I.Introduction to the 8086-Trainer Kit and MASM

ESA 86/88 –2 is a powerful, general-purpose microcomputer system, which can be operated either with 8086 CPU or with 8088 CPU. It is generally supplied with 8086 CPU.

The 8086 and 8088 are third generation CPU is from INTEL that differ primarily in their external data paths. 8088 uses 8-bit wide data bus while 8086 uses a 16-bit wide bus. ESA86/88-2 can be operated with either CPU and the only possible difference would be in the speed of execution (with 8088 CPU, a small speed degradation occurs because of the 8-bit wide data bus). In either case, the CPU is operated in the maximum mode.

Following are the system capabilities:

- Examine and optionally modify the contents of memory (byte or word format)
- Examine and optionally modify the processor registers.
- Assemble and Disassemble 8086/8088 instructions (via line assembler, disassembler).
- Perform fast numerical computations using the optional 8087 Numeric data processor.
- Execute the user program at full speed.
- Debug user program through single step and Breakpoint facilities.
- Write or read data to or from I/O ports (byte or word format).
- Move a block of data or program within the memory
- Download user programs into ESA 86/88-2 from a host computer system.

SPECIFICATIONS:

Central processor

8086 CPU operates at 8MHz. (Memory cycles have zero wait states and I/O cycles have one wait state).

Co-Processor

On-board 8087 Numeric Data processor (optional)

Memory

EPROM: 4 JEDEC compatible slots offer the following options:
64K bytes using 27128s or,
128K bytes using 27256s or,
256K bytes using 27512s
(System firmware is supplied in 2x27256s. The other two sockets are for user expansion).

RAM: 4 JEDEC Compatible offer the following:

128K bytes using 6255s
(64K bytes supplied using 2x62556s. The other two sockets are for user expansion).
RAM has battery backup facility.

Peripherals and Controllers

8251A: Programmable Communication Interface for serial communication supporting all standards with baud rates from 110-19,200 (Baud rate can be selected through on-board Dip switch).

8253-5: (2 Nos.) programmable peripheral Interface devices provide 48 Programmable I/O lines each.

8259A: Programmable interrupt Controller provides interrupt vectors for 8 sources

8288: Bus controller used for generating control signals.

Interrupts

External: NMI and INTR
INTR controlled through 8259A, on-board Interrupt Controller provides interrupt vectors for eight sources. Complete flexibility in selecting off-board or on-board interrupt sources.

On-board interrupt sources

#	8251	(TxRDY and RxRDY)
#	8253-5	(OUT1 and OUT2)
#	8255A	(PC0 and PC3 in handshake mode)
#	8087	(NDP INT)

Internal: Interrupt vectors 1(single step) and 3 (breakpoint) reserved for monitor.

Interface Signals

CPU Bus: Demultiplexed and fully buffered, TTL compatible, Address, Data & Control signals are available on two 50-pin ribbon cable connectors.

Parallel I/O:48 Programmable parallel I/O lines (TTL compatible) through two 26-pin ribbon cable connectors.
Serial I/O:RS 232 C through on-board 9pin D-type female connector.
Power supply: +5V @ 3.0Am

CONFIGURATION AND INSTALLATION

Configuration of ESA 86/88-2 :
ESA 86/88-2 micro computer trainer kit is a versatile and can be configured in a number of ways, as determined by the setting of a DIP switch and other jumpers.

Operational mode selection

ESA 86/88-2 can be operated either in the serial mode or in Hexadecimal keypad mode. In the serial mode, the trainer kit is connected to a CRT terminal or to a host computer system (like PC compatible) through an RS 232 C interface. In the keypad mode, the trainer is operated through Hexadecimal keypad.

SW4 of the DIP switch	Operational mode
OFF	Serial mode
ON	Hexadecimal keypad mode*

(*Factory installed Option)

Printer Enable/Disable

ESA 86/88-2 firmware includes the driver program for centronics compatible parallel printer interface. This driver can be enabled/ disabled as shown below:

SW5 of the DIP Switch	Printer Driver
OFF	Disabled*
ON	Enabled

.
(*Factory installed Option)

Baud rate selection

In the serial mode of operation, ESA 86/88E configures an 8251A USART as follows:

- Asynchronous mode
- 8-bit character length
- 2 stop bits
- No parity
- Baud rate factor of 16X

Timers 0 of an 8253 provide the Transmit and receive baud clocks for the USART. This timer is initialized by the system firmware to provide proper baud clock based on the settings of the DIP Switch as shown below.

DIP SWITCH

SW3	**SW2**	**SW1**	**Baud rate**
OFF	OFF	ON	9,600

Memory selection:

ESA 86/88-2 has four sockets, labeled U9, U8, U7, U6 for RAM. These sockets are configured for 62256(32X 4) devices. Two of these sockets are populated (providing 64K Bytes of RAM) and two are for user expansion.

DEVICE	**DIP SWITCH**		**JUMPER**
	SW7	**SW6**	
27256	ON	OFF	JP10 – 1-2

Hexadecimal keypad Legend Interpretation

Hexadecimal code	**Acronym**	**Name**	**Acronym**	**Name**
EB/AX	EB	Examine Byte	AX	Accumulator
ER/BX	ER	Examine Register	BX	Base Register
GO/CX	GO	Go	CX	Count Register
ST/DX	ST	Single Step	DX	Data Register
IB/SP	IB	Input Byte	SP	Stack Pointer
OB/BP	OB	Output Byte	BP	Base Pointer
MV/SI	MV	Move	SI	Source Index
EW/DI	EW	Examine Word	DI	Destination-index
IW/CS	IW	Input word	CS	Code Segment
OW/DS	OW	Output Word	DS	Data Segment
A/SS	None	N/A	SS	Stack Segment
B/ES	None	N/A	ES	Extra Segment
C/IP	None	N/A	IP	Instruction-Pointer
D/FL	None	N/A	FL	Flag Register
E	None	N/A	None	N/A
F	None	N/A	None	N/A

Summary of Monitor commands

Command Group	Command	Function/Format
Examine/modify	Examine Byte	Displays/modifies memory byte locations **EB <address>,[[<data>] NEXT or PREV].**
	Examine word	Displays/modifies memory word locations **EW <address>,[[<data>],]NEXT of PREV].**
	Examine Register	Displays/modifies processor register contents **ER<reg key>[[<data>] NEXT] [.]**
Input/Output	Input Byter	Displays the data byte at the input port. **IB <port address> NEXT [NEXT].**
	Input word	Displays the data word at the input port. **IW <port address> NEXT [NEXT].**
	Output byte	outputs the data byte to the output port. **OB <port address>NEXT <data> [NEXT <data>]**
	Output word	outputs the data word to the output port. **OW <port address>NEXT <data> [NEXT<data>].**
Execution	Step	Executes one single instruction. **ST [<Start address>NEXT [[<start address>] NEXT]**
	Go	Transfers control from monitor to user program **GO [<address>] [NEXT<breakpoint address>].**
Block Move	Move	Moves block of data within memory **MV <start address> NEXT <end address> NEXT <destination address>.**

EXAMINE BYTE AND EXAMINE WORD COMMANDS

Function: The Examine byte (EB) and Examine Word (EW) commands are used to examine the contents of selected memory locations. In a memory location the contents can be modified in RAM.

Format

EB <address> NEXT [[<data>] PREV/NEXT].
EW <address> NEXT [[<data>] PREV/NEXT].

Operation

1. Both the commands operate in a similar fashion. To use these commands, press the EB key or EW key when prompted for a command.
2. When either key is pressed, a dot appears at the right edge of the address field indicating that an address entry is required.
3. Enter the memory address of the byte (for EB) of word (for EW) to be Examined.
4. After entering the address value, press the "," key. (i.e. the NEXT key).
5. The data byte of word contents of the addressed memory location will be displayed in the data field and a decimal point (a dot) appears at the right edge of data field indicating that the data can be updated. Note that when using the Examine word command, the byte contents of the displayed updated memory location appear in the two least-significant digits of the field and the byte contents of the next consecutive memory location (i.e. entered memory address + 1) appear in the two most significant digits of the data field.
6. If the contents of the memory location addressed are only to be examined, press the ''.'' Key to terminate the command, of press the '','' (NEXT) key to examine the next consecutive memory location (Examine Byte Command) or the next two consecutive memory locations (Examine Word Command) or press the ''PREV'' key to examine previous byte or word location.
7. To modify the contents of an addressed memory location, enter the new data from the hexadecimal keyboard.
8. The data displayed is not updated in memory until either the '','' or ''.'' Key is pressed.

EXAMINE REGISTER COMMAND

Function

The Examine Register (ER) command is used to examine and optionally modify the contents of the 8086/8088's registers.

Format: ER <reg key> [[<data>],] [.]

INPUT/OUTPUT COMMANDS

There are 4 commands available for Input/output of Byte/Word data form/to a specified port. In entering the port address (in any of these four commands), it should be noted that 8086/8088 I/O addressing is limited to 64K (maximum address is FFFFH). Thus no segment value is permitted with the port address.

INPUT BYTE AND INPUT WORD COMMANDS

Function

The Input Byte (IB) and Input Word (IW) commands are used to input (accept) an 8-bit byte or 16-bit word from an input port.

Format

IB <port address >, [,].

IW <port address>, [,].

STEP COMMAND

FUNCTION

This command is used for single step execution of a program. In other words, this step (ST) command permits program instructions in memory to be executed individually. With each instruction executed, control is returned to the monitor from the program being executed,

Format

ST [< start address>], [[<start address>,.

GO COMMAND

Function

The GO command is used to transfer control of the 8086/8088 from the keyboard monitor program to user's program.

