) to a location using DA command.

Program:

```
MOVB AL,#80
MOVW DX,#OFFH
OUTB DX
MOVB AL,#88
MOVW DX,#EFH
OUTB DX
CALL 2014
XCHGW AX,AX
RORL AL,2
JMP 2008
PUSHF
PUSH AX
MOVW BX,#03H
DECW BX
JNE 2019
POP AX
POPF
RET
```

Result:
Therefore the stepper motor interface is performed.

10. EXPERIMENT
8279 – KEYBOARD DISPLAY

AIM: To display a string of characters using 8279 Keyboard Display.

EQUIPMENT REQUIRED: PC loaded with TASM software.

PROCEDURE:

1. First click on start and then select run and type cmd then ok.
2. Type D:// Enter into D Drive.
3. Type cd esa// Enter into ESA
4. Type XT86.
5. In the processor kit, reset the processor by pressing the reset button.
6. Now enter. A, next. DA address, Enter the program in the window.
7. The program is to be entered a space before every instruction.
8. Then type CX and then shift +1,Ex enter key D6,93,67,F3,F3,83
   G 5000(Address).
9. Then on the Display board the output is displayed as per the Hexadecimal input
   SCHOOL.

PROGRAM

```
MOVB AL, #90
MOVW DX, #82
OUTB DX
MOVB AL, #00
OUTB DX
MOVW CX, #08
RPT: MOVB AL, #00
      MOVW DX, #80
     OUTB DX
     LOOP 200C
      MOVW CX, #06
     MOVW SI, #2100
LOOP: MOVB AL, [SI]
       OUTB DX
      INCW [SI]
       LOOP 201A
       INT 03
```

RESULT:

SCHOOL

11. EXPERIMENT
ADC Interface / DAC Interface

AIM: To generate a square wave using 8255.

EQUIPMENT REQUIRED: PC loaded with TASM software.

PROCEDURE:

1. First click on start and then select run and type cmd then ok.
2. Type D:// Enter into D Drive.
3. Type cd esa// Enter into ESA
4. Type XT86.
5. In the processor kit, reset the processor by pressing the reset button.
6. Now enter. A, next. DA address, Enter the program in the window.
7. The program is to be entered a space before every instruction.
8. For the execution shift+1 (or)! is entered.
9. Then type Ex and Enter Key.
10. After that type G.
11. Then enter the starting address.
12. Then the squarewave is displayed on the CRO.
13. Then observe the wave form and take readings like amplitude and frequency.

PROGRAM:

```
MOVB AL, #80
MOVW DX, #0FFE7
OUTB DX
L3 MOVB AL, #00
MOVW DX, #0FFE0
MOVW CX, #00FF
L1 OUTB DX
   LOOP L1
   MOVB AL, #0A
   MOVW CX, #00FF
L2 OUTB DX
   LOOP L2
   LOOP L3
INT 03!
```

RESULT:
Square Wave.

12. EXPERIMENT
ARITHMETIC OPERATIONS USING 8051

AIM: To write an assembly program

EQUIPMENT REQUIRED: PC loaded with TASM software.

PROCEDURE:

1. First click on start and then select run and type cmd then ok.
2. Type D:// Enter into D Drive.
3. Type cd esa// Enter into ESA
4. Type XT86.
5. Type >A 8000, here 8000 is the memory in which you want to write the data.
6. Write the program and then to see the total program press Z then SA and EA memory locations
7. To execute SR enter the starting address and press enter .
8. Then you can see the output.

PROGRAM

ADDITION:

```
ORG 8000H
MOV DPTR, #9000H
MOVX A,@DPTR
MOV 0F0H,A
MOV DPTR,#9001H
MOVX A, @DPTR
ADD A,0F0H
MOV DPTR,#9002H
MOVX @DPTR,A
LJMP 0
```


RESULT:
INPUT:
OUTPUT:

SUBTRACTION:

```
ORG 8000H
MOV DPTR, #9001H
MOVX A,@DPTR
MOV 0F0H,A
MOV DPTR,#9000H
MOVX A, @DPTR
SUB A,0F0H
MOV DPTR,#9002H
MOVX @DPTR,A
LJMP 0
```

MULTIPLICATION:

```
ORG 8000H
MOV DPTR, #9001H
MOVX A,@DPTR
MOV 0F0,A
MOV DPTR,#9000H
MOVX A, @DPTR
MUL AB
MOV DPTR,#9002H
MOVX @DPTR,A
LJMP 0
```

RESULT:
INPUT:
OUTPUT:

DIVISION:

```
ORG 8000H
MOV DPTR, #9001H
MOVX A,@DPTR
MOV 0F0,A
MOV DPTR,#9000H
MOVX A, @DPTR
DIV AB
MOV DPTR,#9002H
MOVX @DPTR,A
LJMP 0
```

RESULT:
INPUT:
OUTPUT:

13. EXPERIMENT
READING AND WRITING IN A PARALLEL PORT

AIM: To write an assembly program

EQUIPMENT REQUIRED: PC loaded with TASM software.

