Government Girls' Polytechnic, Bilaspur

Name of the Lab: **Digital Electronics Lab**

Practical: **Advanced Microprocessor & Microcontroller Lab**

Class :**VI Semester (ET&T)**

Teachers Assessment: 10 End Semester Examination: 40

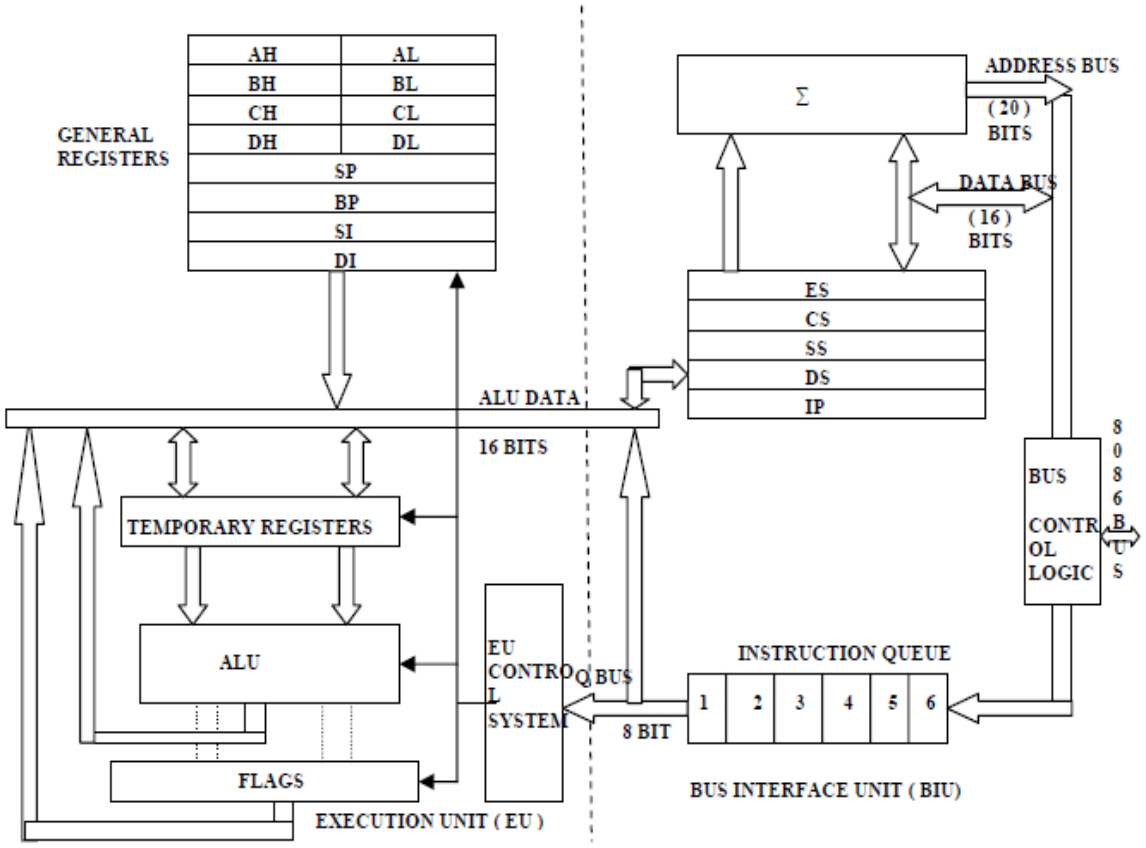
EXPERIMENT NO 1

Object:-Identify different components/unit in 8086 kit.

Theory:-8086 Microprocessor

- It is a 16bit microprocessor.
- 8086 has a 20 bit address bus can access upto 2^20 memory location(1MB).
- It can support upto 64K I/o ports.
- It provides 14, 16-bit registers.
- It has multiplexed address and data bus AD0-AD15 and A16-A19.
- It requires single phase clock with 33% duty cycle to provide internal timing.
- 8086 designed to operate in two modes, minimum mode and maximum mode.
- It can prefatchedupto 6 instruction bytes from memory & queues them in order to speed up instruction execution.
- It requires +5 volt power supply.
- A 40 pin dual in line package.

KIT DIAGRAM OF 8086 MICROPROCESSOR



Block Diagram of 8086

COMPONENTS OF 8086 MICROPROCESSOR

Registers

Accumulator:-or A register is an 8-bit register used for arithmetic, logic, I/O and load/store operations.

Flag Register:-has five 1-bit flags.