Format

GO [<address>] [, <breakpoint address>]

MOVE COMMAND

Function

This command (MV) can be used to move a block of data from one portion of the memory to another potion of the memory by specifying the address locations.

Format

MV <Start address> NEXT <end address> NEXT <destination address>.

II.MASM – INTRODUCTION

MICROSOFT MACRO ASSEMBLER:

A program called assembler is used to convert the mnemonics of the instructions along with the data into equivalent object code modules, these object code modules may further be converted into executable code using the linker and loader programs. This type of programming is called assembly level programming. The assembler is a program that converts an assembly input file also called source file to object file that can be converted into machine codes or an executable file using linker.

The assembly level programming is done using MASM and TASM, which along with it uses a link program to structure the codes operated by MASM in the form of an executable file. It reads source program as an input and provides an object file. The LINK file accepts the .obj file produced by MASM as an input and produces an .exe file.

ENTERING A PROGRAM:

The programs are generally entered using the text editor and we use 'turboc' as editor. For every assembly language program .asm extension must be there. The MASM accepts the file name only with extension of .asm .Even if the file name is without the extension it provides an extension of .asm to it.

SYNTAX: MASM filename.asm;

On the next line the expected .obj file name is to be on the next line entered which creates the objects of the assembly language program. The .obj file is allocated with the entered file name and the .obj extension on the next line a file name is entered and an extension for the expected listing file of the source file in the same way as the object file is entered as an extension of .lst exists. The successful assembly process may generate the .obj, .lst and .crf files which would further be used by linker to generate an .exe file.

LINKING A PROGRAM:

The dos linking program LINK.EXE is used by MASM to link one different object modules of the source program and function library routines to generate an integrated executable code of the source program. The main input to the linker is the .obj file that contains the object modules of the source program. Other supporting information is obtained by the files generated by MASM. The linker program is invoked using the following syntax.

SYNTAX: LINK filename.obj;

The output of the link program is an executable file whose extension is generated as .exe. MASM 5.0 uses the complete procedure of assembling and linking under a single menu invoked compile function.

DEBUGGING A PROGRAM:

It is a DOS utility that facilitates the debugging and trouble shooting of the assembly language programs. The debug utility enables us to have the control of the resources. The debug command at the DOS prompt invokes this facility. A '-'display signals the successful invoke operation to the debug that is further used as debug prompt for entering the various debugging commands as listed:

***COMMAND CHARACTER FUNCTIONS*:**

-R: display all the registers and flags.
-D: display 128 memory locations of RAM
-E: enter hexadecimal data at current display pointer.
-F: fill memory area byte by byte
-a: assemble from current CS: IP
-u: assemble from current CS: IP
-g: execute from current CS: IP
-s: searches a byte or string of bytes
-q: quit the debug
-t: trace the program execution step by step.
-m: trace the move and bytes from offset1
-n: set file name pointer to filename.
-l: loads the filename.exe in the ram.

The TASM software information is included in Appendix.

PROGRAMS
Experiment:1

1.1 AIM: Write an assembly language program to add two 16-bit numbers using various addressing modes.

EQUIPMENT REQUIRED: Personal Computer, O.S: WIN-XP Software: Macro assembler (ESA/MASM) Version: 5.0

PROGRAM:

```
        ASSUME CS:CODE,DS:DATA
        DATA SEGMENT
        DATA1 DW 4444H
        RESULT1 DW 01H DUP(0)
        RESULT2 DW 01H DUP(0)
        RESULT3 DW 01H DUP(0)
        DATA ENDS
        CODE SEGMENT
START:  MOV AX,DATA
        MOV DS,AX
        MOV AX,1234H
        MOV BX,1234H
        ADD AX,BX           ;addition between two
                             16-bit  numbers
                      using  register addressing mode
        MOV RESULT1,AX
        ADD AX,2244h        ;addition using immediate
                             Addressing mode
        MOV RESULT2,AX
        MOV BX,0000H
        ADD AX,DATA1[BX]    ;Register indirect  A.M
        MOV RESULT3,AX
        INT 03H
        CODE ENDS
        END START
```

RESULT:

```
-G
    AX=8AF0  BX=0000 CX=0031 DX=0000 SP=0000
    BP=0000    SI=0000   DI=0000
   -D DS:0
0B7F : 0000  44  44  68  24  AC   46   F0  8A   -00
```

Viva Questions: 1.What are the segment registers in 8086MP?
2.what is the diff b/w ADD and ADC instructions?
3.What is the diff b/w CS and DS registers?
4 .What is the function of INT03 instuction?

Algorithm for 16 bit addition:

Step1: start the program
Step2: Declare data segment and code segment
Step3: In the data segment declare the two operands to perform addition and one variable for storing the result
Step4: Take the two operands in Ax and Bx registers
Step5: add two numbers
Step6: store the result which is in the Ax register in to the variable which is declared in the data segment
Step7: end the program

Experiment:1.2
16-Bit Subtraction using various addressing modes

1.2 AIM: Write an assembly language program to subtract two 16-bit numbers using various addressing modes.

EQUIPMENT REQUIRED: Personal Computer, O.S: WIN-XP Software: Macro assembler (ESA/MASM) Version: 5.0

```
ASSUME CS:CODE,DS:DATA
 DATA SEGMENT
  DATA1 DW 4444H
  RESULT1 DW 01H DUP(0)
  RESULT2 DW 01H DUP(0)
  RESULT3 DW 01H DUP(0)
  RESULT4 DW 01H DUP(0)
  DATA ENDS
  CODE SEGMENT
  START:MOV AX,DATA
      MOV DS,AX
      MOV AX,1234H
      MOV BX,1234H
      SUB AX,BX     ;subtraction between two 16-bit numbers using
                      register  addressing mode

      MOV RESULT1,AX;
      MOV AX,0444H
      SUB AX,0222H              ;subtraction using immediate
                                        addressing mode
      MOV RESULT2,AX
      MOV AX,8888H
      SUB AX,DATA1[0000]            ;Direct A.M
      MOV RESULT3,AX

      MOV AX,9999H
      MOV BX,0000H
      SUB AX,DATA1[BX]              ; Indirect A.M
      INT 03H
      CODE ENDS
      END START
```

RESULT:

-G

AX=5555 BX=0000 CX=003E DX=0000 SP=0000 BP=0000 SI=0000 DI=0000

-D DS:0

13DD:0000 44 44 00 00 22 02 44 C4

Viva Questions: 1.What are the segment registers in 8086MP?
2.what is the diff b/w SUB and SBB instructions?
3.What is the diff b/w CS and DS registers?
4 .What is the function of INT 03 instuction?

Algorithm for 16 bit subtraction

Step1: start the program
Step2: Declare data segment and code segment
Step3: In the data segment declare the two operands to perform subtraction and one variable for storing the result
Step4: Take the two operands in AX and BX registers
Step5: subtract two numbers i.e, (AX – BX)
Step6: store the result which is in the AL register in to the variable which is declared in the data segment
Step7: end the program

Experiment:1.3
16-Bit Multiplication using various addressing modes

1.3 AIM: Write an assembly language program for multiplication of two 16-bit numbers

EQUIPMENT REQUIRED: Personal Computer, O.S: WIN-XP Software: Macro assembler (ESA/MASM) Version: 5.0

PROGRAM:

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
A DW 0002H
B DW 1234H
MUL1 DW 01H DUP(0)
MUL2 DW 01H DUP(0)
MUL3 DW 01H DUP(0)
DATA ENDS
CODE SEGMENT
START:MOV AX,DATA
    MOV DS,AX
    MOV AX,A
    MOV BX,B
    MUL BX          ; register addressing mode
    MOV MUL1,AX
    MOV AX,0004H
    MUL B[0000H]    ; direct addressing mode
    MOV MUL2,AX
    MOV AX,0006H
    MOV BX,0000H
    MUL B[BX]       ;  indirect addressing mode
    MOV MUL3,AX
INT 03H
CODE ENDS
END START
```

RESULT:

-G

AX=6D38 BX=0000 CX=0039 DX=0000 SP=0000BP=0000
SI=0000 DI=0000 DS=13DD

-D DS:0

13DD:0000 02 00 34 12 68 24 D0 48-38 6D 00 00 00

Questions:

1.What are the processer control instructions?
2.what is diff b/w HALT and NOP instructions?
3.What is function of XLAT instruction?

Algorithm for multiplication of two 16-Bit numbers using various addressing modes:

Step1: start the program
Step2: Declare data segment and code segment
Step3: In the data segment declare the two operands to perform multiplication and two variable for storing the result
Step4: Take the two operands in AX and BX registers
Step5: Multiply two numbers i.e., (AX * BX)
Step6: store the result which is in the AX and DX register in to the variables which are declared in the data segment
Step7: end the program

Experiment:1.4
16-Bit Division using various addressing modes

1.4 AIM: Write an assembly language program to perform division between two 16-bit numbers

EQUIPMENT REQUIRED: Personal Computer, O.S: WIN-XP Software: Macro assembler (ESA/MASM) Version: 5.0

PROGRAM:

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
A DW 1234H
B DW 0002H
DIV1QUO DW 01H DUP(0)
DIV1REM DW 01H DUP(0)
DIV2QUO DW 01H DUP(0)
DIV2REM DW 01H DUP(0)
DIV3QUO DW 01H DUP(0)
DIV3REM DW 01H DUP(0)
DATA ENDS
CODE SEGMENT
START:MOV AX,DATA
    MOV DS,AX
    MOV AX,A
    MOV BX,B
    DIV BX; REGISTER ADDRESSING MODE
    MOV DIV1QUO,AX
    MOV DIV1REM,DX
    MOV AX,44F3H
    DIV B[0000H]; DIRECT ADDRESSING MODE
    MOV DIV2QUO,AX
    MOV DIV2REM,DX
    MOV AX,022FH
    MOV BX,0000H
    DIV B[BX];  INDIRECT ADDRESSING MODE
    MOV DIV3QUO,AX
    MOV DIV3REM,DX
INT 03H
CODE ENDS
END START
```

RESULT:

```
-G
    AX=8117 BX=0000 CX=0045 DX=0001 SP=0000 BP=0000
    SI=0000 DI=0000 DS=13DD
    -D DS:0
13DD:0000      34 12 02 00 1A 09 00 00-79 22 01 00 17 81 01 00
```

Questions: **1.What are the processer control instructions?**
2.what is diff b/w HALT and NOP instructions?
3.What is function of XLAT instruction?

