MP& MC LAB MANUAL

Department of Electronics & Communication Engineering

VEMU INSTITUTE OF TECHNOLOGY::P.KOTHAKOTA

NEAR PAKALA, CHITTOOR-517112

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapuramu)

MICROPROCESSORS AND MICROCONTROLLERS

LAB OBSERVATION

Name:_____

H.T.No:_____

Year/Semester:_____

Department of Electronics & Communication Engineering

VEMU INSTITUTE OF TECHNOLOGY::P.KOTHAKOTA

NEAR PAKALA, CHITTOOR-517112

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapuramu)

VEMU Institute of Technology
Dept. of Electronics and Communication Engineering

Vision of the institute

To be one of the premier institutes for professional education producing dynamic and vibrant force of technocrats with competent skills, innovative ideas and leadership qualities to serve the society with ethical and benevolent approach.

Mission of the institute

Mission_1: To create a learning environment with state-of-the art infrastructure, well equipped laboratories, research facilities and qualified senior faculty to impart high quality technical education.

Mission_2: To facilitate the learners to inculcate competent research skills and innovative ideas by Industry-Institute Interaction.

Mission_3: To develop hard work, honesty, leadership qualities and sense of direction in learners by providing value based education.

Vision of the department

To develop as a center of excellence in the Electronics and Communication Engineering field and produce graduates with Technical Skills, Competency, Quality, and Professional Ethics to meet the challenges of the Industry and evolving Society.

Mission of the department

Mission_1: To enrich Technical Skills of students through Effective Teaching and Learning practices to exchange ideas and dissemination of knowledge.

Mission_2: To enable students to develop skill sets through adequate facilities, training on core and multidisciplinary technologies and Competency Enhancement Programs.

Mission_3: To provide training, instill creative thinking and research attitude to the students through Industry-Institute Interaction along with Professional Ethics and values.

Programme Educational Objectives (PEOs)

PEO 1: To prepare the graduates to be able to plan, analyze and provide innovative ideas to investigate complex engineering problems of industry in the field of Electronics and Communication Engineering using contemporary design and simulation tools.

PEO-2: To provide students with solid fundamentals in core and multidisciplinary domain for successful implementation of engineering products and also to pursue higher studies.

PEO-3: To inculcate learners with professional and ethical attitude, effective communication skills, teamwork skills, and an ability to relate engineering issues to broader social context at work place

Programme Outcomes (Pos)

PO_1	**Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO_2	**Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO_3	**Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO_4	**Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO_5	**Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO_6	**The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO_7	**Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO_8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO_9	**Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO_10	**Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO_11	**Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO_12	**Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Programme Specific Outcome(PSOs)

PSO_1	**Higher Education :** Qualify in competitive examination for pursuing higher education by applying the fundamental concepts of Electronics and Communication Engineering domains such as Analog & Digital Electronics, Signal Processing, Communication & Networking, Embedded Systems, VLSI Design and Control systems etc.,
PSO_2	**Employment**: Get employed in allied industries through their proficiency in program specific domain knowledge, Specialized software packages and Computer programming or became an entrepreneur.

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR

III B.Tech. I- SEM (ECE)

(20A04503P) MICROPROCESSORS AND MICROCONTROLLERS LAB

Course Outcomes:

CO. No	Description	Bloom Level
C317.1	Formulate problems and implement algorithms using 8086 and 8051 Assembly language	06
C317.2	Develop 8086 and 8051 programs for different applications	03
C317.3	Interface peripheral devices with 8086 and 8051.	04
C317.4	Apply Assembly/Embedded C programming approach for solving real world problems.	03

List of Experiments:

1. **PROGRAMS FOR 16 BIT ARITHMETIC OPERATIONS (Using various addressing modes)**
   a) Write an ALP to Perform Addition and Subtraction of Multi precision numbers.
   b) Write an ALP to Perform Multiplication and division of signed and unsigned Hexadecimal numbers.
   c) Write an ALP to find square, cube and factorial of a given number.
2. **PROGRAMS INVOLVING BIT MANIPULATION INSTRUCTIONS**
   a) Write an ALP to find the given data is positive or negative.
   b) Write an ALP to find the given data is odd or even.
   c) Write an ALP to find Logical ones **and zeros in a given data.**
3. **PROGRAMS** ON ARRAYS FOR 8086
   a) Write an ALP to find Addition/subtraction of N no‘s.
   b) Write an ALP for finding largest/smallest no.
   c) Write an ALP to sort given array in Ascending/descending order.
4. **PROGRAM FOR STRING MANIPULATIONS FOR 8086**
   a) Write an ALP to find String length.
   b) Write an ALP for Displaying the given String.
   c) Write an ALP for Comparing two Strings.
   d) Write an ALP to reverse String and Checking for palindrome.
5. **PROGRAM FOR DIGITAL CLOCK DESIGN USING 8086**
   a) Write an ALP for Designing clock using INT 21H Interrupt.
   b) Write an ALP for Designing clock using DOS Interrupt Functions.
   c) Write an ALP for Designing clock by reading system time.
6. **INTERFACING STEPPER MOTOR WITH 8086**
   a) Write an ALP to 8086 processor to Interface a stepper motor and operate it in clockwise by choosing variable step-size.
   b) Write an ALP to 8086 processor to Interface a stepper motor and operate it in Anti-clockwise by choosing variable step-size.

7. INTERFACING ADC/DAC WITH 8086

a) Write an ALP to 8086 processor to Interface ADC.
b) Write an ALP to 8086 processor to Interface DAC and generate Square Wave/Triangular Wave/Stepsignal.

8. COMMUNICATION BETWEEN TWO MICROPROCESSORS

a) Write an ALP to have Parallel communication between two microprocessors using 8255
b) Write an ALP to have Serial communication between two microprocessor kits using 8251.

9. PROGRAMS USING ARITHMETIC AND LOGICAL INSTRUCTIONS FOR 8051

a) Write an ALP to 8051 Microcontroller to perform Arithmetic operations like addition, subtraction,
b) Multiplication and Division.
c) Write an ALP to 8051 Microcontroller to perform Logical operations like AND, OR and XOR.
d) Programs related to Register Banks.

10. PROGRAM TO VERIFY TIMERS/COUNTERS OF 8051

a) Write a program to create a delay of 25msec using Timer0 in mode 1 and blink all the Pins of P0.
b) Write a program to create a delay of 50 µsec using Timer1 in mode 0 and blink all the Pins of P2.
c) Write a program to create a delay of 75msec using counter0 in mode 2 and blink all the Pins of P1.
d) Write a program to create a delay of 80 µsec using counter1 in mode 1 and blink all the Pins of P3.

11. UART OPERATION IN 8051

a) Write a program to transfer a character serially with a baud rate of 9600 using UART.
b) Write a program to transfer a character serially with a baud rate of 4800 using UART.
c) Write a program to transfer a character serially with a baud rate of 2400 using UART.

12. INTERFACING LCD WITH 8051

a) Develop and execute the program to interface16*2 LCD to 8051.
b) Develop and execute the program to interface LCD to 8051 in 4-bit or 8-bit mode.

Reference Books:

1. Kenneth.J.Ayala. The 8051 microcontroller, 3rd edition, Cengage learning,2010.
2. Advanced microprocessors and peripherals-A.K ray and K.M.Bhurchandani, TMH, 2nd edition2006.
3. The 8051 Microcontroller and Embedded Systems: Using Assembly and C by Muhammad AliMazidi, Janice GillispieMazidi, Second Edition.

Note: Any TEN of the experiments are to be conducted.

VEMU INSTITUTE OF TECHNOLOGY::P.KOTHAKOTA

NEAR PAKALA, CHITTOOR-517112
(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapur)
Department of Electronics &Communication Engineering

(20A04503P) MICROPROCESSORS AND MICROCONTROLLERS LAB

LIST OF EXPERIMENTS TO BE CONDUCTED

1. **PROGRAMS FOR 16 BIT ARITHMETIC OPERATIONS**
   (a)To perform Addition and Subtraction of multi precision numbers
   (b)To perform Multiplication and Division of signed and unsigned numbers
   (c) To find Square, Cube and factorial of a given numbers.

2. **PROGRAMS INVOLVING BIT MANIPULATION INSTRUCTIONS**
   (a) To find the given date is positive or negative
   (b) To find the given date is positive or negative
   (c) To find the given date is logical ones or zeros.

3. **PROGRAMS ON ARRAYS FOR 8086**
   (a)To find addition /subtraction of N-numbers
   (b)To find largest/smallest numbers
   (c) To sort given array in ascending/descending

4. **PROGRAM FOR STRING MANIPULATIONS FOR 8086**
   (a)To find string length
   (b)To displaying given string
   (c)Comparing of two string
   (d)To reverse string and checking for palindrome

5. **PROGRAM FOR DIGITAL CLOCK DESIGN USING 8086**
   (a) Digital clock design using INT 21H, DOS Interrupts functions
   (b) Designing clock by reading system time

6 .**INTERFACING STEPPER MOTOR WITH 8086**
   (a)To Interface Stepper Motor with 8086 by using 8255 to rotate Stepper Motor in Clockwise directions
   (b)To Interface Stepper Motor with 8086 by using 8255 to rotate Stepper Motor in Anti-clock wise directions

7. **INTERFACING ADC/DAC WITH 8086**
   (a) To Interface ADC
   (b) To Interface DAC

8. **COMMUNICATION BETWEEN TWO MICROPROCESSORS**
   (a) Parallel communication between two microprocessors kits by using 8255.
   (b) Serial communication between two microprocessors kits by using 8251.

9. PROGRAMS USING ARITHMETIC AND LOGICAL INSTRUCTIONS FOR 8051

(a) Arithmetic operations
(b) Logical operations

10. PROGRAM TO VERIFY TIMERS/COUNTERS OF 8051

(a) To Perform Timer 0 and Timer 1 in Mode 0.
(b) To Perform Timer 0 and mode 1 in Mode 1.

ADVANCED EXPERIMENTS (BEYOND CURRICULUM)

11. To find LCM for the given data using 8086.

12. Interfacing 8279 keyboard with 8051.

.