• **Sign** - set if the most significant bit of the result is set.
• **Zero** - set if the result is zero.
• **Auxiliary carry** - set if there was a carry out from bit 3 to bit 4 of the result.
• **Parity** - set if the parity (the number of set bits in the result) is even.
•**Carry**-set if there was a carry during addition, or borrow during subtraction/comparison/rotation.

General Registers

- 8-bit B and 8-bit C registers can be used as one 16-bit BC register pair. When used as a pair the C register contains low-order byte. Some instructions may use BC register as a data pointer.
- 8-bit D and 8-bit E registers can be used as one 16-bit DE register pair. When used as a pair the E register contains low-order byte. Some instructions may use DE register as a data pointer.
- 8-bit H and 8-bit L registers can be used as one 16-bit HL register pair. When used as a pair the L register contains low-order byte. HL register usually contains a data pointer used to reference memory addresses.

• **Stack pointer:-**is a 16 bit register. This register is always decremented/incremented by 2 during push and pop.

• **Program counter:-**is a 16-bit register.

- Instruction Set
    - 8085 instruction set consists of the following instructions:
    - Data moving instructions.
    - Arithmetic - add, subtract, increment and decrement.
    - Logic - AND, OR, XOR and rotate.
    - Control transfer - conditional, unconditional, call subroutine, return from subroutine and restarts.
    - Input/Output instructions.

- Other - setting/clearing flag bits, enabling/disabling interrupts, stack operations, etc.

Addressing mode

- **Register** - references the data in a register or in a register pair. Register indirect - instruction specifies register pair containing address, where the data is located. Direct, Immediate - 8 or 16-bit data.

Instruction Set

- 8085 instruction set consists of the following instructions:
- Data moving instructions.
- Arithmetic - add, subtract, increment and decrement.
- Logic - AND, OR, XOR and rotate.
- Control transfer - conditional, unconditional, call subroutine, return from subroutine and restarts.
- Input/Output instructions.
- Other - setting/clearing flag bits, enabling/disabling interrupts, stack operations,

Address/Data Bus:- these lines serve two functions. As an address bus is 20 bits long and consists of signal lines A0 through A19. A19 represents the MSB and A0 LSB. A 20bit address gives the 8086 a 1Mbyte memory address space. More over it has an independent I/O address space which is 64K bytes in length.
•The 16 data bus lines D0 through D15 are actually multiplexed with address lines A0 through A15 respectively. By multiplexed we mean that the bus work as an address bus during first machine cycle and as a data bus during next machine cycles. D15 is the MSB and D0 LSB.
•When acting as a data bus, they carry read/write data for memory, input/output data for I/O devices, and interrupt type codes from an interrupt controller.

Status signal:-
The four most significant address lines A19 through A16 are also multiplexed but in this case with status signals S6 through S3. These status bits are output on the bus at the same time that data are transferred over the other bus lines.
•Bit S4 and S3 together from a 2 bit binary code that identifies which of the 8086 internal segment registers are used to generate the physical address that was output on the address bus during the current bus cycle.
•Code S4S3 = 00 identifies a register known as ***extra segment register*** as the source of the segment address.

DMA Interface signals:-
The direct memory access DMA interface of the 8086 minimum mode consist of the HOLD and HLDA signals.
•When an external device wants to take control of the system bus, it signals to the 8086 by switching HOLD to the logic 1 level. At the completion of the current bus cycle, the 8086 enters the hold state. In the hold state, signal lines AD0 through AD15, A16/S3 through A19/S6, BHE, M/IO, DT/R, RD, WR, DEN and INTR are all in the high Z state. The 8086 signals external device that it is in this state by switching its HLDA output to logic 1 level.

Internal Architecture of 8086
•8086 has two blocks BIU and EU.

•The BIU performs all bus operations such as instruction fetching, reading and writing operands for memory and calculating the addresses of the memory operands. The instruction bytes are transferred to the instruction queue.
•EU executes instructions from the instruction system byte queue.
•Both units operate asynchronously to give the 8086 an overlapping instruction fetch and execution mechanism which is called as Pipelining. This results in efficient use of the system bus and system performance.
•BIU contains Instruction queue, Segment registers, Instruction pointer, Address adder.
•EU contains Control circuitry, Instruction decoder, ALU, Pointer and Index register, Flag register.

BUS INTERFACR UNIT:
•It provides a full 16 bit bidirectional data bus and 20 bit address bus.
•The bus interface unit is responsible for performing all external bus operations.