Algorithm for division of two 16-Bit numbers using various addressing modes:

Step1: start the program
Step2: Declare data segment and code segment
Step3: In the data segment declare the two operands to perform division and two variables for storing the result
Step4: Take the two operands in AX and BX registers
Step5: Multiply two numbers i.e., (AX / BX)
Step6: store the result which is in the AX and DX register in to the variables which are declared in the data segment (quotient in AX, remainder in DX)
Step7: end the program

EXPERIMENT:2
Experiment:2.1

2.1. AIM: Write an assembly language program for addition of two array numbers (16-bit numbers).

EQUIPMENT REQUIRED: Personal Computer, O.S: WIN-XP Software: Macro assembler (ESA/MASM) Version: 5.0

PROGRAM:

```
ASSUME CS: CODE, DS: DATA
 DATA SEGMENT
 A DW 000CH,000FH,0005H
 B DW 0002H,0002H,0005H
RESULT DW 04H DUP(0)
DATA ENDS
CODE SEGMENT
START:MOV AX,DATA
MOV DS,AX
MOV AX,0000H
MOV BX,0000H
MOV CX,0003H
MOV DX,0000H
AGAIN:MOV AX,A[BX]
ADD AX,B[BX]
MOV RESULT[BX],AX
INC BX
INC BX
DEC CX
JNZ AGAIN
INT 03H
CODE ENDS
END START
```

RESULT:

-G

AX=000A BX=0006 CX=0000 DX=0000 SP=0000 BP=0000
SI=0000 DI=0000 DS=13DD

-D DS:0

13DD:0000 0C 00 0F 00 05 00 02 00-02 00 05 00 0E 00 11 00
13DD:0010 0A 00 00 00 00 00 00 00

Questions: **1.What are the processer control instructions?**
2.what is diff b/w HALT and NOP instructions?
3.What is function of XLAT instruction?

Algorithm for addition of two arrays

Step1: Start the program
Step2: Declare data segment and code segment
Step3: In the data segment declare the two arrays to perform addition and one array for storing the result and the count for multiple additions.
Step4: Take the Offset address of result array in the BX register and count in the CX register
Step5: Move the first operand of first array in the AX register
Step6: Add the value of AX with the first operand of second array
Step7: Store the result in the memory location which is present in the BX register and in the AX register take the next operand and point to the next operand of 2^{nd} array by incrementing BX register.
Step8: Decrement the value of CX register, if the value of CX is not equal to zero then go to STEP4
Step9: End the program.

EXPERIMENT:2
Experiment:2.2

2.2 AIM: Write an assembly language program for subtraction of two array numbers (16-bit numbers).

EQUIPMENT REQUIRED: Personal Computer, O.S: WIN-XP Software: Macro assembler (ESA/MASM) Version: 5.0

PROGRAM:

```
        ASSUME CS:CODE,DS:DATA
        DATA SEGMENT
        A DW 000CH,000FH,0005H
        B DW 0002H,0002H,0005H
        RESULT DW 04H DUP(0)
        DATA ENDS
        CODE SEGMENT
        START:MOV AX,DATA
            MOV DS,AX
            MOV AX,0000H
            MOV BX,0000H
            MOV CX,0003H
            MOV DX,0000H
      AGAIN:MOV AX,A[BX]
            SUB AX,B[BX]
            MOV RESULT[BX],AX
            INC BX
            INC BX
            DEC CX
            JNZ AGAIN
        INT 03H
        CODE ENDS
        END START
```

RESULT:

```
-G
    AX=0000 BX=0006 CX=0000 DX=0000 SP=0000 BP=0000
    SI=0000   DI=0000  DS=13DD
-D DS:0
13DD:0000   0C 00 0F 00 05 00 02 00-02 00 05 00 0A 00 0D 00
13DD:0010   00 00 00 00 00 00 00 00
```

Questions: **1.What are the processer control instructions?**
2.what is diff b/w HALT and NOP instructions?
3.What is function of XLAT instruction?

Algorithm for subtraction of two arrays

Step1: Start the program
Step2: Declare data segment and code segment
Step3: In the data segment declare the two arrays to perform subtraction and one array for storing the result and the count for multiple subtractions.
Step4: Take the Offset address of result array in the BX register and count in the CX register
Step5: Move the first operand of first array in the AX register
Step6: Subtract the value of AX with the first operand of second array
Step7: Store the result in the memory location which is present in the BX register and in the AX register take the next operand and point to the next operand of 2^{nd} array by incrementing BX register.
Step8: Decrement the value of CX register, if the value of CX is not equal to zero then go to STEP4
Step9: End the program.

Experiment:2.3

2.3 AIM: Write an assembly language program for multiplication of two array numbers (16-bit numbers).

EQUIPMENT REQUIRED: Personal Computer, O.S: WIN-XP Software: Macro assembler (ESA/MASM) Version: 5.0

PROGRAM:

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
A DW 000CH,000FH,0005H
B DW 0002H,0002H,0005H
RESULT DW 04H DUP(0)
DATA ENDS
CODE SEGMENT
START:MOV AX,DATA
    MOV DS,AX
    MOV AX,0000H
    MOV BX,0000H
    MOV CX,0003H
    MOV DX,0000H
AGAIN:MOV AX,A[BX]
    MUL B[BX]
    MOV RESULT[BX],AX
    INC BX
    INC BX
    DEC CX
    JNZ AGAIN
INT 03H
CODE ENDS
END START
```

RESULT:

```
-G
    AX=0019 BX=0006 CX=0000 DX=0000 SP=0000 BP=0000
    SI=0000   DI=0000  DS=13DD
 -D DS:0
13DD:0000   0C 00 0F 00 05 00 02 00-02 00 05 00 18 00 1E 00
13DD:0010   19 00 00 00 00 00 00 00
```

Questions: **1.What is the importance of SI in 8086 mp?**
2.Which general purpose register is used as pointer?
3.What is the function of dx register?

Algorithm for Multiplication of two arrays

Step1: Start the program
Step2: Declare data segment and code segment
Step3: In the data segment declare the two arrays to perform multiplication and one array for storing the result and the count for multiple multiplications.
Step4: Take the Offset address of result array in the BX register and count in the CX register
Step5: Move the first operand of first array in the AX register
Step6: Multiply the value of AX with the first operand of second array
Step7: Store the result in the memory location which is present in the BX register and in the AX register take the next operand and point to the next operand of 2^{nd} array by incrementing BX register.
Step8: Decrement the value of CX register, if the value of CX is not equal to zero then go to STEP4
Step9: End the program.

Experiment:2.4

2.4 AIM: Write an assembly language program for division of two array numbers (16-bit numbers).

EQUIPMENT REQUIRED: Personal Computer, O.S: WIN-XP Software: Macro assembler (ESA/MASM) Version: 5.0

PROGRAM:

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
A DW 000CH,000FH
B DW 0002H,0002H
RESULT DW 0002H DUP(0)
REMINDER DW 0002H DUP(0)
DATA ENDS
CODE SEGMENT
START:MOV AX,DATA
    MOV DS,AX
    MOV AX,0000H
    MOV BX,0000H
    MOV CX,0002H
    MOV DX,0000H
AGAIN:MOV AX,A[BX]
    DIV B[BX]
    MOV RESULT[BX],AX
    MOV REMINDER[BX],DX
    INC BX
    INC BX
    DEC CX
    JNZ AGAIN
INT 03H
CODE ENDS
END START
```

RESULT:

```
-G
    AX=0007 BX=0004 CX=0000 DX=0001 SP=0000 BP=0000
    SI=0000   DI=0000   DS=13DD
 -D DS:0
13DD:0000   0C 00 0F 00 02 00 02 00-06 00 07 00 00 00 01 00
```

Algorithm for Division of two arrays

Step1: Start the program
Step2: Declare data segment and code segment
Step3: In the data segment declare the two arrays to perform division and one array for storing the result and the count for multiple divisions.
Step4: Take the Offset address of result array in the BX register and count in the CX register
Step5: Move the first operand of first array in the AX register
Step6: Divide the value of AX with the first operand of second array
Step7: Store the result in the memory location which is present in the BX register and in the AX register take the next operand and point to the next operand of 2^{nd} array by incrementing BX register.
Step8: Decrement the value of CX register, if the value of CX is not equal to zero then go to STEP4
Step9: End the program.