VEMU INSTITUTE OF TECHNOLOGY

P.KOTHAKOTA, NEAR PAKALA, CHITTOOR, AP

Department of Electronics &Communication Engineering

CONTENTS

6.	**INTERFACING STEPPER MOTOR WITH 8086** (a)To Interface Stepper Motor with 8086 by using 8255 to rotate Stepper Motor in Clockwise directions (b)To Interface Stepper Motor with 8086 by using 8255 to Rotate Stepper Motor in Anti-clock wise directions	
7.	**INTERFACING ADC/DAC WITH 8086** (a) To Interface ADC (b) To Interface DAC	
8.	**COMMUNICATION BETWEEN TWO MICROPROCESSORS** (a)Parallel communication between two microprocessors kits by using 8255. (b) Serial communication between two microprocessors kits by using 8251.	
9.	**PROGRAMS USING ARITHMETIC AND LOGICAL INSTRUCTIONS FOR 8051** (a) Arithmetic operations (b) Logical operations	
10.	**PROGRAM TO VERIFY TIMERS/COUNTERS OF 8051** (a) To Perform Timer 0 and Timer 1 in timer Mode (b) To Perform Timer 0 and mode 1 in timer Mode	
Advanced experiments (Beyond Curriculum)		
11.	To find LCM for the given data.	
12.	Interfacing 8279 keyboard with 8051.	

DO's & DONT's IN LABORATORY

1. While entering the Laboratory, the students should follow the dress code (Wear shoes, White Apron & Female students should tie their hair back).
2. The students should bring their observation note book, Lab manual, record note book, calculator, and necessary stationary items.
3. While sitting in front of the system, check all the cable connections and Switch on the computer.
4. If a student notices any fluctuations in power supply, immediately the same thing is to be brought to the notice of technician/lab in charge.
5. At the end of practical class the system should be switch off safely and arrange the chairs properly.
6. Each program after completion should be written in the observation note book and should be corrected by the lab in charge on the same day of the practical class.
7. Each experiment should be written in the record note book only after getting signature from the lab in charge in the observation note book.
8. Record should be submitted in the successive lab session after completion of the experiment.
9. 100% attendance should be maintained for the practical classes.

SCHEME OF EVALUVATION

S NO	DATE	NAME OF EXPERIMENT	TOTAL (20M)			
			Observation (10M)	Viva voce (10M)	Sign.	Total
1.		PROGRAMS FOR 16 BIT ARITHMETIC OPERATION				
2.		PROGRAMS INVOLVING BIT MANIPULATION INSTRUCTIONS				
3.		PROGRAMS ON ARRAYS FOR 8086				
4.		PROGRAMS FOR STRING MANIPULATION FOR 8086				
5.		PROGRAM FOR DIGITAL CLOCK DESIGN USING 8086				
6.		INTERFACING STEPPER MOTOR WITH 8086				
7.		INTERFACING ADC/DAC WITH 8086				
8.		COMMUNICATION BETWEEN TWO MICROPROCESSORS				
9.		PROGRAMS USING ARITHMETIC AND LOGICAL INSTRUCTIONS FOR 8051				
10.		PROGRAM TO VERIFY TIMERS/COUNTERS OF 8051				
Advanced experiments (Beyond Curriculum)						
11.		TO FIND LCM FOR THE GIVEN DATA				
12.		INTERFACING 8279 KEYBOARD WITH 8051.				

Signature of Lab In-charge

8086 MICROPROCESSOR

INTRODUCTION TO MASM PROGRAMMING

The microprocessor development system consists of a set of hardware and software tools. The hardware of development systems usually contains a standard PC (Personal Computer), printer and an emulator. The software tools are also called program development tools and they are Editor, Assembler, and Library builder, Linker, Debugger and Simulator. These software tools can be run on the PC in order to write, assemble, debug, modify and test the assembly language programs.

EDITOR (TEXT EDITOR):

The Editor is software tool which, when run on a PC, allow the user to type/enter and modify the assembly language program. The editor provides a set of commands for insertion, deletion, modifications of letters, characters, statements, etc., The main faction of an editor is to help the user to constrict the assembly language program in the right format. The program created using editor is known as source program and usually it is saved with file extension "ASM".

ASSEMBLER:

The assembler is a software tool which run on a PC, converts the assembly language program to machine language program. Several types of assemblers are available and they are one pass assembler, two pass assembler, macro assembler, cross assembler, resident assembler and Meta assembler.

One Pass Assembler: In the one pass assembler source code is processed only once, and we can use only backward reference.

Two Pass Assembler: Most of the popularly used assemblers are two pass assembler. In two pass assembler, the first pass is made through source code for the purpose of assigning an address to all the labels and to store this information in a symbol table. The second pass is made to actually translate the source code into machine code.

Some examples of assemblers are TASM (Borland's Turbo Assembler), MASM (Microsoft Macro Assembler), ASM86 (INTEL'S 8086 Assembler), etc,.

TASM:

The Turbo Assembler (TASM) mainly PC-targeted assembler package was Borland's offering in the X86 assembler programming tool market. As one would expect, TASM worked well with Borland's high-level language compilers for the PC, such as Turbo Pascal, Turbo Basic and Turbo C. Along with the rest of the Turbo suite, Turbo Assembler is no longer maintained.

The Turbo Assembler package came bundled with the linker Turbo Linker, and was interoperable with the Turbo Debugger. For compatibility with the common Microsoft Macro Assembler

(MASM), TASM was able to assemble such source code files via its MASM mode. It also had an ideal mode that enabled a few enhancements.

The effective execution of a program in assembly language we need the following

1. MASM assembler
2. NE (Norton's Editor) editor (or) Edlin editor
3. Linker
4. Debug utility of DOS

How to use TASM:

Install the specified TASM software on PC with DOS operating system. The program implementation and its execution are illustrated in four stages, there are

1. Editing of program
2. Assembling the program
3. Linking the program
4. Debugging and execution of the program

C:\ tasm>edit (file name).asm ↵

>tasm (file name).asm ↵ **Assembling**

Turbo assembler version 3.2 copy right (C) 1988, 1992 Borland International
Assembling file : (file name)
Error messages : None
Warning messages : None
Passes : 1

PROGRAM:

Remaining memory : 392K

C:\ tasm>tlink (file name).obj ↵ **Turbo Linking**

Turbo link version 5.1 copy right (C) 1992 Borland International **the Program**

C:\ tasm>debug (file name).exe ↵ **Debugging & Operation**

- r ↵ **of the**
- p ↵ **Program**

- q ↵

DEBUG COMMANDS:

Command	Command Character & Syntax	Description
Assembler	- A [address]	Assembles the instructions at a particular address.
Quit	- q	Quits from debug
Compare	- C range address	Compares two memory ranges
Display	- D range	Displays the contents of memory
Enter	- E address [List]	Enters new or modifies old memory contents
Fill	- F range list	Fills in a range of memory
Go	- G – address	Executes a program in memory
Hex	- H V_1V_2	Adds & Subtracts two hex values (V_1 & V_2)
Load	- L [address] [drive]	Load disk data into memory
Trace	- T	Traces disk data into memory
Un assemble	- U	Un assembles hex bytes into assembler instructions

MASM:

The Microsoft Macro Assembler (abbreviated MASM) is an x86 high-level assembler for DOS and Microsoft Windows. Currently it is the most popular x86 assembler. It supports a wide variety of macro facilities and structured programming idioms, including high-level functions for looping and procedures. Later versions added the capability of producing programs for Windows. MASM is one of the few Microsoft development tools that target 16-bit, 32-bit and 64-bit platforms. Earlier versions were MS-DOS applications. Versions 5.1 and 6.0 were OS/2 applications and later versions were Win32 console applications. Versions 6.1 and 6.11 included Phar Lap's TNT DOS extender so that MASM could run in MS-DOS.

MASM can be used along with a link program to structure the codes generated by MASM in the form of an executable file. This assembler reads the source program as its inputs and provides an object file. The link accepts the object file produced by this MASM assembler as input and produces an EXE file.

The effective execution of a program in assembly language we need the following

1. MASM assembler
2. NE (Norton's Editor) editor (or) Edlin editor
3. Linker
4. Debug utility of DOS

LIBRARY BUILDER:

The library builder is used to create library files which are collection of procedures of frequently used functions.

The input to library builder is a set of assembled object of program modules/procedures.

The library builder combines the program modules/procedures into a single file known as library file and it is saved with file extension ".LIB". Some examples of library builder are Microsoft's LIB Borland's TLIB, etc,.

LINKER:

The linker is a software tool which is used to combine releasable object files of program modules and library functions into a single executable file.

The linker also generates a link map file which contains the address information about the linked files. Some examples of linkers Microsoft's linker LINK, Borland's Turbo linker TLINK, etc,.

DEBUGGER:

The debugger is a software tool that allows the execution of a grogram in single step or break-point mode under the control of user. The process of locating and correcting the errors in a program using a debugger is known as debugging.

The debugger tools can help the user to isolate a problem in the program. Once the problem/errors are identified, the algorithm can be modified. Then the user can the editor to correct the source program, reassemble the corrected source program, relink and run the program again.

SIMULATOR:

The simulator is a program which can be run on the development system (Personal computer) to simulate the operations of the newly designed system. Some of the operations that can be simulated are given below.

- Execute a program and display result.
- Single step execution of a person.
- Break – point execution of a program.
- Display the contents of register/memory.

EMULATOR:

An emulator is a combination of hardware and software. It is usually used to test and debug the hardware and software of a newly designed microprocessor based system. The emulator has a multi core cable which connects the PC of development system and the newly designed hardware of microprocessor system.

VARIABLES AND CONSTRAINTS USED IN ASSEMBLERS:

Variables: The variables are symbols (or terms) used in assembly language program statements in order to represent variable data and address. While running a program, a value has to be attached to each

variable in the program. The advantage of using variables is that the value of the variable can be dynamically varied while running program.

Rules of Framing Variable names:

1. The variable name can have any of the following characters. A to Z a to z, 0 to 9
   @ , _ (underscore).
2. The first character in the variable name should be an alphabet (A to Z or a to z) or an underscore.
3. The length of variable name depends on assembler and normally the maximum length of variable name is 32 characters.
4. The variable name are case insensitive. Therefore the assembler do not distinguish between the upper and lower case letters/alphabets.

Constraints: The decimal, binary or hexadecimal number used to represent the data address in assembly language program statement is called constants or numerical constants. When constants are used to represent the address the address the address/data then their values are fixed and cannot be changed while running a program. The binary, hexadecimal and decimal constants can be differentiated by placing a specific alphabet at the end of the constant.