Specifically it has the following functions:
•Instruction fetch, Instruction queuing, Operand fetch and storage, Address relocation and Bus control.
•The BIU uses a mechanism known as an instruction stream queue to implement a ***pipeline architecture.***
•This queue permits prefetch of up to six bytes of instruction code. When ever the queue of the BIU is not full, it has room for at least two more bytes and at the same time the EU

is not requesting it to read or write operands from memory, the BIU is free to look ahead in the program by prefetching the next sequential instruction.
•These prefetching instructions are held in its FIFO queue. With its 16 bit data bus, the BIU fetches two instruction bytes in a single memory cycle.
•After a byte is loaded at the input end of the queue, it automatically shifts up through the FIFO to the empty location nearest the output.
•The EU accesses the queue from the output end. It reads one instruction byte after the other from the output of the queue. If the queue is full and the EU is not requesting access to operand in memory.
•These intervals of no bus activity, which may occur between bus cycles are known as ***Idle***

state.
•If the BIU is already in the process of fetching an instruction when the EU request it to read or write operands from memory or I/O, the BIU first completes the instruction fetch bus cycle before initiating the operand read / write cycle.
•The BIU also contains a dedicated adder which is used to generate the 20bit physical address that is output on the address bus. This address is formed by adding an appended 16 bit segment address and a 16 bit offset address.
•For example**:** The physical address of the next instruction to be fetched is formed by combining the current contents of the code segment CS register and the current contents of the instruction pointer IP register.
•The BIU is also responsible for generating bus control signals such as those for memory read or write and I/O read or write.

EXECUTION UNIT
The Execution unit is responsible for decoding and executing all instructions.
•The EU extracts instructions from the top of the queue in the BIU, decodes them, generates operands if necessary, passes them to the BIU and requests it to perform the read or write bys cycles to memory or I/O and perform the operation specified by the instruction on the operands.
•During the execution of the instruction, the EU tests the status and control flags and updates them based on the results of executing the instruction.
•If the queue is empty, the EU waits for the next instruction byte to be fetched and shifted to top of the queue.
•When the EU executes a branch or jump instruction, it transfers control to a location corresponding to another set of sequential instructions.
•Whenever this happens, the BIU automatically resets the queue and then begins to fetch instructions from this new location to refill the queue.
Module 1 and learning unit 4:

Signal Description of 8086•The Microprocessor 8086 is a 16-bit CPU available in different clock rates and packaged in a 40 pin CERDIP or plastic package.
•The 8086 operates in single processor or multiprocessor configuration to achieve high performance. The pins serve a particular function in minimum mode (single processor mode) and other function in maximum mode configuration (multiprocessor mode).
•The 8086 signals can be categorised in three groups. The first are the signal having common functions in minimum as well as maximum mode.
•The second are the signals which have special functions for minimum mode and third are the signals having special functions for maximum mode.

EXPERIMENT NO. 2

Aim: Execute a sample program

Apparatus required: 8051 Simulator / kit, PC, Power supply, connecting wires

Algorithm

1. Clear C reg. for carry
2. Get the data immediately.
3. Add the two data
4. Store the result in memory pointed by DPTR

Program

```
        ORG 4100
        CLR C
        MOV A # data1
        ADD A # data2
        MOV DPTR # 4500
        MOV X @ DPTR A
HERE:   SJMP HERE
```

Observation:

Input: 66

23

Output: 89 (4500)

Result: Thus the program to perform addition of two 8 bit numbers using 8051 instruction set was executed.

EXPERIMENT NO. 3

Aim: Develop 8051 assembly language programmes on trainer kit for addition of two 8 bit numbers.

Apparatus required: 8051 Simulator / kit, PC, Power supply, connecting wires

Algorithm

1. Clear C reg. for carry
2. Get the data immediately.
3. Add the two data
4. Store the result in memory pointed by DPTR

Program

```
        ORG 4100
        CLR C
        MOV A # data1
        ADD A # data2
        MOV DPTR # 4500
        MOV X @ DPTR A
HERE:   SJMP HERE
```

Observation:

Input: 66

23

Output: 89 (4500)

Result: Thus the program to perform addition of two 8 bit numbers using 8051 instruction set was executed.

EXPERIMENT NO. 4

Aim: Develop 8051 assembly language programmes on trainer kit for subtraction of two 8-bit numbers .

Apparatus required: 8051 Simulator / kit, PC, Power supply, connecting wires

Algorithm

1. Clear C reg. for carry
2. Get the data immediately.
3. Subtract the two data
4. Store the result in memory pointed by DPTR

Program

```
        ORG 4100

        CLR C

        MOV A # data1

        SUBB A # data2
        MOV DPTR # 4500

        MOV X @ DPTR A
HERE:   SJMP HERE
```

Observation:

Input: 66

23

Output: 43 (4500)

Result: Thus the program to perform Subtraction of two 8 bit numbers using 8051 instruction set was executed.