Experiment:2.5

2.5 AIM: Write an assemble language program to find the largest numbers in array

EQUIPMENT REQUIRED: Personal Computer, O.S: WIN-XP Software: Macro assembler (ESA/MASM) Version:

PROGRAM:

```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
A DW 0002H, 0001H, 0004H, 0003H
COUNT EQU 04H
B DW 01 DUP (0)
DATA ENDS
CODE SEGMENT
START: MOV AX, DATA
MOV DS, AX
MOV AX, 0000H
MOV BX, 0000H
MOV CX, COUNT
MOV AX, A[BX]
AGAIN: INC  BX
INC BX
CMP AX,A[BX]
JNL NEXT
MOV AX,A[BX]
NEXT:  LOOP AGAIN
MOV B, AX
INT 03H
CODE ENDS
END START
```

RESULT:

-G

```
AX=0004 BX=0008 CX=0000 DX=0000 SP=0000 BP=0000
SI=0000  DI=0000  DS=13DD
```

-D DS:0

```
13DD:0000  02 00 01 00 04 00 03 00-04 00 00 00 00 00 00 00 00
```

Questions:

1. What is the diff b/w CMP and SUB instructions

2. Which flag bit is effected when JNZ is Executed?

3. What is the diff b/w MOV and XCHG instructions?

Algorithm to find the largest numbers in array

Step1: Start the program
Step2: Declare data segment and code segment
Step3: In the data segment declare the array and the count.
Step4: Take the Offset address of result array in the BX register and count in the CX register
Step5: Move the content of the array to the AX register
Step 6: Increment the value of BX
Step 7: Compare the value of AX with the second operand of the array
Step8: Jump on not less to step 6
Step9: After the count is zero, all the comparisons are done and the largest value is stored in B.
Step9: End the program.

Experiment:2.6

2.6 AIM: Write an assembly language program to arrange numbers in an array in descending order.

EQUIPMENT REQUIRED: Personal Computer, O.S: WIN-XP Software: Macro assembler (ESA/MASM) Version: 5.0

PROGRAM:

```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
LIST DW 02H, 01H, 04H, 03H
COUNT EQU 04H
DATA ENDS
CODE SEGMENT
START: MOV AX, DATA
MOV DS, AX
MOV DX, COUNT- 1
AGAIN0: MOV CX, DX
MOV SI, OFFSET LIST
AGAIN1: MOV AX, [SI]
CMP AX, [SI+2]
JL PRL
XCHG [SI+2], AX
XCHG [SI], AX
PRL:  ADD SI, 02
LOOP AGAIN1
DEC DX
JNZ AGAIN0
INT 03
CODE ENDS
END START
```

RESULT:

-G

```
AX=0001 BX=0000 CX=0000 DX=0000 SP=0000 BP=0000
SI=0006   DI=0000   DS=13DD
```

-D DS:0

```
13DD:0000   01 00 02 00 03 00 04 00-00 00 00 00 00 00 00 00 00
```

Question: **1.What is SAL and SAR instructions?**
2.What is SHR and SHL instructions?

Algorithm to arrange a given series of numbers in descending order

Step1: Start the program
Step2: Declare data segment and code segment
Step3: In the data segment declare the array of numbers and count
Step4: Move the count value in DX register
Step6: move the offset address of array in the SI register
Step7: From the offset address that is present in the SI register copy the data into AX register
Step8: compare the value present in AX with the data present in the memory location SI+2
Step9: If AX is less than the value present in memory location SI+2, go to step 11
Step10: exchange the value of AX with the value present in the memory location SI+2
Step11: add two to SI register
Step12: decrement the value of DX, if it is not equal to zero, and then go to step5
Step13: end the program

Experiment:3

3.0 AIM: Write an assembly language program for searching for a number or character in string

EQUIPMENT REQUIRED: Personal Computer, O.S: WIN-XP
Software: Macro assembler (ESA/MASM) Version: 5.0

PROGRAM:

```
ASSUME CS:CODE,DS:DATA ,ES:EXTRA
DATA SEGMENT
STRING1 DB 0AH,0DH,"ENTER A
   CHARACTER:",0AH,0DH,"$"
STRE DB "FOUND$"
STRNE DB "NOT FOUND$"
DATA ENDS
EXTRA SEGMENT
STRING2 DB "CMRCET$"
STRLEN DW ($-STRING2)
EXTRA ENDS
CODE SEGMENT
START:MOV AX,DATA
MOV DS,AX
MOV AX,EXTRA
MOV ES,AX
MOV DX,OFFSET STRING1
MOV AH,09H
INT 21H
MOV AH,08H
INT 21H
REABLE
MOV DX,OFFSET STRNE
MOV AH,09H
INT 21H
MOV AH,4CH
INT 03H
JMP EXIT
LABLE:MOV DX,OFFSET STRE
MOV AH,09H
INT 21H PNE SCASB
JZ L
MOV AH,4CH
INT 03H
EXIT: INT 03H
```

```
CODE ENDS
END START
```

RESULT: ENTER A CHARACTER: 'C'
FOUND

Experiment:4

4.1 AIM: Write an assembly language program for block transfer of a given string from data segment to extra segment

EQUIPMENT REQUIRED: Personal Computer, O.S: WIN-XP Software: Macro assembler (ESA/MASM) Version: 5.0

PROGRAM:

```
ASSUME CS: CODE, DS:DATA, ES:EXTRA
DATA SEGMENT
SRCDATA DB "MPMC LAB $"
COUNT EQU 08H
DATA ENDS
EXTRA SEGMENT
DSTDATA DB 12 DUP (0)
EXTRA ENDS
CODE SEGMENT
START: MOV AX, DATA
MOV DS, AX
MOV AX, EXTRA
MOV ES, AX
MOV SI, OFFSET SRCDATA
MOV DI, OFFSET DSTDATA
CLD
REP MOVSB
MOV AH,09H
INT 21H
MOV AH,4CH
INT 03H
CODE ENDS
END START
```

RESULT:

```
-G
   AX=4C24 BX=0000 CX=0000 DX=0000
   D DS:0
   0B40:0000   4D 50 4D 43 20 4C 42 20 24 MPMC LAB $
  D ES:0
   0C40:0010   4D 50 4D 43 20 4C 42 20 24 MPMC LAB $
```

Experiment:4.2

4.2 AIM: Write an assembly language program for reverse of a given string

EQUIPMENT REQUIRED: Personal Computer, O.S: WIN-XP
Software: Macro assembler (ESA/MASM) Version: 5.0

PROGRAM:

```
ASSUME CS: CODE, DS: DATA, ES: EXTRA
DATA   SEGMENT
STRING1   DB "MPMC LAB $"
STRLEN EQU ($-STRING1)
DATA ENDS
EXTRA SEGMENT
STR2 DB 01 DUP (0)
EXTRA ENDS
CODE SEGMENT
START:  MOV AX, DATA
MOV DS, AX
MOV AX, EXTRA
MOV ES, AX
MOV BX, OFFSET STRING1
MOV SI, BX
MOV DI, OFFSET STRING2
ADD DI, STRLEN-1
CLD
MOV CX, STRLEN
A1: MOV AL, [SI]
MOV ES:[DI], AL
INC SI
DEC DI
LOOP A1
MOV AH,09H
INT 21H
MOV AH,4CH
INT 03H
CODE ENDS
END START
```

RESULT:

-G

AX=4C24 BX=0000 CX=0000 DX=0000

D DS:0

0B40:0000 4D 50 4D 43 20 4C 41 42 20 24 42 41 4C 20 43 40 MPMC LAB $

BAL CMPM

Experiment:4.3

4.3 AIM: Write an assembly language program for comparison of two strings

EQUIPMENT REQUIRED: Personal Computer, O.S: WIN-XP
Software: Macro assembler (ESA/MASM) Version: 5.0

PROGRAM:

```
ASSUME CS:CODE,DS:DATA,ES:EXTRA
DATA SEGMENT
STRING1 DB 'EMPTY$'
STRLEN EQU ($-STRING1)
NOTSFUL DB 'STRINGS ARE UNEQUAL$'
SFUL DB 'STRINGS ARE EQUAL$'
DATA ENDS
EXTRA SEGMENT
STRING2 DB 'EMPTY$'
EXTRA ENDS
CODE SEGMENT
START:MOV AX,DATA
MOV DS,AX
MOV AX,EXTRA
MOV ES,AX
MOV CX,STRLEN
CLD
MOV SI,OFFSET STRING1
MOV DI,OFFSET STRING2
REP CMPSB
JZ FORW
MOV AH,09H
MOV DX,OFFSET NOTSFUL
INT 21H
JMP EXITP
FORW:MOV AH,09H
MOV DX,OFFSET SFUL
INT 21H
EXITP:
NOP
MOV AH,4CH
INT 21H
CODE ENDS
END START
```

RESULT: AX=4C24 BX=0000 STRINGS ARE EQUAL

Experiment:4.4

4.4 AIM: Write an assembly language program to display string on monitor

EQUIPMENT REQUIRED: Personal Computer, O.S: WIN-XP
Software: Macro assembler (ESA/MASM) Version: 5.0

PROGRAM:

```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
MSG DB  "MPMCLAB$"
DATA ENDS
CODE SEGMENT
START:MOV AX,DATA
MOV DS,AX
MOV DX,OFFSET MSG
MOV AH,09H
INT 21H
MOV AH,4CH
INT 03H
CODE ENDS
END START
```

RESULT: MPMCLAB

```
-G
  AX=4C24  BX=0000 CX=001F  DX=0000
  D DS:0
  0B40:0000   4D 50 4D 43  4C 41 42 24              MPMCLAB$
```