Example of Valid Constant:

1011 --------- Decimal (BCD) constant

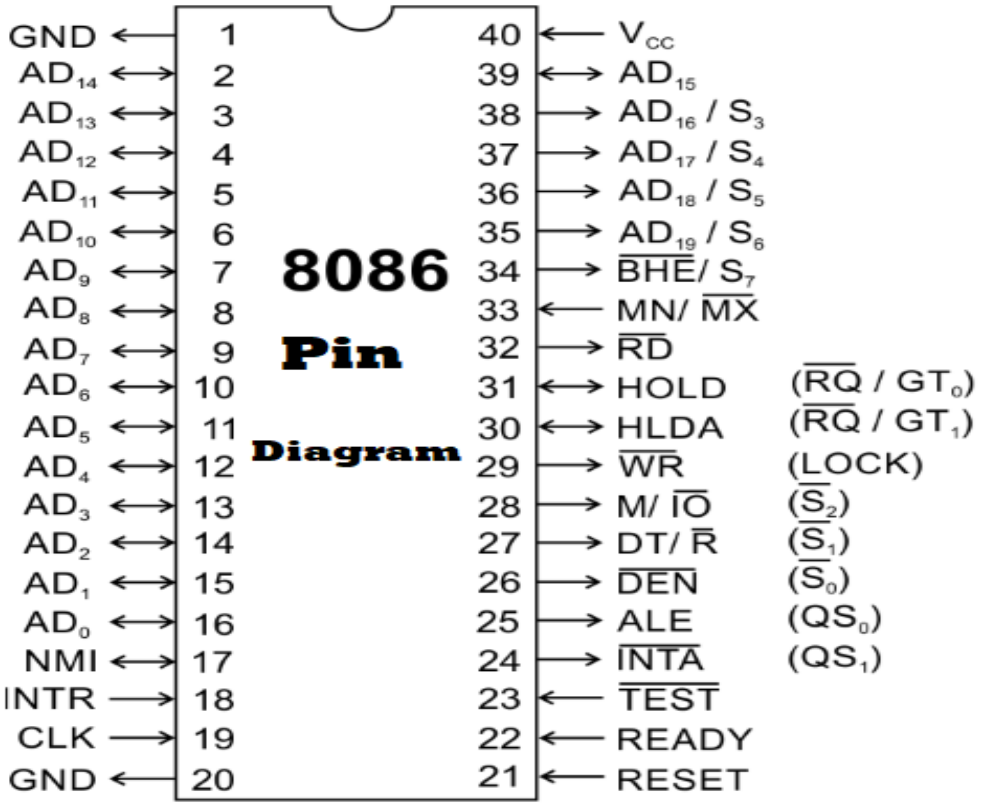
1060 D ------ Decimal constant

Examples of invalid Constant:

1131 B ------ The character 3 should not be used in binary constant.

0E2 ---------- The character H at the end of hexadecimal number in missing.

PIN DIAGRAM:


8086
Pin
Diagram
GND 1
AD14 2
AD13 3
AD12 4
AD11 5
AD10 6
AD9 7
AD8 8
AD7 9
AD6 10
AD5 11
AD4 12
AD3 13
AD2 14
AD1 15
AD0 16
NMI 17
INTR 18
CLK 19
GND 20
40 VCC
39 AD15
38 AD16 / S3
37 AD17 / S4
36 AD18 / S5
35 AD19 / S6
34 BHE/ S7
33 MN/ MX
32 RD
31 HOLD (RQ / GT0)
30 HLDA (RQ / GT1)
29 WR (LOCK)
28 M/ IO (S2)
27 DT/ R (S1)
26 DEN (S0)
25 ALE (QS0)
24 INTA (QS1)
23 TEST
22 READY
21 RESET

1.1.A. MULTI-BYTE 16-BITADDITION:

MEMORY LOCATION	OPCODE	LABEL	INSTRUCTION

OBERVATION TABLE: 1.1.A. MULTI-BYTE 16-BITADDITION

MEMORY LOCATION	DATA	MEMORY LOCATION	DATA
INPUT-1		OUTPUT	
INPUT-2			

EXP. NO: 01 **DATE:**

16 BIT ARITHMETIC OPERATIONS

AIM: To write an ALP program (a) to perform Addition and Subtraction of multi precision numbers (b) to perform Multiplication and Division of signed and unsigned numbers and (c) to find Square, Cube and factorial of a given numbers.

APPARATUS REQUIRED:

(i) 8086 Microprocessor Kit
(ii) TASM Software/Win86E
(iii) FPS (+5V)
(iv) PC
(v) USB Cable

ALGORITHM FOR 16-BIT ADDITION:

FLOW CHART FOR 16-BIT ADDITION

PROGRAMS:

1.1.A. 16-BITADDITION:

```
.MODEL SMALL
.STACK 100
.DATA
        OPR1  DW  8888H
        OPR2  DW  6666H
        RES  DW  3  DUP(?),'$'
.CODE
        MOV  AX,@DATA
        MOV  DS,AX
        MOV  AX,OPR1
        MOV  BX,OPR2
        ADD  AX,BX
        MOV  RES,AX
        MOV AL, 00H
        RCL   AL,01H
        MOV  [RES+2],AX
        MOV  AH,09H
        MOV  DX,OFFSET RES
        INT  21H
        MOV  AH,4CH
        INT  21H
        END
```

CALCULATION:

INPUT: 1 OPR1 = 8888 H
OPR2 = 6666 H

OUTPUT: RES = 00EEEE H

INPUT: 2 OPR1 =
OPR2 =

OUTPUT: RES =

1.1.B.MULTI-BYTE 16-BIT SUBTRACTION:

MEMORY LOCATION	OPCODE	LABEL	INSTRUCTION

OBERVATION TABLE: 1.1.B.MULTI-BYTE 16-BIT SUBTRACTION:

INPUT-1		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA
INPUT-2			

1.1.B. 16-BIT SUBTRACTION

```
.MODEL SMALL
.STACK 100
.DATA
        OPR1  DW  8888H
        OPR2  DW  6666H
        RES  DW  3  DUP(?),'$'
.CODE
        MOV  AX,@DATA
        MOV  DS,AX
        MOV  AX,OPR1
        MOV  BX,OPR2
        SUB  AX,BX
        MOV  RES,AX
        MOV AL, 00H
        RCL   AL,01H
        MOV  [RES+2],AX
        MOV  AH,09H
        MOV  DX,OFFSET RES
        INT  21H
        MOV  AH,4CH
        INT  21H
END
```

CALCULATION:

INPUT: 1 OPR1 = 8888 H
OPR2 = 6666 H

OUTPUT: RES = 2222 H

INPUT: 2 OPR1 =
OPR2 =

OUTPUT: RES =

ALGORITHM FOR 16-BIT SUBTRACTION:

FLOWCHART FOR 16-BIT SUBTRACTION:

1.2. A.MULTIPLICATION OF TWO 16-BIT SIGNED NUMBERS

ALGORITHM:

FLOWCHART:

1.2. A.MULTIPLICATION OF TWO 16-BIT SIGNED NUMBERS

```
.MODEL  SMALL
.STACK  100
.DATA
        OPR1  DW  -7593H
        OPR2  DW  6845H
        RES    DW  02 DUP(?),'$'
.CODE
        MOV  AX,@DATA
        MOV  DS,AX
        MOV  AX,OPR1
        IMUL OPR2
        MOV  RES,AX
        MOV  [RES+2],DX
        MOV  AH,09H
        MOV  DX,OFFSET RES
        INT    21H
        MOV  AH,4CH
        INT    21H
 END
```

CASE I: Two positive numbers
INPUT: OPR1 = 7593H
OPR2 = 6845H

OUTPUT: RESLW = 689FH (AX)
RESHW = 2FE3H (DX)

CASE II: one positive number & one negative number
INPUT: OPR1 = 8A6DH ← 2's Complement of (-7593H)
OPR2 = 6845H

OUTPUT: RESLW = 9761H (AX) ← 2's Complement
RESHW = D01CH (DX) ← of (- 2FE3689FH)

CASE III: two negative numbers
INPUT: OPR1 = 8A6DH ← 2's Complement of (-7593H)
OPR2 = 97BBH ← 2's Complement of (-6845H)

OUTPUT: RESLW = 689FH (AX)
RESHW = 2FE3H (DX)

1.2. A.MULTIPLICATION OF TWO 16-BIT SIGNED NUMBERS

MEMORY LOCATION	OPCODE	LABEL	INSTRUCTION

OBERVATION TABLE MULTIPLICATION OF TWO 16-BIT SIGNED NUMBERS:

INPUT-1		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA
INPUT-2			

ALGORITHM:

FLOWCHART

1.2.B.DIVISION OF TWO 16-BIT SIGNED NUMBERS:

MEMORY LOCATION	OPCODE	LABEL	INSTRUCTION

OBERVATION TABLE DIVISION OF TWO 16-BIT SIGNED NUMBERS:

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA

1

1.2.B.DIVISION OF TWO 16-BIT SIGNED NUMBERS

```
.MODEL  SMALL
.STACK  100
.DATA
        OPR1  DW  -26F8H
        OPR2  DB    56H
        RES     DW  02 DUP(?),'$'
.CODE
        MOV  AX,@DATA
        MOV  DS,AX
        MOV  AX,OPR1
        IDIV   OPR2
        MOV  RES,AX
        MOV  [RES+2],DX
        MOV  AH,09H
        MOV  DX,OFFSET RES
        INT     21H
        MOV  AH,4CH
        INT     21H
END
```

CALCULATION:

CASE I: Two Positive Numbers

INPUT: OPR1 = 26F8H (DIVIDEND)
OPR2 = 56H (DIVISOR)

OUTPUT: RESQ = 74H (AL)
RESR = 00H (AH)

CASE II: One Positive Number & One Negative Number

INPUT: OPR1 = D908H ← 2's Complement of (-26F8H)
OPR2 = 56H

OUTPUT: RESQ = 8CH (AL) ← 2's Complement of (- 74H)
RESR = 00H (AH)

CASE III: One Positive Number & One Negative Number

INPUT: OPR1 = 26F8H
OPR2 = AAH ← 2's Complement of (-56H)

OUTPUT: RESQ = 8CH (AL) ← 2's Complement of (- 74H)
RESR = 00H (AH)

FLOWCHART

CALCULATION:

1.2.C. MULTIPLICATION OF TWO 16-BIT UNSIGNED NUMBERS:

MEMORY LOCATION	OPCODE	LABEL	INSTRUCTION

OBERVATION TABLE DIVISION OF TWO 16-BIT UNSIGNED NUMBERS:

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA

1.2.C. MULTIPLICATION OF TWO 16-BIT UNSIGNED NUMBERS:

```
.MODEL  SMALL
.STACK  100
.DATA
        OPR1  DW  7777H
        OPR2  DW  8888H
        RES  DW  2  DUP(?),'$'
.CODE
        MOV  AX,@DATA
        MOV  DS, AX
        MOV  AX, OPR1
        MOV  BX, OPR2
        MUL  BX
        MOV  RES, AX
        MOV  [RES+2], DX
        MOV  AH,09H
        MOV  DX, OFFSET RES
        INT  21H
        MOV  AH, 4CH
        INT  21H
END
```

CALCULATION:

INPUT: 1 OPR1 = 7777 H
OPR2 = 8888 H

OUTPUT: RES = 3FB6AF38 H

INPUT: 2 OPR1 =
OPR2 =

OUTPUT: RES =

ALGORITHM:

FLOWCHART

1.2. D.DIVISION OF TWO 16-BIT UNSIGNED NUMBERS

MEMORY LOCATION	OPCODE	LABEL	INSTRUCTION

OBERVATION TABLE DIVISION OF TWO 16-BIT UNSIGNED NUMBERS:

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA

1.2. D.DIVISION OF TWO 16-BIT UNSIGNED NUMBERS

```
.MODEL  SMALL
.STACK  100
.DATA
        OPR1  DW 0FFFE H
        OPR2  DW 3333 H
        RES  DW 2 DUP (?),'$'
.CODE
        MOV  AX,@DATA
        MOV  DS, AX
        XOR  AX, AX
        MOV  AX, OPR1
        MOV  BX, OPR2
        DIV   BX
        MOV  RES, AX
        MOV  [RES+2], DX
        MOV  AH, 09H
        MOV  DX, OFFSET RES
        INT  21
        END
```

CALCULATION:

INPUT: 1 OPR1 = 0FFFE H
OPR2 = 3333 H

OUTPUT: QUE = 0004 H
REM = 3332 H

INPUT: 2 OPR1 =
OPR2 =

OUTPUT: QUE =
REM =

ALGORITHM:

FLOWCHART

CALCULATION

1.3. A.SQUARE OF A GIVEN NUMBER

MEMORY LOCATION	OPCODE	LABEL	INSTRUCTION

1.3.B.CUBE OF A GIVEN NUMBER

MEMORY LOCATION	OPCODE	LABEL	INSTRUCTION

1.3. C.FACTORIAL OF A GIVEN NUMBER

MEMORY LOCATION	OPCODE	LABEL	INSTRUCTION

OBERVATION TABLE SQUARE OF A GIVEN NUMBER

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA

1.3.A.SQUARE OF A GIVEN NUMBER

```
.MODEL SMALL
.STACK
.DATA
 X DB 08H                  ; NUMBER TO BE SQUARED
 SQR DW (?)                ; LOCATION TO STORE NUMBER
.CODE
        MOV AX, @DATA   ; INITIALIZE DATA SEGMENT
        MOV DS, AX
        MOV AL,X
        MUL AL
        MOV SQR,AX        ; SQUARE THE NUMBER
        MOV AH, 4CH
        INT 21H
 END                      ; END PROGRAM
```

ALGORITHM:

FLOWCHART

1.3.B.CUBE OF A GIVEN NUMBER

```
.MODEL SMALL
.DATA
 X DB 02H                    ; NUMBER TO BE SQUARED
 CUB DW (?)                  ; LOCATION TO STORE NUMBER
.CODE
        MOV AX, @DATA        ; INITIALIZE DATA SEGMENT
        MOV DS, AX
        MOV AL, X           ; STORE THE NUMBER IN AL REGISTER
        MUL  AL
         MOV  BL,AL
        MOV AL, X
         MUL  BL
        MOV CUB, AX          ; SQUARE THE NUMBER
        MOV CUB+2, DX
        MOV AH,4CH
        INT 21H
 END                         ; END PROGRAM
```

ALGORITHM:

FLOWCHART

1.3. C.FACTORIAL OF A GIVEN NUMBER

```
. Model small
. Stack
. Data
. Code
        MOV  AX, @DATA
        MOV  DS, AX
        MOV  AX, 0001H
        MOV  CL, 05H
  UP: MUL   CL
        DEC   CL
        JNZ  UP
        INT 21H
   END
```

ALGORITHM:

FLOWCHART

OBERVATION TABLE CUBE OF A GIVEN NUMBER

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA

OBERVATION TABLE FACTORIAL OF A GIVEN NUMBER

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA

CALCULATION

RESULT:

CONCLUSION:

VIVA QUESTIONS:

1. Define IMUL &IDIV?

2. What is the BX &DX register roles in multiplication and division?

3. Why 8086 is called as 16 bit microprocessor?

4. What is the use of source index (SI)?

5. Functions of DX register?

2. A.POSITIVE AND NEGATIVE NUMBERS

MEMORY LOCATION	OPCODE	LABEL	INSTRUCTION

OBERVATION TABLE POSITIVE AND NEGATIVE NUMBERS

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA

EXP. NO:02 **DATE:**

BIT MANIPULATION INSTRUCTIONS

AIM: To write an assembly language program (a) to find the given date is positive or negative, (b) to find the given date is odd or even and (c) to find the given date is logical ones or zeros using 8086 bit manipulation instructions.

APPARATUS REQUIRED:

(i) 8086 Microprocessor Kit
(ii) TASM Software/Win86E
(iii) FPS (+5V)
(iv) PC
(v) USB Cable

ALGORITHM:

FLOWCHART

PROGRAMS

2. A.POSITIVE AND NEGATIVE NUMBERS

```
        .MODEL SMALL
        .DATA

         ARRAY DB 12H, -98H,-45H,83H,-28H, 67H, 92H, -54H, -63H, 76H
         NEGI  DB 10 DUP (?)
         POSI  DB 10 DUP (?)
 .CODE
        MOV AX, @DATA   ; INITIALIZE THE DATA SEGMENT
        MOV DS, AX
        MOV CL, 0AH     ; INITIALIZE THE COUNTER
        XOR DI, DI      ; INITIALIZE THE POINTER FOR NEGATIVE NUMBER
        XOR SI, SI      ; INITIALIZE THE POINTER FOR POSITIVE NUMBER
        LEA BP, ARRAY:
BACK:
        MOV AL, DS:[BP] ; GET THE NUMBER
        TEST AL, 80H    ; MASK ALL BITS EXCEPT
        MSB
        JZ NEXT         ; IF LSB = 0 GOT TO NEXT
        LEA BX, NEGI
        MOV [BX+DI], AL
        INC DI          ; INCREMENT THE NEGATIVE POINTER
         JMP SKIP
 NEXT:
        LEA BX, POSI
        MOV [BX+SI], AL
        INC SI          ; INCREMENT THE POSITIVE POINTER
SKIP:
        INC BP          ; INCREMENT ARRAY BASE POINTER
        LOOP BACK       ; DECREMENT THE COUNTER
        MOV AH, 4CH
        INT 21H
        END             ; END PROGRAM
```

2. B.ODD AND EVEN NUMBERS

MEMORY LOCATION	OPCODE	LABEL	INSTRUCTION

OBERVATION TABLE ODD AND EVEN NUMBERS

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA

2. B.ODD AND EVEN NUMBERS

```
.MODEL SMALL
.DATA
 ARRAY  DB  12H,  98H,  45H,  83H,  28H,  67H,  92H,  54H,  63H,  76H
ARR_EVEN DB10 DUP (?)
ARR_ODD DB 10 DUP (?)
.CODE
   MOV AX, @DATA   ; INITIALIZE THE DATA SEGMENT
   MOV DS, AX
  MOV CL, 0AH      ; INITIALIZE THE COUNTER
   XOR DI, DI      ; INITIALIZE THE ODD POINTER
   XOR SI, SI      ; INITIALIZE THE EVEN POINTER
   LEA BP, ARRAY
 BACK:
   MOV AL, DS:[BP]    ; GET THE NUMBER
   TEST AL, 01H ; MASK ALL BITS EXCEPT
   LSB
   JZ  NEXT        ; IF LSB = 0 GOT TO NEXT
   LEA   BX,   ARR_ODD
   MOV [BX+DI], AL
   INC  DI         ; INCREMENT THE ODDPOINTER
   JMP  SKIP
NEXT:
  LEA BX, ARR_EVEN
  MOV [BX+SI], AL
   INC  SI         ; INCREMENT THE EVEN POINTER
   INC  BP         ; INCREMENT ARRAY BASE POINTER

SKIP:

  LOOP BACK    ; DECREMENT THECOUNTER
   MOV AH, 4CH
   INT 21H
   END             ; END PROGRAM
```

ALGORITHM:

FLOWCHART

2. C.LOGICAL ONES AND ZEROS

MEMORY LOCATION	OPCODE	LABEL	INSTRUCTION

OBERVATION TABLE LOGICAL ONES AND ZEROS:

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA

2. C.LOGICAL ONES AND ZEROS

```
.MODEL SMALL
 .DATA
  NUM DB 0FAH
   ONES DB 0
     ZEROS DB 0
 .CODE
  START:
          MOV AX, @DATA      ;  INITIALIZE THE DATA SEGMENT
           MOV  DS, AX
           MOV AL, NUM        ;  GET BYTE
           MOV CX, 08H        ;  SET COUNTER
  BACK:
           ROR AL, 1          ;  MOVE MSB IN CARRY
           JNC  ZERINC        ; CHECK BYTE FOR 0
           AND 1
           INC ONES           ; IF 1, INCREMENT ONE COUNT
           JMP NEXT
  ZERINC:
          INC ZEROS          ; IF 0, INCREMENT ZERO COUNTER
  NEXT:
           DEC CX             ; REPEAT UNIT CX = 0
          JNZ BACK
           MOV AH, 4CH
           INT 21H
           END START
           END                ; END PROGRAM
```

ALGORITHM:

FLOWCHART

RESULT:

CONCLUSION:

VIVA QUESTIONS:

1. Explain about AAA, DAA?

2. What are the logical instructions supported by 8086?

3. Name 5 different addressing modes?

4. What are Hardware interrupts in 8085?

5. What is meant by a bus?

3. A.SUM OF N NUMBERS

MEMORY LOCATION	OPCODE	LABEL	INSTRUCTION

OBERVATION TABLE SUM OF N NUMBERS

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA

EXP. NO: 03 **DATE:**

PROGRAMS ON ARRAYS FOR 8086

AIM: To write an assembly language program (a)To find addition /subtraction of N-numbers (b) To find largest/smallest numbers (c) To sort given array in ascending/descending using 8086.