EXPERIMENT NO. 5

Aim: Develop 8051 assembly language programmes on trainer kit for multiplication of two 8-bit numbers/2 decimal numbers.

Apparatus required: 8051 Simulator / kit, PC, Power supply, connecting wires

Algorithm

1. Get the data in A Reg.
2. Get the data to be multiplied in B Reg..
3. Multiply the two data.
4. The higher order of the data is in B Reg.
5. The Lower order of the data is in A Reg.& Store the result.

Program

```
        ORG 4100
        CLR C
        MOV A # data1
        MOV B # data2
        MUL A B
        MOV DPTR # 4500
        MOV X @ DPTR A
        INC DPTR
        MOV A B
        MOVX @ DPTR,A
HERE:   SJMP HERE
```

Observation:

Input: 80

80

Output: 00 (4500)

19 (4501)

Result: Thus the program to perform Multiplication of two 8 bit numbers using 8051 instruction set was executed.

EXPERIMENT NO. 6

Aim: Develop 8051 assembly language programmes on trainer kit for Division of two 8-bit numbers/2 decimal numbers.

Apparatus required: 8051 Simulator / kit, PC, Power supply, connecting wires

Algorithm

1. Get the data in A Reg.
2. Get the data to be multiplied in B Reg..
3. Divide the two data.
4. The quotient is in A reg.
5. The remainder is in B Reg.& Store the result.

Program

```
        ORG 4100
        CLR C
        MOV A # data1
        MOV B # data2
        DIV A B
        MOV DPTR # 4500
        MOV X @ DPTR A
        INC DPTR
        MOV A B
        MOVX @ DPTR,A
HERE:   SJMP HERE
```

Observation:

Input: 05

03

Output: 01 (4500)

02 (4501)

Result: Thus the program to perform Division of two 8 bit numbers using 8051 instruction

EXPERIMENT NO. 7

Aim: study Interface ELC matrix display

Apparatus required: 8051 Simulator / kit, PC, Power supply, connecting wires

Theory: A matrix display interface for a presentation system, adapted to selectively display presentation data on a display interface when the presentation system is operated on a computer, the matrix display interface comprising: a plurality of data units, arranged on the display interface, and written with the presentation data in advance; a main switching element, having a first prompt text, for switching to display or conceal all the data units when clicked; at least one column switching element, having a second prompt text, for switching to display or conceal a whole column of the corresponding data units when clicked; at least one row switching element, having a third prompt text, for switching to display or conceal a whole row of the corresponding data units when clicked; and at least one grid switching element, corresponding to the data units, for switching to display or conceal the corresponding data units when clicked. matrix display interface comprising: a plurality of data units, arranged on the display interface, and written with the presentation data in advance; and at least one switching element, having a prompt text, for switching to display or conceal the corresponding data gle-row adjacent juxtaposition, single-row interval juxtaposition, multiple-row adjacent juxt a position and multiple-row interval.

A matrix display interface for a presentation system, adapted to selectively display presentation data on a display interface when the presentation system is operated on a computer, the units when clicked The matrix display interface for a presentation system according to claim 2, wherein the presentation data written into the data units is one selected from a group consisting of texts,.

The matrix display interface for a presentation system according to claim 2, wherein an arrangement mode of the data units is one selected from a group consisting of no-interval checkerboard arrangement, equal interval checkerboard arrangement, and unequal interval checkerboard arrangement. The matrix display interface for a presentation system according to claim 2, wherein the switching elements are selected from a group consisting of a main switching element, at least one column switching element, at least one row switching element, and at least one grid switching element. The matrix display interface for a presentation system according to claim 2, wherein the main switching element is used for switching to display or conceal all the data units.

The matrix display interface for a presentation system according to claim 2, wherein the column switching element is used for switching to display or conceal a whole column of the corresponding data units. The matrix display interface for a presentation system according to claim 2, wherein the row switching element is used for switching to display or conceal a whole row of the corresponding data units. The matrix display interface for a presentation system according to claim 2, wherein the grid switching element is used for switching to display or conceal the corresponding data units.