Experiment:5
INTERFACING

5. AIM: Write an assembly language program to Design a Digital Clock.

EQUIPMENT REQUIRED: 8086 kit, power supply, RS-232 cable, RTC Interface Module

PROGRAM TO SETRTC:

```
OUTPUT 2500AD
ORG 2400H
MOV AX, 0000H
MOV ES, AX
MOV BL,0H
MOV SI, 2300H       ; initialize pointer
MOV DX,0FFE6H   ;initialize all 8255 ports
MOV AL,80H        ;as output
OUT DX,AL
MOV DX,FFE4H      ;hold line high
MOV AL,10H
OUT DX,AL
MOV AL,28H        ;hold setup time delay
HDLY: DEC AL
JNE HDLY

SET: MOV DX,0FFE2H    ;output address to
RTC
MOV AL,BL
OUT DX,AL
NOP
NOP
MOV DX,0FFE0H ;get rtc parameter from
MOV AL,[SI]       ;memory and output
OUT DX,AL         ;to rtc
MOV DX,0FFEH   ;make write&hold state
MOV AL,50H       ;high
OUT DX,AL
NOP
NOP
MOV AL,10H        ;make write signal low
OUT DX,AL         ;keeping hold high
INC BL            ;increment address count
```

```
CMP BL,0DH        ;all parameters over?
JE STOP           ;if yes , stop
INC SI            ;else point to next parameter
JMP SHORT SET     ;and repeat
STOP: MOV AL,00H  ;hold and write low
OUT DX,AL
INT 3
END
```

Program to READRTC:

```
; This program continuously reads RTC values and output the same
; on the PC console
; first execute the setrtc program then execute this program
; to get rtc values.
; This program can be in Serial mode only

        OUTPUT 2500AD
        ORG 2000H
         MOV AX,0000H
         MOV CS,AX
         MOV ES,AX
         MOV DS,AX
         JMP START

;Display Message Strings

    MES: DB 0AH,0AH,0DH,48H,52H,53H,20H,20H,20H,20H
         DB 4DH,49H,4EH,20H,20H,20H,20H,53H,45H,43H
         DB 20H,20H,20H,20H,41H,2FH,50H,20H,20H,20H
         DB 20H,20H,44H,41H,59H,20H,20H,20H,20H,20H
         DB 44H,44H,2DH,4DH,4DH,2DH,59H,59H

   LINE: DB 0AH,0DH,2DH,2DH,2DH,20H,20H,20H,20H,2DH
         DB 2DH,2DH,20H,20H,20H,20H,2DH,2DH,2DH,20H
         DB 20H,20H,20H,2DH,2DH,2DH,20H,20H,20H,20H
         DB 20H,2DH,2DH,2DH,20H,20H,20H,20H,20H,2DH
         DB 2DH,2DH,2DH,2DH,2DH,2DH,2DH,0AH,0DH,00H

  SPACE: DB 20H,20H,20H,20H,20H,00H
  HYP: DB 2DH,00H
  AM: DB 41H,4DH,20H,00H,00H
  PM: DB 50H,4DH,20H,00H
```

```
WEEK: DB 53H,55H,4EH,00H
 DB 4DH,4FH,4EH,00H
 DB 54H,55H,45H,00H
 DB 57H,45H,44H,00H
 DB 54H,48H,52H,00H
 DB 46H,52H,49H,00H
 DB 53H,41H,54H,00H
 DB 20H,20H,20H,00H

 BAK: DB 08H,00H
 NEW: DB 0DH,00H

 START: MOV SI,3300H          ;initialise pointer
        CALL READ             ;get rtc values
        MOV DX,0FFE6H         ;initialise 8255 port A
        MOV AL,90H            ;as I/P , port B as O/P
OUT DX,AL
        CS:
        LEA DX,MES            ;display message for rtc
        MOV SI,DX             ;parameters
        CALL FAR 0FE00:01AFH  ;Display message and line
        REPT: LEA DX,NEW
        MOV SI,DX
        CALL FAR 0FE00:01AFH
        MOV BX,3304H          ;get hours value
        CALL DAT
        AND AL,3FH
        CALL FAR 0FE00:0052H  ;Display Hours
        LEA DX,SPACE
        MOV SI,DX
        CALL FAR 0FE00:01AFH
        MOV BX,3302H          ;get min. value
        CALL DAT
        CALL FAR 0FE00:0052H  ;Display Minutes
        MOV SI,DX
        CALL FAR 0FE00:01AFH
        MOV BX,3300H          ;get seconds
        CALL DAT
        CALL FAR 0FE00:0052H  ;Display seconds
        MOV SI,DX
        CALL FAR 0FE00:01AFH
        MOV BX,2305H          ;check for 12/24
        MOV AL,[BX]           ;hrs format
```

```
 AND AL,08H
JNE DD
MOV AL,[BX]              ;check for AM or PM
AND AL,04H               ;if bit2=1 display PM
JNE PM1
LEA DX,AM
MOV SI,DX
CALL FAR 0FE00:01AFH
JMP SHORT MM

PM1: LEA DX,PM           ;routine to display PM
MOV SI,DX
CALL FAR 0FE00:01AFH
JMP SHORT MM

DD: LEA DX,HYP           ;if 24 hour format
MOV SI,DX                ;display PM
CALL FAR 0FE00:01AFH
MOV SI,DX
CALL FAR 0FE00:01AFH
MOV SI,DX
CALL FAR 0FE00:01AFH

MM: LEA DX,SPACE
MOV SI,DX
CALL FAR 0FE00:01AFH
MOV AX,00H
MOV BX,3306H             ;get day of week
MOV AL,[BX]
AND AL,07H
LEA DX,WEEK              ;display day of week

CHK: CMP AL,00H
JE DISP
MOV CX,04H

INCR: INC DX
LOOP INCR
DEC AX
JMP SHORT CHK

DISP: MOV SI,DX
CALL FAR 0FE00:01AFH
```

```
        LEA DX,SPACE
        MOV SI,DX
        CALL FAR 0FE00:01AFH
        MOV BX,3307H            ;get date value
        CALL DAT
        CALL FAR 0FE00:0052H    ;display date
        LEA DX,HYP
        MOV SI,DX
        CALL FAR 0FE00:01AFH
        MOV BX,3309H            ;get month value
        CALL DAT
        CALL FAR 0FE00:0052H    ;display month
        MOV SI,DX
        CALL FAR 0FE00:01AFH
        MOV BX,330BH            ;get year value
        CALL DAT
        CALL FAR 0FE00:0052H    ;display year
        CALL REWRT              ;check for update seconds

       CLEAR: PUSH CX           ;routine to clear old value
       MOV CX,0029H

       BACK: LEA DX,BAK
       MOV SI,DX
       CALL FAR 0FE00:01AFH
       LOOP BACK
       POP CX
       JMP REPT

      REWRT: MOV BX,3300H       ;routine to check updating
      MOV AL,[BX]               ;of seconds value in
      PUSH AX                   ;location 2300H
      MOV SI,3300H
CALL DELAY
     CALL READ                  ;repeat read and display
     POP AX                     ;operation
     MOV BX,3300H
     CMP AL,[BX]
     JE REWRT
     RET

     READ: MOV DX,FFE4H         ;hold line high
     MOV AL,10H
```

```
OUT DX,AL
MOV AL,28H              ;hold setup time delay

HDLY: DEC AL
JNE HDLY
MOV AL,30H              ;read and hold line high
OUT DX,AL
MOV CL,00H              ;clear rtc reg. counter

NEXT: MOV DX,FFE2H          ;output address to rtc
MOV AL,CL
OUT DX,AL
MOV AL,04H              ;read access time delay
RDLY: DEC AL
JNE RDLY
MOV DX,FFE0H              ;read data from rtc
IN AL,DX
AND AL,0FH              ;AND with 0Fh and store
MOV [SI],AL             ;in memory
INC SI                ;increment address and rtc reg.
INC CL
CMP CL,0DH              ;are all reg. addressed?
JNE NEXT              ;no, get next parameter
MOV DX,0FFE4H ;else make hold and read lines low
MOV AL,00H
OUT DX,AL
RET

 DAT:  MOV AL,[BX]              ;routine to club the lower
AND AL,0FH            ;nibbles of the memory
MOV AH,AL             ;locations
INC BX
MOV AL,[BX]
AND AL,0FH
ROL AL,1
ROL AL,1
ROL AL,1
ROL AL,1
OR AL,AH
RET
```

```
        DELAY:   MOV  AX,1FFH
        DLY:     DEC  AX
      JNE  DLY
      RET
      END
```

RESULT:

DIGITAL CLOCK MODULE IS INTERFACED WITH 8086 KIT AND VERIFIED OUTPUT

EXPERIMENT:6

6.1.1 AIM: Write an assembly language program to generate square wave form.

EQUIPMENT REQUIRED: 8086 kit, DAC kit, Power supply, RS-232 cable, FRC cable, Interface card.

PROGRAM:

```
OUTPUT 2500AD
ORG 2000H
START:  MOV DX,0FFE6H
MOV AL,80H
OUT DX,AL
RPT2:MOV AX,FFH
MOV DX,0FFE0H
OUT DX,AL
MOV DX,0FFE2H
OUT DX,AL
MOV CX,FFH
DELAY1:LOOP DELAY1
MOV AX,00H
MOV DX,0FFE0H
OUT DX,AL
MOV DX,0FFE2H
OUT DX,AL
MOV CX,FFH
DELAY2:LOOP DELAY2
JMP SHORT RPT2
END
```

RESULT:

Digital to analog conversion has done and square wave generated.