APPARATUS REQUIRED:

(i) 8086 Microprocessor Kit
(ii) TASM Software/Win86E
(iii) FPS (+5V)
(iv) PC
(v) USB Cable

ALGORITHM:

FLOWCHART

3. B.LARGEST NUMBER

MEMORY LOCATION	OPCODE	LABEL	INSTRUCTION

OBERVATION TABLE 3.B.LARGEST NUMBER

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA

PROGRAM

3. A.SUM OF N NUMBERS

```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
ARRAY  DB  12H, 24H, 26H, 63H, 25H, 86H, 2FH, 33H, 10H, 35H
 SUM DW 0
 DATA ENDS
CODE SEGMENT
START:
        MOV  AX, DATA
        MOV  DS, AX
        MOV CL, 10
        XOR  DI, DI
        LEA  BX, ARRAY
 BACK: MOV AL, [BX+DI]
        MOV AH, 00H
        MOV SUM, AX
        INC  DI
        DEC  CL
        JNZ  BACK
        INT 21H
       CODE  ENDS
        END  START
```

ALGORITHM:

FLOWCHART

3. C.ASCENDING ORDER

MEMORY LOCATION	OPCODE	LABEL	INSTRUCTION

OBERVATION TABLE ASCENDING ORDER

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA

3. B.LARGEST/SMALLEST NUMBER

```
. MODEL SMALL
. STACK
. DATA
LIST DB 02H, 09H, 03H, 06H, 08H, 07H
. CODE
        MOV AX, @DATA
        MOV DS, AX
        MOV SI, OFFSET LIST
        MOV CL, 05H
        MOV AX, 0000H
        MOV AL, [SI]
  UP: INC SI
        CMP AL, [SI]
        JNB GO
        MOV AL, [SI]
  GO: LOOP UP
        INT 21H
        END
```

ALGORITHM:

FLOWCHART

3.C.ASCENDING/DESECENDING ORDER

```
.MODEL SMALL
.STACK 100
.DATA

      LIST   DB   56H,12H,72H,32H,13H
      COUNT   EQU   ($-LIST)
   .CODE

      MOV   AX,@DATA
      MOV   DS, AX
      MOV   CX, COUNT
      MOV   DX, CX

AGAIN: MOV   SI, OFFSET LIST
       MOV   CX, DX

BACK: MOV   AL,[SI]
      INC   SI
      CMP   AL,[SI]
      JC   NEXT
      XCHG   [SI],AL
      DEC   SI
      MOV   [SI],AL
      INC   SI

NEXT:LOOP   BACK
     DEC   DX
     JNZ   AGAIN
     MOV   AH, 09H
     MOV   DX, OFFSET LIST
     INT 21H

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

RESULT:

CONCLUSION:

.

VIVA QUESTIONS:

1. What is the Significance of int 21h?

2. What is the purpose of offset?

3. What is the Significance of inserting label in programming

4. Give the concept of Jump with return and jump with non return.

5. What are the flags that are effected by compare statement?

4. A.LENGTH OF THE GIVEN STRING

MEMORY LOCATION	OPCODE	LABEL	INSTRUCTION

OBERVATION TABLE LENGTH OF THE GIVEN STRING

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA

EXP NO: 4 **DATE:**

STRING MANIPULATIONS FOR 8086

AIM: To write an assembly language program (a)to find string length (b)for displaying the given string (c) for comparing two string (d)To reverse string and checking for palindrome using 8086.

APPARATUS REQUIRED:

(i) 8086 Microprocessor Kit
(ii) TASM Software/Win86E
(iii) FPS (+5V)
(iv) PC
(v) USB Cable

ALGORITHM:

FLOWCHART

4.B. DISPLAYING THE GIVEN STRING:

MEMORY LOCATION	OPCODE	LABEL	INSTRUCTION

OBERVATION TABLE DISPLAYING THE GIVEN STRING:

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA

PROGRAMS:

4. A.LENGTH OF THE GIVEN STRING

```
 .MODEL SMALL
 .STACK 100
 .DATA
        LIST   DB   'MICRO PROCESSOR$'
        LEN    DW 01 DUP (?),'$'
.CODE
        MOV AX, @DATA
        MOV  DS, AX
        MOV  ES, AX
        MOV AL,'$'
        LEA   SI, LIST
        MOV  DX, 0000H
BACK:  SCASB
        JE   EXIT
        INC  DX
        JMP   BACK
EXIT:   MOV  LEN, DX
        MOV  AH, 09H
        MOV  DX,OFFSET LEN
        INT   21H
        MOV AH, 4CH
        INT  21H
        END
```

4.B. DISPLAYING THE GIVEN STRING:

```
.MODEL SMALL
.STACK 100
.DATA
STRING DB "VEMU INSTITUTE OF TECHNOLOGY", $
.CODE
MOV  AX,@DATA
MOV  DS, AX
MOV  AH, 09H
LEA    DX, STRING
INT     21H
END
```

4. C. COMPARING OF TWO STRINGS :

MEMORY LOCATION	OPCODE	LABEL	INSTRUCTION

OBERVATION TABLE COMPARING OF TWO STRINGS :

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA

4. C. COMPARING OF TWO STRINGS :

```
.MODEL SMALL
.STACK 100
.DATA
        LIST1   DB  'MICRO PROCESSOR'
        LEN1    EQU   ($-LIST1)
        LIST2   DB   'MICRO PROCESSOR'
        LEN2    EQU  ($-LIST2)
        MSG1  DB  0DH, 0AH, "STRINGS ARE EQUAL", 0DH, 0AH,"$"
        MSG2  DB  0DH, 0AH, "STRIANGS ARE NOT EQUAL", 0DH, 0AH,"$"

.CODE

        MOV   AX,@DATA
        MOV   DS, AX
        MOV   ES, AX
        MOV   AX, LEN1
        MOV   BX, LEN2
        CMP    AX, BX
        JNE     EXIT
        MOV   CX, AX
        MOV   SI, OFFSET LIST1
        MOV   DI, OFFSET LIST2
        REP    CMPSB
        JNE     EXIT
        MOV   AH, 09H
        MOV   DX, OFFSET MSG1
        INT     21H
        JMP    NEXT
EXIT:  MOV   AH, 09H
        MOV   DX,OFFSET MSG2
        INT     21H
NEXT: MOV  AH, 4CH
        INT     21H
        END
```

ALGORITHM:

FLOWCHART

4. D. REVERSE STRING AND CHECKING FOR PALINDROME

MEMORY LOCATION	OPCODE	LABEL	INSTRUCTION

OBERVATION TABLE

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA

4. D. REVERSE STRING AND CHECKING FOR PALINDROME

```
.Model Small
.Data
        Str1    Db "Alam"                   ; String To Be Checked For Palindrome
        Slen    Equ ($-Str1)
        Str2    Db 40Dup(0)
        Msg1    Db "Palindrome$"
        Msg2    Db "Not Palindrome$"
.Code

Start   Mov     Ax,@Data
        Mov     Ds,Ax
        Mov     Es,Ax                       ; Initialize Extra Segment

        Mov     Cx,Slen                     ; Length Of The String
        Lea     Si, Str1
        Add     Si,Slen – 1                 ; Get The Last Byte Of The Data
        Lea     Di, Str2


Up:     Mov     Al,[si]
        mov     [di],al                     ; Load The Byte From [Si] To [Di]
        dec     si
        inc     Di
        loop    up                          ; Repeat The Process

        Lea  si ,str

                Lea di,str2

                Mov cx,slen                 ; Clear The Direction Flag

                cld
Repe    Cmpsb                               ; Compare The String Bytes Present In Si & Di
        Jne     Down                        ; Jump If The Strings Are Not Equal


Down:   Lea     Dx, Msg1
        Jmp     Down1
        Lea     Dx,Msg2

Down1:  Mov Ah, 09h
                Int     21h
                Int     3                           ; Terminate the Program
                End     Start
```

ALGORITHM:

FLOWCHART

RESULT:

CONCLUSION:

.

VIVA QUESTION:

1. What is a palindrome?

2. Specify string instructions.

3. What is the syntax of compare string instruction?

4. What is the use of data segment in 8086 processor?

5. What is use of index registers?

5. A. DIGITAL CLOCK DESIGN

MEMORY LOCATION	OPCODE	LABEL	INSTRUCTION

OBERVATION TABLE

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA

EXP NO: 5 **DATE:**

DIGITAL CLOCK DESIGN USING 8086

AIM:

To write an assembly language program for designing clock using INT 21H, DOS Interrupts functions and by reading system time using 8086.

APPARATUS REQUIRED:

(i) 8086 Microprocessor Kit
(ii) TASM Software/Win86E
(iii) FPS (+5V)
(iv) PC
(v) USB Cable

ALGORITHM:

FLOWCHART

5.B. DESIGNING CLOCK BY READING SYSTEM TIME

MEMORY LOCATION	OPCODE	LABEL	INSTRUCTION

OBERVATION TABLE

INPUT		OUTPUT	
MEMORY LOCATION	DATA	MEMORY LOCATION	DATA

PROGRAM

5.A. DIGITAL CLOCK DESIGN :

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
MESSAGE DB 'CURRENT TIME IS:$'
HR       DB ?
MIN     DB?
SEC      DB?
MSEC  DB?
DATA  ENDS
CODE SEGMENT
START:
                MOV  AX,DATA
                MOV   DS,AX
                MOV   AH,2CH
                INT     21H
                MOV   HR,CH
                MOV   MIN,CL
                MOV   SEC,DH
                MOV  MSEC,DL
                MOV   AH,09H
                LEA    DX, MESSAGE
                INT     21H
                MOV    AL,HR
                AND    AL,AL
                AAM
                MOV    BX,AX
                CALL DISPLAY
                MOV    DL,':'
                MOV   AH,02H
                INT      21H
                MOV   AL,MIN
                AAM
                MOV   BX,AX
                CALL DISPLAY
                MOV    DL,':'
                MOV    AH,02H
                INT     21H
                MOV   AL,SEC
                AAM
                MOV   BX,AX
                CALL DISPLAY
                MOV  DL,':'
```

```
 MOV   AH,02H
 INT     21H
 MOV   AL, MSEC
 AAM
 MOV   BX,AX
CALL DISPLAY
 MOV   AH,4CH
INT      21H
 DISPLAY   PROC NEAR
 MOV  DL,BH
 ADD  DL,30H
 ADD  DL,30H
 MOV AH,02H
 INT  21H
 MOV  DL,BL
 ADD  DL,30H
 MOV AH,02H
 INT  21H
 RET
 DISPLAY  ENDP
CODE ENDS
END START
```

ALGORITHM:

FLOWCHART

5.B. DIGIAL CLOCK BY READING SYSTEM TIME

```
ASSUME CS: CODE
CODE SEGMENT
EXTERN GET_TIME: NEAR
.MODEL SMALL
.STACK 100H
.DATA
TIME_BUF DB ’00:00:00$’
.CODE
MAIN PROC
        MOV AX,@DATA
        MOV DS, AX
        LEA BX, TIME_BUF
        CALL GET_TIME
         LEA DX, TIME_BUF
         MOV AH, 09H
         INT 21H
         MOV AH, 4CH
         INT 21H
         MAIN ENDP

END MAIN
```

RESULT:

CONCLUSION:

.