PROCEDURE:

1. First click on start and then select run and type cmd then ok.
2. Type D:// Enter into D Drive.
3. Type cd esa// Enter into ESA
4. Type XT86.
5. Type >A 8000, here 8000 is the memory in which you want to write the data.
6. Write the program and then to see the total program press Z then SA and EA memory locations
7. To execute SR enter the starting address and press enter .
8. Then you can see the output.

PROGRAM

```
MOV A, #20
MOV 90, A
MOV R1,90
MOV A, #00
SJMP 8008
```

RESULT:

OUTPUT

```
A       B     SP    PSW   DPH    DPL    TH0   TL0   TH1   TL1   P1  P3  PCH  PCL
<E0>  <F0>  <81>  <D0>  <83>   <82>   <8C>  <8A>  <8D>  <8B>
00      00    07     01     00     00     00    00     00    20   FB   80    08
R0  R1  R2  R3  R4  R5  R6  R7    PSW
00  20  1B 99  DF  6F  B3  20    00000007
```

14. EXPERIMENT
TIMER IN DIFFERENT MODES

AIM: - To perform the operation of timer in different modes using 8051.

EQUIPMENT REQUIRED: PC loaded with TASM software.

PROCEDURE:

1. First click on start and then select run and type cmd then ok.
2. Type D:// Enter into D Drive.
3. Type cd esa// Enter into ESA
4. Type XT86.
5. Type >A 8000, here 8000 is the memory in which you want to write the data.
6. Write the program and then to see the total program press Z then SA and EA memory locations
7. To execute SR enter the starting address and press enter .
8. Then you can see the output.

PROGRAM: -

ADDRESS	OPCODE	LABEL	MNEMONIC	OPERAND
			MOV	89, #01
		WAIT :	MOV	8A,#0F2
			MOV	8C,#0FF
			CPL	95
			ACALL	DELAY
			SJMP	WAIT
		DELAY :	SETB	8C
		HERE :	JNB	8D, HERE
			CLR	8C
			CLR	8D
			RET	

RESULT: -

Input:

Output:

15. EXPERIMENT
SERIAL COMMUNICATION USING 8051

AIM: - To perform serial communication between the Master and Slave microprocessor using 8051 micro controller.

APPARATUS:

8051 Micro controller
Key board
Power supply

THEORY: -

The serial port is full duplex, meaning it can transmit and receive simultaneously. It is also receive buffered, meaning it can commence reception of second byte before a previously received byte has been read from the receive register. The serial port receive and transmit registers are both accessed at special function register SBUF to SBUF accesses a physically separate receive register. However, if the first byte still has not been by the time reception of the second byte is complete, one of the bytes will be last.

OPERATING MODES FOR SERIAL PORT: -

MODE 0: Serial data enters and exists through RXD, TXD outputs the shift clock. 8 bits are received/transmitted. The baud rate is fixed at 1/12 the oscillator frequency.
MODE 1: 10 bits are transmitted (through TXD) or received ((through RXD): a start bit (0), 8 (LSB first) bits and a stop bit (1). The baud rate is variable.
MODE2: 11 bits are transmitted (through TXD) or received (through RXD): a start bit (0), 8 bits (LSB first), a 9^{th} data bit and a stop bit (1).

PROGRAMMING 8051 FOR SERIAL DATA TRANSFER: -

1. Clear T_1 with CLR T_1 instruction.
2. Write a character to be sent into SBUF register
3. Check the T_1 flag (register) bit with instruction JNB T_1, XXXX to see if the character has been transferred.
4. Go to step 1 to transfer the next character. The baud rate is 1/32 (or) 1/64 the oscillator frequency.

PROGRAMMING 8051 FOR RECEIVING SERIAL DATA: -

1. Clear RI to CLR RI instruction.
2. Check the RI flag bit with instruction JNB, RI, XXXX to see if an entire character has been transferred.
3. If R1 to see, SBUF has the byte save this byte.
4. Go to step 1 to receive the next character.

PROCEDURE: -

Connect the master and slave in series and connect them to system. Press EXE MEM, PROG MEM and then type the starting address of program (i. E. 8000) in master. Press EXE MEM, PROG MEM and then 8100 in slave. Press EXE MEM, external data and then enter the data at 9200 in master. Execute the slave first and then master {Press GO

8100 execute in slave and GO 8000 execute in master}. Then check contents of O/P register for received data in slave.

PROGRAM FOR MASTER OPERATION: -

ADDRESS	OPCODE	LABEL	MNEMONIC	OPERAND
8000			LCALL	160E
			MOV	DPTR, #9200
			MOVX	A, @DPTR
			LCALL	160E
			NOP	
			NOP	
			NOP	
			NOP	
			INC	DPTR
			MOVX	A, @DPTR
			LCALL	160E
		HERE:	SJMP	HERE

PROGRAM FOR SLAVE OPERATION: -

ADDRESS	OPCODE	LABEL	MNEMONIC	OPERAND
8100			LCALL	16E2
			MOV	DPTR, #9200
			LCALL	16E2
			MOVX	@DPTR, A
			INC	DPTR
			LCALL	16E2
			MOVX	@DPTR, A
		HERE:	SJMP	HERE

Result: -

Input: -

Output: -