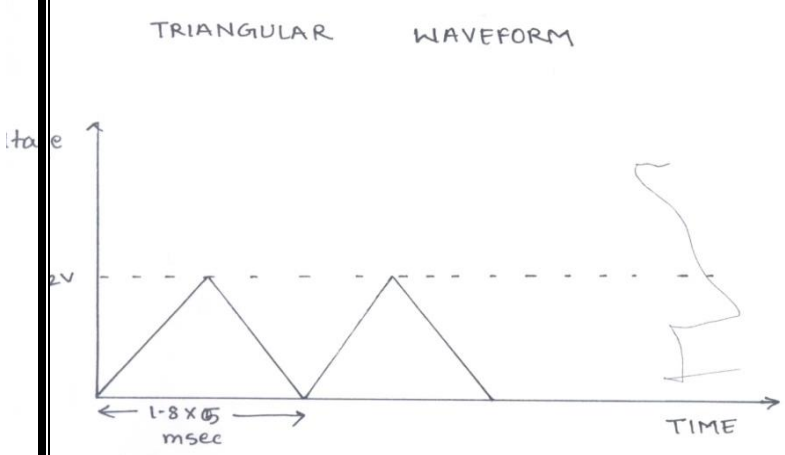
EXPERIMENT:6.2

6.2 AIM: Write an assembly language program to generate triangular wave form

EQUIPMENT REQUIRED: 8086 kit,DAC kit,power supply,RS-232 cable, FRC cable,Interface card.

PROGRAM:

```
OUTPUT 2500AD
ORG 2000H
START:MOV DX,0FFE6H
MOV AL,80H
OUT DX,AL
RPT1:MOV CX,0FFH
MOV AL,00H
UP:INC AL
MOV DX,0FFE0H
OUT DX,AL
MOV DX,0FFE2H
OUT DX,AL
LOOP UP
MOV CX,0FFH
DOWN:DEC AL
 MOV DX,0FFE0H
OUT DX,AL
MOV DX,0FFE2H
OUT DX,AL
LOOP DOWN
JMP SHORT RPT1
END
```


TRIANGULAR WAVEFORM
2V
1·8 x 5
msec
TIME

EXPERIMENT:6.3

6.3 AIM: Write an assembly language program to convert analog voltage to digital voltage.

EQUIPMENT REQUIRED: 8086 kit, ADC kit, power supply, RS-232 cable, FRC cable, Interface card.

PROGRAM:

```
        OUTPUT 2500AD
        ORG  2000H
         MOV  AX,0000H
         MOV  ES,AX
         CALL FAR 0FE00:0031H
         MOV  DX,0FFE6H
         MOV  AL,82H
         OUT  DX,AL
         JMP SHORT  START
         MES:          DB 0DH, 'DIGITAL VALUE =   ',0H
         START:        CALL FAR 0FE00:01CEH
         LEA   DX,MES
         MOV  AX,DX
         CALL FAR 0FE00:0013H
         STC:       MOV  AL,02H
         MOV  DX,0FFE0H
         OUT  DX,AL
         NOP
         NOP
         NOP
         MOV  AX,01H
         OUT  DX,AL
         MOV  CX,1000H
         KK:        LOOP KK
         MOV  AL,04H
         OUT  DX,AL
         MOV  DX,0FFE0H
         MOV  BL,00H
         TVAR:   MOV  DX,0FFE2H
          IN   AL,DX
          AND  AL,01H
          JE   DISP
          INC  BL
          JMP SHORT  TVAR
    DISP:MOV  AL,BL
```

```
CALL FAR 0FE00:0052H
MOV  AL,02H
MOV  DX,0FFE0H
OUT  DX,AL; routine to introduce delay between
                        Successive conversions
MOV  BX,03H
DY:MOV  CX,01FFFH
SS:LOOP SS
DEC  BX
JNZ  DY
JMP SHORT  START          ;repeat continuously
END
```

EXPERIMENT:7

7. AIM: Write an assembly language program to develop a Parallel Communication between two 8086 Microprocessor kits using PPI(8255)

EQUIPMENT REQUIRED: 8086 Microprocessor trainer kits
FRC Cables
+5V power supply

PROGRAM:

KIT 1:
```
MOV DX,0FFE6
MOV AL,80H
OUT DX,AL
MOV DX,0FFE0
MOV AL,55H
OUT DX,AL
INT 03
```

KIT 2:
```
MOV DX,0FFE6
MOV AL,90H
OUT DX,AL
MOV DX,0FFE0
IN  AL,DX
INT 03
```

RESULT:

Kit 1:	**Kit 2:**
G 4000	G 4000
AX=0055	AX=0955

EXPERIMENT:8

8. AIM: Serial communication between two microprocessor kits using 8251
EQUIPMENT REQUIRED: 8086 Microprocessor trainer kit
+5V power supply, RS232 cable

PROGRAM

```
OUTPUT 2500AD
DSEG SEGMENT
ORG 4000H
DATABUFFER DS 10
DSEG ENDS
CSEG SEGMENT
ASSUME CS:CSEG,DS:DSEG
ORG 5000H
CMD51 EQU 0FFF2H
DATA51 EQU 0FFF0H
CALL INIT8251
MOV CX,08
 RPTTX: MOV AH,41H
CALL SRLOUT
LOOP RPTTX
MOV AH,50H
CALL SRLOUT
INT 3
SRLOUT: MOV DX,CMD51
IN AL,DX
AND AL,04H
JZ SRLOUT
MOV AL,AH
MOV DX,DATA51
OUT DX,AL
RET

ORG 6000H
RECV:
CALL INIT8251
MOV BX,OFFSET DATABUFFER
RPTRX:
CALL SRLIN
MOV [BX],AL
INC BX
CMP AL,31H
JNZ RPTRX
```

```
 INT 3
 SRLIN: MOV DX,CMD51
 IN AL,DX
AND AL,02H
JZ SRLIN
MOV DX,DATA51
OUT DX,AL
NOP
MOV AL,65H
OUT DX,AL
MOV CX,0FFH
L1: LOOP L1
NOP
MOV AL,25H
OUT DX,AL
NOP
MOV CX,0FFH
L2: LOOP L2
MOV AL,65H
OUT DX,AL
NOP
MOV CX,0FFH
L3: LOOP L3
MOV AL,0CFH
OUT DX,AL
NOP
MOV CX,0FFH
L4: LOOP L4
MOV AL,25H
OUT DX,AL
NOP
MOV CX,0FFH
L5: LOOP L5
RET
CSEG ENDS
END
```

EXPERIMENT:9

9. AIM: Write an assembly language program to rotate single phase permanent magnet stepper motor in:

1).Clock wise direction.

2).Anti clock wise direction.

EQUIPMENT REQUIRED: 8086 kit,Stepper motor kit,power supply,RS-232 cable, FRC cable,Interface card.

Principle:A stepper motor is a device need to obtain an accurate control of rotating shafts. A stepper motor employ's rotation of its shafts.A motor used for moving things in small increment is known as a stepper motor. It rotates from one fixed position to another fixed position. They are also used to read write head on the desired track of a floppy disk. To rotate the shaft of a stepper motor a sequence of pulses are applied to the windings in a predefined sequence. So the number of pulses required for complete rotation on its rotor. The rotation per pulse is given by 360°/NT Where NT is the no of teeth on the rotor. Generally they are available in 10° to 30° rotations. They are available with 2ϕ and 4ϕ a common field connections.

Unlike most of the motors this rotates in steps from one fixed position to next common step sizes range from 0.9° to30°. It is stepped from one fixed position to next by changing the currents through the field in the motor. The two common connections are referred to as two phases and four phases.

The switch pattern for x changing from one step to step in clock wise direction is simply rotates one position right from counter clock wise direction it is rotated one position left.

With a pulse applied to the winding input, the rotor rotates by one teeth position at an angle 'x',the angle 'x'given by

$$X=360°/\text{no. of teeth}$$

$$\text{No. of count}=\frac{\text{no. of teeth}}{360°}\times 1.8°$$

CLOCKWISE DIRECTION:

```
OUTPUT 2500AD
ORG 2000H
START:MOV DX,0FFE6H;
MOV AL,80H
OUT DX,AL
MOV AL,11H
MOV DX,0FFE0H
MOV BX,C8H
R1:OUT DX,AL
CALL DELAY
RCR AL,01
DEC BX
JMP R1
DELAY:MOV CX,800H
SS:LOOP SS
RET
END
```

ANTI CLOCKWISE DIRECTION :

```
OUTPUT 2500AD
ORG 2000H
START:MOV DX,0FFE6H
MOV AL,80H
OUT DX,AL
MOV AL,11H
MOV DX,0FFE0H
R1:OUT DX,AL
CALL DELAY
RCL AL,01
JMP R1
DELAY:MOV CX,800H
SS:LOOP SS
RET
END
```

RESULT:

STEPPER MOTOR IS INTERFACED TO 8086 & MOTOR ROTATION IS CONTROLLED IN CLOCKWISE AND ANTI CLOCKWISE DIRECTIONS.

EXPERIMENT:10

10.1 AIM: Write an assembly language program to perform Addition of two numbers present in a data memory locations 9000 and 9001.Store the result in 9002 Data Memory?

EQUIPMENT REQUIRED: 8051 Micro Controller Hardware kit
5V Power Supply
Keyboard

PROGRAM:

```
MOV  DPTR,#9000
MOVX  A,@DPTR
MOV B,A
MOV DPTR,#9001
MOVX A,@DPTR
ADD  A,B
MOV DPTR,#9002
MOVX  @DPTR,A
LCALL 0003
```

RESULT:

INPUT:	**OUTPUT:**
MD 9000 02	MD 9002 06
9001 04	

EXPERIMENT:10.2

10.2 AIM: Write an assembly language program to perform logical AND operation of two numbers present in a data memory locations 9000 and 9001.Store the result in 9002 Data Memory?