VIVA QUESTIONS:

1. What is the clock rate of 8086?

2. What are the functional units in 8086?

3. What is meant by Maskable Interrupts?

4. Give example for Non-Maskable Interrupts?

5. What is the Maximum clock frequency in 8086?

INTERFACING STEPPER MOTOR

Address	OPCODE	LABEL	INSTRUCTION	COMMENT
2000	B8 00 00		MOV AX, 0000H	Initialise Segment
2003	8E C0		MOV ES, AX	Registers
2005	BA E6 FF		MOV AL,0FFE6H	Initialise
2008	B0 80		MOV AL, 80H	ALL 8255 PORTS O/P
200A	EE		OUT DX, AL	
200B	9A ED 01 00 FE		CALL FAR 0FE00:01EDH	
2010	EB 33		JMP SHORT START	
2012	0A 0D 45 4E 54 45	MES1:	DB 0AH, 0DH, 'Enter Direction'	
2018	52 20 44 49 52 45			
201E	43 54 49 4F 4E			
2023	0A 0D 41 2D 41 4E		DB 0AH, 0DH, 'A-Anticlockwise C-Clockwise', 00H	
2029	54 49 43 4C 4F 43			
202F	4B 57 49 53 45 20			
2035	20 20 20 20 43 2D			
203B	43 4C 4F 43 4B 57			
2041	49 53 45 00			
2045	2E	CS		
2046	8D 16 12 20	START:	LEA DX, MES1	DISPLAY MESSAGE
204A	8B C2		MOV AX, DX	ON LCD OR CONSOLE
204C	9A 13 00 00 FE		CALLFAR0FE00:0013H	
2051	9A A9 00 00 FE	GET:	CALLFAR0FE00:00A9H	WAIT FOR USER ENTRY
2056	3C 41		CMP AL, 41H	IF KEY='A',
2058	74 11		JE ANTI	ROTATE ANTI-CLOCKWISE

205A	3C 43		CMP AL, 43H	IF KEY='C',
205C	74 02		JE CLO	ROTATE CLOCKWISE
205E	EB F1		JMP SHORT GET	ACCEPT VALID KEY
2060	E8 13 00	CLO:	CALL COMMON	
2063	EE	R1:	OUT DX, AL	
2064	E8 1F 00		CALL DELAY	INTRODUCE DELAY
2067	D0 D8		RCR AL, 1	ROTATE BITS IN
2069	EB F8		JMP SHORT R1	DATA BYTE RIGHT & REPEAT
206B	E8 08 00	ANTI:	CALL COMMON	
206E	EE	R2:	OUT DX, AL	
206F	E8 14 00		CALL DELAY	
2072	D0 D0		RCL AL, 1	ROTATE BITS IN
2074	EB F8		JMP SHORT R2	DATA BYTE LEFT & REPEAT
2076	9A 31 00 00 FE	COMMON:	CALL FAR 0FE00:0031H	
207B	9A 00 00 00 FE	SS:	CALL FAR 0FE00:00H	
2080	B0 11		MOV AL, 11H	; OUTPUT VALUE
2082	BA E0 FF		MOV DX, 0FFE0H	TO PORT A
2085	C3		RET	
2086	B9 00 08	DEALY:	MOV CX, B00H	; DELAY ROUTINE
2089	E2 FE		LOOP SS	
208B	C3		RET	

EXP NO:06 **DATE:**

INTERFACING STEPPER MOTOR WITH 8086

AIM: Write an ALP to 8086 processor to interface a Stepper Motor by using 8255 to rotate Stepper Motor in Clockwise &Anti-clock wise directions by choosing variable step-size.

APPARATUS REQUIRED:

i) 8086 Microprocessor Kit
ii) FPS (+5V)
iii) 8255 Interface
iv) Stepper Motor

PROCEDURE:-

1. Connect power supply 5V & GND to both microprocessor trainer kit & Stepper motor inter facing kit.
2. Connect data bus between microprocessor trainer kit & Stepper motor interfacing kit.
3. Enter the program to rotate Stepper motor in clock wise & anticlockwise.
4. Execute the program by typing GOE.
5. Assume the interface is connectedoverJ4 of the trainer.
6. This program illustrates the control of direction of rotation of the Stepper motor depending upon user choice.
7. The program executes in a continuous loop.
8. The program can be executed in STAND-ALONE MODE or SERIAL MODE of operation.
9. The program starts at memory location 0:2000H
10. PleasereferESA86/88E user's manual for mnemonic

PROGRAM FOR CLOCKWISE AND ANTI-CLOCKWISE DIRECTION:

```
MOV AX, 0000H      ; INITIALISE SEGMENT
MOV ES, AX         ; REGISTERS
MOV DX, AL         ; INITIALISE
MOV AL, 80H        ; ALL 8255 PORTS O/P
OUT DX, AL
CALL FAR 0FE00:01EDH
JMP SHORT START
MES1: DB 0AH, 0DH, 'ENTER DIRECTION'
DB 0AH, 0DH, 'A-ANTICLOCKWISE
                C-CLOCKWISE', 00H
START:
LEA DX, MES1       ; DISPLAY MESSAGE
MOV AX, DX         ; ON LCD OR CONSOLE
CALL FAR 0FE00:0013H
GET:
CALL FAR 0FE00:00A9H     ; WAIT FOR USER
CMP AL, 41H              ; ENTRY IF KEY='A',
JE   ANTI                ; ROTATE ANTI-CLOCKWISE
CMP AL, 43H              ; IF KEY='C'
JE CLO                   ; ROTATE CLOCKWISE
JMP SHORT GET            ; ACCEPT VALID KEY ONLY
```

; ROUTINE FOR CLOCKWISE ROTATION OF MOTOR:

```
CLO:    CALL COMMON
  R1:   OUT DX, AL
        CALL DELAY        ; INTRODUCE DELAY
        RCR AL, 1         ; ROTATE BITS IN
        JMP SHORT R1      ; DATA BYTE RIGHT &
                          ; REPEAT
```

; ROUTINE FOR ANTI- CLOCKWISE ROTATION OF MOTOR:

```
ANTI:  CALL COMMON
   R2:  OUT DX, AL
        CALL DELAY
        RCL AL, 1              ; ROTATE BITS
        JMP SHORT R2           ; LEFT & REPEAT
COMMON: CALL FAR 0FE00:0031H
CALL FAR 0FE00:00H
  MOV AL, 11H                  ; OUTPUT VALUE
  MOV DX, 0FFE0H               ; TO PORT A
RET
DELAY:
      MOV CX, B00H          ; DELAY ROUTINE
SS:  LOOP SS
 RET
```

RESULT:

CONCLUSION:

.

VIVA QUESTIONS:

1. What is difference between shifts and rotate instructions?

2. Which are strings related instructions?

3. Which are addressing modes and their examples in 8086?

4. What are the interrupts of 8086?

5. What is the data and address size in 8086?

INTERFACING ADC

ADDRESS	OPCODE	LABEL	INSTRUCTION	COMMENT
2000	B8 00 00			
2003	8E C0			
2005	BA E6 FF			
2008	B0 8B			
200A	EE			
200B	EB 1F			
200D	0D 43 48 41 4E 4E			
2013	45 4C 20 4E 4F 3A			
2019	20 00			
201B	0A 0D 44 49 47 49			
2021	54 41 4C 20 56 41			
2027	4C 55 45 3A 00			
202C	B9 00 00			
202F	E8 08 00			
2032	E8 3E 00			
2035	E8 77 00			
2038	EB 75			
203A	B0 00			
203C	0A C1			
203E	BA E0 FF			
2041	EE			
2042	B0 20			
2044	0A C1			
2046	EE			
2047	90			
2048	90			
2049	90			

204A	B0 00			
204C	0A C1			
204E	EE			
204F	BA E4 FF			
2052	EC			
2053	24 01			
2055	75 FB			
2057	EC			
2058	24 01			
205A	74 FB			
205C	B0 40			
205E	0A C1			
2060	BA E0 FF			
2063	EE			
2064	BA E2 FF			
2067	EC			
2068	50			
2069	B0 00			
206B	0A C1			
206D	BA E0 FF			
2070	EE			
2071	58			
2072	C3			

2073	9A 31 00 00 FE			
2078	9A 3D 01 00 FE			
207D	50			
207E	2E 8D 16 0D 20			
2083	8B C2			
2085	9A 31 00 00 FE			
208A	8A C1			
208C	3C 0A			
208E	72 02			
2090	04 06			
2092	9A 52 00 00 FE			
2097	2E 8D 16 1B 20			
209C	8B C2			
209E	9A 31 00 00 FE			
20A3	58			
20A4	9A 52 00 00 FE			
20A9	9A 31 00 00 FE			
20AE	C3			
20AF	9A A9 00 00 FE			
20B4	3C 2C			
20B6	74 0E			
20B8	3C 2D			
20BA	74 14			

20BC	3C 0D			
20BE	74 1A			
20C0	3C 1B			
20C2	74 16			
20C4	EB E9			
20C6	FE C1			
20C8	80 F9 10			
20CB	75 02			
20CD	B1 00			
20CF	C3			
20D0	FE C9			
20D2	80 F9 10			
20D5	75 02			
20D7	B1 0F			
20D9	C3			
20DA	CC			

EXP NO: 07 **DATE:**

INTERFACING ADC / DAC WITH 8086

AIM: 1. To write a ALP to 8086 processor to interface DAC for conversion of analog data to digital output.

2. To write an ALP to 8086 processor to interface DAC and generate of triangular wave/square wave.

APPARATUS REQUIRED:

i) 8086 Microprocessor Kit -1
ii) FPS (+5V)
iii) ADC Interfacing Kit
iv) DAC Interfacing Kit
V) FRC Connector.
VI) Cathode Ray Oscilloscope

PROCEDURE for DAC:

1. Assume the interface is connected over J4 of trainer
2. This program that generates a Square or Triangular wave at Xout or Yout
3. The program can be executed in Stand alone or Serial mode
4. Execute the program from memory location 2000H

PROCEDURE for ADC:

1. Assume the interface is connected over J4 of trainer
2. This program will convert analog voltage to digital voltage.
3. The program can be executed in Stand alone or Serial mode
4. Execute the program from memory location 2000H

INTERFACING DAC

ADDRESS	OPCODE	LABEL	INSTRUCTION	COMMENT
2000	B8 00 00			
2003	8E C8			
2005	8E CO			
2007	BA E6 FF			
200A	B0 80			
200C	EE			
200D	9A ED 01 00 FE			
2012	EB 3B			
2014	0A 0D 44 55 41 4C			
201A	20 44 41 43 20 49			
2020	4E 54 45 52 46 41			
2026	43 45 20			
2029	0A 0D 53 20 2D 20			
202F	53 51 55 41 52 45			
2035	20 57 41 56 45			
203A	0A 0D 54 20 2D 20			
2040	54 52 49 41 47 4C			
2046	55 41 52 20 57 41			
204C	56 45 00			
204F	2E 8D 16 14 20			
2054	8B C2			
2056	9A 13 00 00 FE			
205B	9A A9 00 00 FE			
2060	3C 53			
2062	74 2F			

2064	3C 54			
2066	74 02			
2068	EB F1			
206A	9A 31 00 00 FE			
206F	B9 FF 00			
2072	B0 00			
2074	FE C0			
2076	BA E0 FF			
2079	EE			
207A	BA E2 FF			
207D	EE			
207E	E2 F4			
2080	B9 FF 00			
2083	8B C1			
2085	FE C8			
2087	BA E0 FF			
208A	EE			
208B	BA E2 FF			
208E	EE			
208F	E2 F4			
2091	EB DC			
2093	9A 31 00 00 FE			
2098	B0 FF			
209A	BA E0 FF			
209D	EE			
209E	BA E2 FF			
20A1	EE			
20A2	B9 FF 00			

29A5	E2 FE			
20A7	B0 00			
20A9	BA E0 FF			
20AC	EE			
20AD	BA E2 FF			
20B0	EE			
20B1	B9 FF 00			
20B4	E2 FE			
20B6	EB E0.			

7. A. ADC WITH 8086

PROGRAM OF ADC:

```
        MOV AX,0000H                    ;   Initialise Segment
        MOV ES,AX                       ;   Register
        MOV DX,FFE6H                    ;   Initialise
        MOV AL,8BH                      ;   8255 Port A
         OUT DX,AL                      ;   as O/P,Port B
        JMP SHORT MAIN                  ;   Port C as I/P
        CHA: DB 0DH,'CHANNEL NO:',0H
        DIG : DB 0AH, 0DH,'DIGITAL VALUE:',0H
        MAIN: MOV CX,00H
        NEXT:CALL CONVERT               ;   Call A/D Conv. routine
CALL DISP                               ;   Call Display routine
CALL UPDT                               ;   Call Ch. Update routine
JMP SHORT NEXT                          ;    Repeat continuously
CONVERT: MOV AL,00H                     ;   Initialise
OR AL,CL                                ;   Add Channel
MOV DX,0FFE0H                           ;   Load Channel Value
OUT DX,AL
MOV AL,20H                              ;   Assert Star Signal TO ADC
OR AL,CL
OUT DX,AL
NOP
NOP
NOP
MOV AL,00H
OR AL,CL
OUT DX,AL
MOV DX,0FFE4                            ;Wait for end
EOC: IN AL,DX                           ;of conversion
AND AL,01H
JNC EOC
CHECK:IN AL,DX                          ; Check for end of conversion
AND AL,01
JE CHECK                                ; Signal From ADC
MOV AL,40H                              ; Assert Read
OR AL,CL                                ; Signal
MOV DX,0FFE0H                           ; Read Convert
Out DX,AL
MOV DX,0FFE2H
IN AL,DX
PUSH AX                                 ; De-assert read signal
MOV AL, 00H
OR AL,CL
MOV DX,0FFE0H
```

```
OUT DX,AL
POP AX
RET
Disp: CALL FAR  0FE00:0031H
      CALL FAR  0FE00:01EDH
      PUSH AX                    ; Routine to display
      LEA DX,CHA
MOV AX,DX                        ; Display channel
CALL FAR  0FE00:0013H
MOV AL,CL
CMP AL,0AH                       ; Convert channel number
JB  VALUE
ADD AL,06H
CALL FAR 0FE00:0052H             ; Display Ch.No.
LEA DX,DIG
MOV AX,DX                        : Display message for Digital o/p
CALL FAR 0FE00:0013H
POP  AX
CALL FAR 0FE00:0052H             : Display  Digital o/p
CALL FAR 0FE00:0031H
RET
CALL FAR 0FE00:00A9H             ; Get next channel key
CMP AL,2CH                       ; if key =','
JE INCR                          ; Increment Channel
CMP AL,2DH                       ; if key ='-'
JE DNCR                          ; Decrement Channel
CMP AL,0DH                       ; if key ='CR'
JE OVER                          ; OR <ESC>
CMP AL,1BH                       ; Terminate the program

JE OVER
JMP SHORT UPDT
INCR:INC CL                      ; Channel Increment  Routine
CMP CL,10H
JNE RET1
MOV CL, 00H
RET1:RET
DECR:DEC CL                      ;  Channel Decrement  Routine


CMP CL,0FFH
JNE RET2
MOV CL, 0FH
RET2:RET
OVER:INT 03H                     ; Program termination
```

7.B.DAC WITH 8086

PROGRAM OF DAC:

GENERATE SQUARE WAVE SIGNALS:

```
        MOV AX, 0000H
        MOV CS, AX
        MOV ES, AX
        MOV DX, 0FFE6H      ; INITIALISE ALL 8255
        MOV AL, 80H         ; PORTS AS O/P PORTS
        OUT DX, AL
        CALL FAR 0FE00:01EDH; NEWLINE ROUTINE
        JMP SHORT START     ; DISPLAY MESSAGE STRING

  MES: DB 0AH, 0DH,         'DUAL DAC INTERFACE '
       DB 0AH, 0DH,         'S  - SQUARE WAVE '
       DB 0AH, 0DH          'T  -  TRIAGLUAR WAVE ', 0H
START: LEA DX, MES        ; DISPLAY MESSAGE ON LCD
       MOV AX, DX         ; OR CONSOLE
       CALL FAR 0FE00:0013H
GETKEY: CALL FAR 0FE00: 00A9H; WAIT FOR USER ENTRY
       CMP AL, 53H              ; IF S, JUMP TO SQUARE
       JE SQUARE                ; WAVE ROUTINE IF T,
       CMP AL, 54H              ; JUMP TO TRIANGLE
       JE TRIANGLE              ; WAVE ROUTIN
       JMP SHORT GETKEY         ; WAIT FOR VALID KEY
```

TRIANGULAR WAVE GENERATION ROUTINE

```
TRIANGLE: CALL FAR 0FE00:0031H
  RPT 1 : MOV CX, 0FFH          ; SET COUNT
       MOV AL, 00H              ; START FROM 0
     UP: INC AL                 ; INCREMENT DATA FOR
       MOV DX, 0FFE0H           ; + IVE GOING SLOPE &
       OUT DX, AL               ; OUTPUT AT PORT A & B
       MOV DX, 0FFE2H
       OUT DX, AL
       LOOP UP
       MOV CX, 0FFH             ; SET COUNT
       MOV AX, CX               ; START FROM FFH
   DOWN: DEC AL                 ; DECREMENT DATA FOR
       MOV DX, 0FFE0H           ; -IVE GOING SLOPE AT
       OUT DX, AL               ; PORT A &  B
       MOV DX, 0FFE2H
       OUT DX, AL
       LOOP DOWN
       JMP SHORT RPT 1          ; REPEAT CONTINUOSLY
```

SQUARE WAVE GENERATION ROUTINE

```
SQUARE: CALL FAR 0FE00: 0031H
    RPT 2: MOV AL, 0FFH          ; O/P FFH AT PORTS
           MOV DX, 0FFE0H
           OUT DX, AL
           MOV DX,0FFE2H
           OUT DX,AL
           MOV CX, FFH           ; DELAY
    DLY 1: LOOP DLY 1
           MOV AL, 00H           ; O/P 0 AT PORTS
           MOV DX, 0FFE0H
           OUT DX, AL
           MOV DX, 0FFE2H
           OUT DX, AL
           MOV CX, FFH
    DLY 2: LOOP DLY 2            ; DELAY
           JMP SHORT RPT 2       ; REPEAT CONTINUOUSLY
```

RESULT:

CONCLUSION:

.

VIVA QUESTIONS:

1) Which is by default pointer for CS/ES?

2) How many segments present in it?

3) What is the size of each segment?

4) Basic difference between 8085 and 8086?

5) Which are the registers present in 8086?

8.A. PARALLEL COMMUNICATION:

ADDRESS	OPCODE	LABEL	INSTRUCTION	COMMENT

EXP NO: 08 **DATE:**

COMMUNICATION BETWEEN TWO MICROPROCESSORS

AIM: 1. To write an ALP for parallel communication between two microprocessors kits by using 8255.

2. To write an ALP for serial communication between two microprocessors kits by using 8251.

APPARATUS REQUIRED:

i) 8086 Microprocessor Kit -02
ii) FPS (+5V)
iii) 8255-PPI
iv) 8251 USART
V) FRC connector

PROGRAM OF PARALLEL COMMUNICATION:

```
     MOV AL,90
     MOV DX,3006
     OUT DX
LOOP1:MOV DX,3000
     IN AL,DX
     NOT AL
     MOV DX,3002
     OUT DX
     MOV AL,02
     MOV DX,3006
     OUT DX
     CALL DELAY
     MOV AL,03
     MOV DX,3006
     OUT DX
     CALL DELAY
     MOV AL,0A
     MOV DX,3006
     OUT DX
     CALL DELAY
     MOV AL,0B
     MOV DX,3006
     OUT DX
     CALL DELAY
     MOV AL,0E
     MOV DX,3006
     OUT DX
     CALL DELAY
     MOV AL,0F
```

```
        MOV DX,3006
        OUT DX
        CALL DELAY
        JMP LOOP1

DEALY:  MOV CX, 7FFF
        LOOP NEXT
        RET
```

8. B.SERIAL COMMUNICATION:

ADDRESS	OPCODE	LABEL	INSTRUCTION	COMMENT
2000	B0 36			
2002	BA 86 00			
2005	EE			
2006	BA 80 00			
2009	B0 0A			
200B	EE			
200C	B0 00			
200E	EE			
200F	BC 00 30			
2012	BA 92 00			
2015	EE			
2016	EE			
2017	EE			
2018	EE			
2019	E8 43 00			
201C	B0 40			
201E	EE			
201F	E8 3D 00			
2022	B0 CE			
2024	EE			
2025	E8 37 00			
2028	B0 27			
202A	EE			
202B	E8 31 00			
202E	BE 00 21			
2031	BA 92 00			

2034	EC			
2035	24 02			
2037	74 F8			
2039	BA 90 00			
203C	EC			
203D	3C 1B			
203F	8A D8			
2041	74 1B			
2043	BA 92 00			
2046	EC			
2047	24 81			
2049	3C 81			
204B	75 F6			
204D	8A C3			
204F	BA 90 00			
2052	EE			
2053	88 04			
2055	46			
2056	E9 D8 FF			
2059	EE			
205A	46			
205B	E9 E5 FF			
205E	CC			
205F	B9 02 00			
2062	E2 FE			
2064	C3			

PROGRAM OF SERIAL COMMUNICATION:

CONTROL REGISTER 8253=0080H
DATA PORT 8251=0090H
COMMAND PORT 8251=0092H

```
ORG 2000H
MOV AL,36
MOV DX,0086H
OUT DX,AL
MOV DX,0080H
MOV AL,0A
OUT DX,AL
MOV AL,00
OUT DX,AL
MOV SP,3000
MOV DX,0092
OUT DX,AL
OUT DX,AL
OUT DX,AL
OUT DX,AL
CALL DELAY
MOV AL,40
OUT DX,AL
CALL DELAY
MOV AL,CE
OUT DX,AL
CALL DELAY
MOV AL,27
OUT DX,AL
CALL DELAY
MOV SI,2100
L1:MOV DX,0092
IN AL,DX
CMP AL,1B
JE L1
MOV DX,0090
IN AL,DX
AND AL,81
CMP BL,AL
JE L3
L2:MOV DX,0092
IN AL,DX
AND AL,81
CMP AL,81
JNE L2
```

```
        MOV AL,BL
        MOV DX,0090
        OUT DX,AL
        MOV [SI],AL
        INC SI
        JMP L1
        OUT DX,AL
        INC SI
        JMP L2
    L3:INT 03
DEALY:MOV CX,0002
    A3:LOOP A3
        RET
```

RESULT:

CONCLUSION:

.