EQUIPMENT REQUIRED: 8051 Micro Controller Hardware kit
5V Power Supply
Keyboard

PROGRAM:

```
MOV  DPTR,#9000
MOVX  A,@DPTR
MOV B,A
MOV DPTR,#9001
MOVX A,@DPTR
ANL  A,B
MOV DPTR,#9002
MOVX  @DPTR,A
LCALL 0003
```

RESULT:

INPUT:
MD 9000 05
9001 06

OUTPUT:
MD 9002 04

EXPERIMENT:10.3

10.3 AIM: Write an assembly language program to perform Addition of two numbers present in a data memory locations 9000 and 9001.Store the result in 9002 Data Memory?

EQUIPMENT REQUIRED: 8051 Micro Controller Hardware kit
5V Power Supply
Keyboard

PROGRAM:

```
MOV  DPTR,#9000
MOVX  A,@DPTR
MOV B,A
MOV DPTR,#9001
MOVX A,@DPTR
ADD  A,B
MOV DPTR,#9002
MOVX  @DPTR,A
LCALL 0003
```

RESULT:

INPUT:
MD 9000 08
9001 02

OUTPUT:
MD 9002 0A

EXPERIMENT:10.4

10.4 AIM: Write an assembly language program to perform Subtraction of two numbers present in a data memory locations 9000 and 9001.Store the result in 9002 Data Memory?

EQUIPMENT REQUIRED: 8051 Micro Controller Hardware kit
5V Power Supply
Keyboard

PROGRAM:

```
MOV  DPTR,#9000
MOVX  A,@DPTR
MOV B,A
MOV DPTR,#9001
MOVX A,@DPTR
SUBB  A,B
MOV DPTR,#9002
MOVX  @DPTR,A
LCALL 0003
```

RESULT:

INPUT:	**OUTPUT:**
MD 9000 08	MD 9002 02
9001 06	

EXPERIMENT:10.5

10.5 AIM: Write an assembly language program to perform Multiplication of two numbers present in a data memory locations 9000 and 9001.Store the result in 9002 Data Memory?

EQUIPMENT REQUIRED: 8051 Micro Controller Hardware kit
5V Power Supply
Keyboard

PROGRAM:

```
MOV  DPTR,#9000
MOVX  A,@DPTR
MOV B,A
MOV DPTR,#9001
MOVX A,@DPTR
MUL  AB
MOV DPTR,#9002
MOVX  @DPTR,A
LCALL 0003
```

RESULT:

INPUT:	**OUTPUT:**
MD 9000 08	MD 9002 10
9001 02	

EXPERIMENT:10.6

10.6 AIM: Write an assembly language program to perform Logical OR Operation of two numbers present in a data memory locations 9000 and 9001.Store the result in 9002 Data Memory?

EQUIPMENT REQUIRED: 8051 Micro Controller Hardware kit
5V Power Supply
Keyboard

PROGRAM:

```
MOV  DPTR,#9000
MOVX  A,@DPTR
MOV B,A
MOV DPTR,#9001
MOVX A,@DPTR
ORL  A,B
MOV DPTR,#9002
MOVX  @DPTR,A
LCALL 0003
```

RESULT:

INPUT:	**OUTPUT:**
MD 9000 05	MD 9002 07
9001 06	

EXPERIMENT:11

11. AIM: Write an assembly language program to verify timer or counter in 8051.

EQUIPMENT REQUIRED:

8051 Micro Controller Hardware kit
5V Power Supply
Keyboard

SOFTWARE :KEIL

PROGRAM:

```
MOV TMOD,#01          ;Timer 0, mode 1(16-bit mode)
HERE: MOV TL0,#0F2H   ;TL0=F2H, the low byte
MOV TH0,#0FFH         ;TH0=FFH,  the high byte
CPL  P1.5             ;  toggle P1.5
ACALL  DELAY
SJMP  HERE

DELAY:
SETB TR0              ;start the timer 0
AGAIN: JNB TF0,AGAIN  ; monitor timer flag 0
                      ;until it rolls over
CLR TR0 ;stop timer 0
CLR TF0 ;clear timer 0 flag
RET
```

Result: An assembly language program to verify timer or counter in 8051 is verified

EXPERIMENT:12

12 AIM: Write an assembly language program to verify interrupt handling in 8051.

EQUIPMENT REQUIRED:

8051 Micro Controller Hardware kit
5V Power Supply
Keyboard

SOFTWARE :KEIL

PROGRAM:

```
        ORG 0000H
        LJMP MAIN
              ;--ISR for hardware interrupt INT1 to turn on led
        ORG 0013H                ;INT1 ISR
        SETB P1.3                ;turn on LED
        MOV R3,#255
        BACK: DJNZ R3,BACK   ;keep the buzzer on for a while
        CLR P1.3                 ;turn off the buzzer
        RETI                     ;return from ISR
                          ;MAIN program for initialization
       ORG 0030H
       MAIN: SETB TCON.2     ;make INT1 edge-triggered int.
       MOV IE,#10000100B       ;enable External INT 1
       HERE: SJMP HERE         ;stay here until get interrupted
       END
```

RESULT: THE INTERRUPT HANDLING MECHANISM IS VERIFIED IN 8051.

EXPERIMENT:13

13 AIM: Write an assembly language program to verify UART operation in 8051 microcontroller.

EQUIPMENT REQUIRED:

8051 Micro Controller Hardware kit
5V Power Supply
Keyboard

SOFTWARE : KEIL

PROGRAM:

```
ORG 8000H
MOV SCON,#50H
MOV TMOD,#20H
MOV TH1,#0FDH
SETB TR1
MOV A,#"H"
MOV SBUF,A
L1:JNB TI,L1
CLR TI
LCALL 03H
END
```

RESULT: UART operation in 8051 microcontroller is verified.

EXPERIMENT:14

14. AIM: Write an assembly language program to make the Communication between 8051 kit and PC.

EQUIPMENT REQUIRED:

8051 Micro Controller Hardware kit
5V Power Supply
Keyboard

SOFTWARE :KEIL/MASM

PROGRAM:

```
ORG 8000H
MOV DPTR,#STRING
LCALL 11E0H
LJMP 0
STRING: DB 'MPMCLAB',00H
```

RESULT: Communication between 8051 kit and PC is verified.

EXPERIMENT:15

15. AIM: Write an assembly language program for interfacing LCD to 8051

EQUIPMENT REQUIRED: System: CORE2DUO, O.S: WIN-XP,8051kit
SOFTWARE: Xassembler (X8051)

PROGRAM

```
CMD_PORT    EQU   03H
PORT_A      EQU   00H
PORT_B      EQU   01H
GETKB       EQU   142H
GETCH       EQU   12A5H

        ORG   8000H
        ACALL INIT
        MOV   A,#80H
        CALL  CMD
        MOV   DPTR,#DISP1
        MOV   R3,#08H
        CALL  STRING
        MOV   A,#0C0H
        CALL  CMD
        MOV   DPTR,#DISP2
        MOV   R3,#08H
        CALL  STRING

START:  MOV   R6,#80H
        MOV   DPTR,#0E102H
        MOVX  A,@DPTR
        JB    ACC.3,KBD
        CLR   P3.4
        CALL  GETCH
        SJMP  DISP
KBD:    CALL  GETKB
DISP:   PUSH  A
        MOV   A,#01H
        CALL  CMD
        MOV   A,#80H
        CALL  CMD
        POP   A
        CALL  DWR
```

```
DELAY2: MOV    R3,#01H
L1:    MOV    R1,#0FFH
CALL   DELAY
DJNZ   R3,L1
MOV    A,#0C0H
CALL   CMD
MOV    DPTR,#MESS
MOV    R3,#08H
CALL   STRING
MOV    A,#02H
CALL   CMD
JMP    START

INIT:  MOV    A,#80H
MOV    P2,#0E8H
R0,#03H
MOVX   @R0,A
MOV    R1,#1FH
CALL   DELAY
MOV    R3,#03H
INIT1: MOV    A,#38H
CALL   CMD
MOV    R1,#3FH
CALL   DELAY
DJNZ   R3,INIT1
MOV    A,#38H
CALL   CMD
MOV    A,#0CH
CALL   CMD
MOV    A,#01H
CALL   CMD
RET

;COMMAND WRITE SUBROUTINE

CMD:   MOV    P2,#0E8H
MOV    R0,#PORT_A
MOVX   @R0,A
MOV    A,#0FBH
MOV    R0,#PORT_B
MOVX   @R0,A
MOV    A,#0F8H
MOV    R0,#PORT_B
```

```
        MOVX   @R0,A
        MOV    A,#0FCH
        MOV    R0,#PORT_B
        MOVX   @R0,A
        MOV    A,#0F8H
        MOV    R0,#PORT_B
        MOVX   @R0,A
        MOV    R1,#10H
        CALL   DELAY
        RET

       ;DATE WRITE SUBROUTINE

        DWR:   MOV    P2,#0E8H
        MOV    R0,#PORT_A
        MOVX   @R0,A
        MOV    A,#0FAH
        MOV    R0,#PORT_B
        MOVX   @R0,A
        MOV    A,#0F9H
        MOV    R0,#PORT_B
        MOVX   @R0,A
        MOV    A,#0FDH
        MOV    R0,#PORT_B
        MOVX   @R0,A
        MOV    A,#0F9H
        MOV    R0,#PORT_B
        MOVX   @R0,A
        MOV    R1,#10H
        CALL   DELAY
        RET
:DELAY SUBROUTINE

        DELAY: MOV    R2,#02H
        DLY:DJNZ   R2,DLY
        DJNZ   R1,DLY
        RET

;STRING WRITE SUBROUTINE

        STRING: CLR    A
        MOVC   A,@A+DPTR
        JZ     L3
```

```
CALL    DWR
INC     DPTR
DJNZ    R3,STRING
L3:     RET

MESS:   DB      'IS TYPED',00H
DISP1:  DB      'WAITING',00H
DISP2:  DB      'FOR KEY',00H

END
```

EXPERIMENT:16

16. AIM: Write an assembly language program for interfacing keyboard to 8051

EQUIPMENT REQUIRED: System: CORE2DUO, O.S: WIN-XP,8051kit
SOFTWARE: Xassembler (X8051)

PROGRAM:

```
OUTPUT      EQU   03FAH
PUTBYTE     EQU   139EH
SEG         EQU   0E8H
ORG   8000H
MOV   DPTR,#MESG
 LCALL  OUTPUT
 BACK:  LCALL  DILIT ;BACK
 LCALL  KSCAN
 MOV    A,R4
 MOV    71H,A
 LCALL  PUTBYTE
 SJMP   BACK
 KSCAN: MOV    P2,#0E8H
 MOV    R0,#03H
 MOV    A,#92H
 MOVX   @R0,A
 KSCN:  MOV    R3,02H
 MOV    R4,#10H
 MOV    R1,#04H
 NXTGRP: MOV   A,R1
 MOV    R0,#02H
 MOVX   @R0,A
 RRC    A
 MOV    R1,A
 MOV    R0,#00H
 MOVX   A,@R0
 JNZ    NXTKEY
 MOV    A,R4
 SUBB   A,#08H
 MOV    R4,A
 DEC    R3
 MOV    A,R3
 DJNZ   R3,NXTGRP
 SJMP   KSCN        ;jump to kscn
```

```
NXTKEY: RRC A ;rotate acc right through carry flag
JNC     NEXT      ;jump if no carry to  next
RET
NEXT: XCH A,R4 ;Exchange acc with r4 data
ADD     A,#01H     ; add 01h to acc
XCH     A,R4   ;exchange acc with r4 data
SJMP    NXTKEY     ;jump to nxtkey
DILIT:  MOV     DPTR,#BACKSP
LCALL   OUTPUT
RET
MESG:   DB 'KEY PRESSED=  ',00H
BACKSP: DB 08H,08H,00H
DLY:    MOV     R6,#FFH
DLY1:   MOV     R7,#FFH
DJNZ    R7,$
DJNZ    R6,DLY1
RET
```

RESULT: If key N is typed from the keyboard then it is displayed on the display as “N is Typed”.

EXPERIMENT:17

17. AIM: Write an assembly language program to interface DMA.

EQUIPMENT REQUIRED: 8086 kit, DMA controller kit, power supply,RS-232 cable, FRC cable, Interface card.

PROGRAM:

PROGRAM FOR READ MODE:

```
MOV  AL,84H
OUT  38,AL
MOV AL,00H
OUT 34H,AL
MOV AL,40H
OUT 34H,AL
MOV AL,0AH
OUT 35,AL
MOV AL,80H
OUT 35H,AL
CALL C0E6H
JMP B814H
```

PROGRAM FOR WRITE MODE:

```
MOV  AL,48H
OUT  38,AL
MOV AL,00H
OUT 36H,AL
MOV AL,40H
OUT 36H,AL
MOV AL,0AH
OUT 37,AL
MOV AL,40H
OUT 37H,AL
CALL  C0CCH
JMP  B7FAH
```

RESULT: DATA TRANSFERRED BY USING DMA AND DISPLAYED BY USING LEDS

IV.Programs beyond the syllabus

ALP to convert BCD to ASCII

1. AIM: Write an assembly language to convert BCD to ASCII?

EQUIPMENT REQUIRED: System: p4, O.S: WIN-98
Software: Macro assembler (ESA/MASM) Version:5.0

```
PROGRAME:    DATA SEGMENT
             BCDIP     DB   56H
             DUBCDOP   DW   0
             DATA ENDS

             CODE SEGMENT
             ASSUME  CS:CODE,DS:DATA
             START:    MOV AX,DATA
                       MOV DS,AX
                       XOR AX,AX
                       MOV AL, BCDIP
                       MOV DL,AL
                       AND AL,0F0H
                       MOV CL,4
                       ROR AL,CL
                       MOV BH,AL
                       AND DL,0FH
                       MOV BL,DL
                       MOV UBCDOP,BX
                       ADD BX,3030H
                       MOV AH,4CH
                       INT 21H
                       CODE ENDS
                       END START
```

RESULT:

INPUT:	**OUTPUT:**
AX=0056	AX=3536

ALP TO FIND FACTORIAL OF A GIVEN NUMBER

2. AIM: Write an assembly language program to factorial of a given number?

EQUIPMENT REQUIRED: System: p4, O.S: WIN-98

SOFTWARE: Macro assembler (ESA/MASM) Version: 5.00

PROGRAME:

```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
C DB  04H
B DB 01 DUP (0)
DATA ENDS
CODE SEGMENT
START: MOV AX, DATA
MOV DS, AX
MOV AX, 0001H
MOV CX, C
NEXT: MUL CX
LOOP NEXT
MOV  B,AX
INT 03
CODE ENDS
END START
```

RESULT:

-G

```
AX=0018 BX=0000 CX=0000 DX=0000 SP=0000 BP=0000
SI=0000 DI=0000
DS: 1098 ES=1088 SS=1098 CS=1099 IP=0013
-D DS:0   04 18
-U CS:0
```

ALP TO ADD TWO NUMBERS BY USING DOS INTERRUPTS

3.AIM: Write an assembly language program to add two numbers by using DOS Interrupts?

EQUIPMENT REQUIRED: System: p4, O.S: WIN-98
SOFTWARE: Macro assembler (ESA/MASM) Version: 5.00

PROGRAM:

```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
MSG1 DB 0AH,0DH,'ENTER THE FIRST NUMBER:  $'
MSG2 DB 0AH,0DH,'ENTER THE SECOND NUMBER:
$'
MSG3 DB 0AH,0DH,'RESULT:  $'
RES DB 00
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:  MOV AX,DATA
 MOV DS,AX
 LEA DX,MSG1
 MOV AH,09H
 INT 21H
 MOV AH,01H
 INT 21H
 SUB AL,30H
 MOV BL,AL
 LEA DX,MSG2
 MOV AH,09H
 INT 21H
MOV AH,01H
 INT 21H
 SUB AL,30H
 ADD AL,BL
 ADD AL,30H
 MOV RES,AL
 LEA DX,MSG3
 MOV AH,09H
 INT 21H
 MOV DL,RES
 MOV AH,02H
 INT 21H
 MOV AH,4CH
 INT 21H
```

```
        CODE ENDS
        END START
```

RESULT:

ENTER THE FIRST NUMBER:04
ENTER THE SECOND NUMBER:02
RESULT :06

ALP TO FIND THE NUMBER OF POSITIVE& NEGATIVE NUMBERS IN A GIVEN SERIES.

4. **AIM:** Write an assembly language program to find the number of positive& negative numbers in a given series.

EQUIPMENT REQUIRED: System: p4, O.S: WINDOWS 98
SOFTWARE: Macro assembler (ESA/MASM) Version: 5.00

PROGRAM:

```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
LIST DB 12H, -11H, 13H, -20H, 25H,-14H
COUNT EQU 07H
DATA ENDS
CODE SEGMENT
START: MOV AX, DATA
MOV DS, AX
XOR BX, BX
XOR DX, DX
MOV CL, COUNT
MOV SI, OFFSET LIST
AGAIN:  MOV AX,[SI]
SHL AX, 01H
 JC NEG 1
 INC BX
 JMP NEXT
 NEG 1:  INC DX
 NEG 2:   ADD SI, 02
DEC CL
JNZ AGAIN
INT 03
CODE ENDS
END START
```

RESULT:

```
-G
   AX=4CDC  BX=0004  CX=0003  DX=0003  SP=0000  BP=0000
SI=0000DI=0000
DS=0000 ES=11B9 CS=11CA IP=001C
   -D DS:0
   -U  CS:0
```

Question: 1.What is SAL and SAR instructions?
2.What is SHR and SHL instructions?

ALP TO FIND EVEN & ODD NUMBERS IN A GIVEN SERIES.

5.AIM: Write an assembly language program to find even &odd in a given series.

EQUIPMENT REQUIRED: System: p4, O.S: WIN-98
SOFTWARE: Macro assembler (ESA/MASM) Version: 5.00

PROGRAM:

```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
 LIST DB 12H, 11H, 13H, 20H, 25H, 18H
 COUNT EQU 06H
 DATA ENDS
 CODE SEGMENT
 START:  MOV AX, DATA
 MOV DS, AX
 XOR BX, BX
 XOR DX, DX
 MOV CL, COUNT
 MOV SI, OFFSET LIST
 AGAIN:  MOV AX, [SI]
 ROR AX, 01H
 JC NODD
 INC BX
 JMP NEXT
 NODD:    INC DX
 NEXT:   ADD SI, 02
 DEC CL
 JNZ AGAIN
 INT 03
 CODE ENDS
 END START
```

RESULT:

```
-G
    AX=4C03 BX=0003 CX=0003 DX=0003 SP=0000 BP=0000
    SI=0006 DI=0000 DS=11C9 ES=11B9 SS=11C9 CS=11CA
    IP=001C
   -D DS:0  12 11 13 20 25 18 06
    -U  CS:0
```

Questions: 1.What are the diff types of rotate instructions?
2.Diffrence b/w rotate and shift instructions?
3.What is the diff b/w RCR and ROR instructions?