VIVA QUESTIONS:

1. Expand USART?

2. What is MODEM?

3. Where do we prefer the serial communication?

4. What is the function of instruction pointer (IP) register?

5. What is the difference between IN and OUT instructions?

8051 MICROCONTROLLER

9. A.ARITHMETIC OPERATIONS

ADDRESS	OPCODE	LABEL	INSTRUCTION	COMMENT

INPUT		OUTPUT	
REGISTERS	DATA	REGISTERS	DATA
A	08	R0	
Add with A	02	R1	
		R2	
		R3	
		R4	
		R5	

EXP NO: 09 **DATE:**

ARITHMETIC AND LOGICAL INSTRUCTIONS FOR 8051

AIM: To write an assembly language program to perform arithmetic and logical operations by using 8051microcontroller.

APPARATUS REQUIRED:

i). 8051 Microcontroller Kit -1
ii).FPS (+5V)

9. A.ARITHMETIC OPERATIONS

PROGRAM:

```
MOV A,#08H
ADD A,#02H
MOV R0,A
SUBB A,#01H
MOV R1,A
MOV f0,#03H
MUL AB
MOV R2,A
MOV R3,B
MOV f0,#03H
DIV AB
MOV R4,A
MOV R5,f0
HERE: SJMP HERE
```

9.B.LOGICAL OPERATIONS

ADDRESS	OPCODE	LABEL	INSTRUCTION	COMMENT

INPUT		OUTPUT	
REGISTERS	DATA	REGISTERS	DATA
A	08	R0	
ANLwith A	02	R1	
		R2	
		R3	
		R4	

8. B.LOGICAL OPERATIONS

PROGRAM:

```
        MOV A,#08H
        ANL A,#02H
        MOV  R0,A
        ORL A,#01H
        MOV R1,A
        XRL A,#44H
        MOV R2,A
HERE:SJMP HERE
```

RESULT:

CONCLUSION:

.

VIVA QUESTIONS:

1.What are the types of instructions in 8051?

2. Discuss the various bit manipulation instructions.

3. Discuss the various arithmetic and logical instructions.

4. Describe the 8051 oscillator and clock?

5. What is the program counter? What is its use?

10.TIMER

ADDRESS	OPCODE	LABEL	INSTRUCTION	COMMENT

```
Disassembly
      2: mov tmod,#01h
C:0x0000    758901    MOV       TMOD(0x89),#0x01
      3: start:mov th0,#0feh
C:0x0003    758CFE    MOV       TH0(0x8C),#CCAP4H(0xFE)
      4: mov tl0,#031h
C:0x0006    758A31    MOV       TL0(0x8A),#0x31
      5: setb tr0
C:0x0009    D28C      SETB      TR0(0x88.4)
      6: jnb tf0,$
C:0x000B    308DFD    JNB       TF0(0x88.5),C:000B
      7: cpl p1.0
C:0x000E    B290      CPL       T2(0x90.0)
      8: clr tr0
C:0x0010    C28C      CLR       TR0(0x88.4)
      9: clr tf0
C:0x0012    C28D      CLR       TF0(0x88.5)
     10: sjmp start
C:0x0014    80ED      SJMP      START(C:0003)
```

EXP NO: 10 **DATE:**

PROGRAM TO VERIFY TIMERS/COUNTERS IN 8051

AIM:

a) Write a program to create a delay of 25msec using Timer0 in mode 1 and blink all the Pins of P0.

b) Write a program to create a delay of 50 μsec using Timer1 in mode 0 and blink all the Pins of P2.

c) Write a program to create a delay of 75msec using counter0 in mode 2 and blink all the Pins of P1.

d) Write a program to create a delay of 80 μsec using counter1 in mode 1 and blink all the Pins of P3.

APPARATUS REQUIRED:

1. 8051 Trainer kit
2. Power Supply
3. Keyboard
4. RS 26 Core Cable
5. CRO Probes

PROGRAM:

```
        ORG 00H
       MOV TMOD,#01H
START: MOV TH0,#0FEH
       MOV TL0,#031H
        SETB TR0
        JNB TF0, $
        CPL P1.0
        CLR TR0
        CLR TF0
        SJMP START
        END
```

PROGRAM:

```
MOV  TMOD(89),#01H
HERE : MOV TL0(8A),#0F2H
MOV  TH0(8C),#0FFH
CPL  P1.0(90)
ACALL  DELAY
SJMP   HERE
DELAY:SETB  TR0(8C)
AGAIN: JNB  TF0(8D),AGAIN
CLR   TR0(8C)
CLR   TF0(8D)
RET
```

RESULT:

CONCLUSION:

VIVA QUESTIONS:

1. What are the addresses of the two timers?

2. What is the purpose of timers?

3. What are the modes in which they work?

4. What are the SFRs used to define modes for counters?

5. What is the purpose of counters?

ADVANCED EXPERMINENTS

EXP NO: 11 **Date:**

LCM FOR THE GIVEN DATA

AIM: To write an Assembly Language Program to find LCM for the given data using 8086.

APPARATUS REQUIRED:

(i) 8086 Microprocessor Kit
(ii) TASM Software/Win86E
(iii) FPS (+5V)
(iv) PC
(v) USB Cable

ALGORITHM:

PROGRAM:

ASM code:

```
        . Model small
        . Stack
        . Data
        Num1  DW 0005h
        Num2  DW 0002h
        Ans  DW  ?
        . Code
        Mov  AX, @data
        Mov  DS,AX
        Mov  AX,Num1
        Mov  BX,Num2
        Mov  dX,0000h
Next:   Push  AX
        Push DX
        Div  BX
        Cmp  DX,0000h
        JE  LAST
        POP  DX
        POP AX
        Add AX,Num1
        JNC Next
LAST:   Pop Ans+2
        Pop Ans
        Mov AH,4ch
        Int  21h
        End
```

INPUT:

OUTPUT:

RESULT:

CONCLUSION:

VIVA QUESTIONS:

1. What is stack?

2. Which interrupt has highest priority?

3. What is an Interrupt?

4. What is a compiler?

5. What is meant by Maskable Interrupts?

EXP. NO: 12 **DATE:**

INTERFACING 8279 KEYBOARD WITH 8051

AIM: To write an Assembly Language Program to display string ESA in the display field of the study card using 8279 keyboard and display controller decode method with 8051 microcontroller.

.**APPARATUS REQUIRED:**

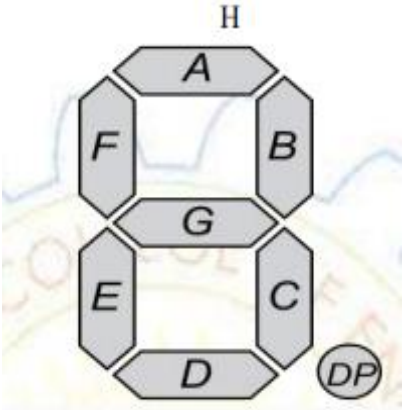
1. 8051 Trainer kit
2. FPS(+5V)
3. 8279 interfacing Keyboard
4. RS 26 core cable

ALGORITHM:

PROGRAM:

ADDRESS	OPCODE	LABEL	INSTRUCTION	COMMENT
8000			ORG 8000H	
8000	90 F1 81		MOV DPTR,#F181	
8003	74 FF		MOV A,#FF	
8005	F0		MOVX @DPTR,A	
8006	7A 90		MOV R2,#90	
8008	7B 00		MOV R3,#00	
800A	90 F1 81		MOV DPTR,#F181	
800D	74 90		MOV A,#90	
800F	F0		MOVX @DPTR,A	
8010	74 11		MOV A,#11H	
8012	F0		MOVX @DPTR,A	
8013	78 08		MOV R0,#08H	
8015	74 00	START:	MOV A,#00H	
8017	90 F1 80		MOV DPTR,#F180	
801A	F0		MOVX @DPTR,A	
801B	18		DEC R0	
801C	B8 00 F6		CJNE R0,#0,8015	
801F	79 00		MOV R1,#00	
8021	90 00 00		MOV DPTR,#9000	
8024	E4		CLR A	
8025	93	START1:	MOVC A,@A+DPTR	
8026	90 F1 80		MOV DPTR,#F180	
8029	F0		MOVX @DPTR,A	
802A	09		INC R1	
802B	90 00 00		MOV DPTR,#9000	
802E	E9		MOV A,R1	
802F	B9 04 F3		JNE R1,#04,8025	
8032	80 FE		SJMP 8032	
9000		TABLE:	ORG 9000H	
9000	04 97 D6		DB 04H,97H,D6H	
9003	77 04 04		DB 77H,04H,04H	
9006	00		DB 00H	

DISPLAY CODES:



HEX NUMBER	SEVEN SEGMENT CONVERSION								SEVEN SEGMENT DISPLAY
	D	C	B	A	DP.	G	F	E	

RESULT:

CONCLUSION:

VIVA QUESTIONS:

1. What is the function performed by 8279?

2. What are the two most important functions performed by 8279?

3. Which pin is used to blank the display of 8279?

4. When 8279 is used for keyboard and display interfacing it take care of?

5. How many types of interfacing are there?