A matrix display interface for a presentation system, adapted to selectively display presentation data on a display interface when the presentation system is operated on a computer, the matrix display interface comprising: a plurality of data units, arranged on the display interface, and written with presentation data in advance; a plurality of mask units, covered on positions of the data units, for switching the display of the presentation data in the data units; a main switching element, having a first prompt text, for switching to display or conceal all the mask units when clicked; at least one column switching element, having a second prompt text, for switching to display or conceal a whole column of the corresponding mask units when clicked; at least one row switching element, having a third prompt text, for switching to display or conceal a whole row of the corresponding mask units when clicked; and at least one grid switching element, corresponding to the data units, for switching

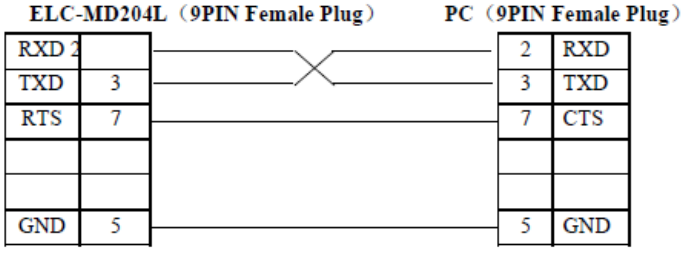
to display or conceal the corresponding mask units when clicked.

The matrix display interface for a presentation system according to claim 10, wherein the presentation data written into the data units is one selected from a group consisting of texts. The matrix display interface for a presentation system according to claim 10, wherein a display type of the main switching element is one selected from a group consisting of a transparent element with a ground color, an opaque element with a ground color, a transparent element without a ground color, and an opaque element without a ground color. The matrix display interface for a presentation system according to claim 10, wherein a display type of the column switching elements is one selected from a group consisting of a transparent element with a ground color, an opaque element with a ground color, a transparent element without a ground color, and an opaque element without a ground color. The matrix display interface for a presentation system according to claim 10, wherein a display type of the row switching elements is one selected from a group consisting of a transparent element with a ground color, an opaque element with a ground color, a transparent element without a ground color, and an opaque element without a ground color. The matrix display interface for a presentation system according to claim 10, wherein a display type of the grid switching elements is one selected from a group consisting of a transparent element with a ground color, an opaque element with a ground color, a transparent element without a ground color, and an opaque element without a ground color. The matrix display interface for a presentation system according to claim 10, further comprising a special effect switch having a fourth prompt text, for playing a display special effect of the data units. The matrix display interface for a presentation system according to claim 10, wherein the display special effect is to switch any one of a font type, a font size, a font color, a flicker frequency, and fade-out or fade-in of the presentation data written into the data units.

Pin Definition of the Serial Port of ELC-MD204L:

PIN	DEFINITION
1	TD+
2	RXD
3	TXD
4	NC
5	GND
6	TD-
7	RTS
8	RD-
9	RD+

MD2-PC Connection Guide:



RESULT- ELC matrix interface is completed.

EXPERIMENT NO. 8

Aim: Interfacing design for A/D & D/A converter

APPARATUS REQUIRED: 8051 Trainer kit, DAC interface board.

THEORY ;- DAC 0800 is an 8- bit DAC and the output voltage variation is between -5v and + 5v. The output voltage varies in steps of 10/256 =0.04 (appx).The digital data input and the corresponding output voltage are presented in the table below.

Input data in HEX	Output voltage
00	-5.00
01	-4.96
02	-4.92
..	..
7F	0.00
..	..
FD	4.92
FE	4.96
EF	5.00

Reffering to table 1 with 00H as input to DAC the analog output is -5 V Similarly, with FF H input , the output is +5V. outputting digital data 00 and FF at regular intervals, to DAC, result in different wave foms namely square ,triangular etc.

Square Wave Generation

```
                ORG             4100
                MOV             DPTR , PORT  ADDRESS  OF  DAC
START;          MOV             A,#00
                MOVX            @DPTR ,A
                LCALL           DELAY
                MOV             A,#FF
                MOVX            @DPTR ,A
                LCALL           DELAY
                LJUMP           START
DELAY:          MOV             R1,#05
LOOP:           MOV             R2 ,#FF
HERE:           DJNZ            R2, HERE
                DJNZ            R1,LOOP
                RET
                SJMP            START
```

Saw tooth Wave Generation

```
                ORG             4100
                MOV             DPTR, PORT ADDRESS  OF  DAC
                MOV             A,#00
LOOP:           MOVX            @DPTR , A
                INC             A
                SJMP            LOOP
```

Triangular Wave Generation

```
        ORG          4100
        MOV          DPTR ,PORT ADDRESS OF DAC
START:  MOV          A,#00
LOOP1:  MOVX         @DPTR ,A
        INC          A
        JNZ          LOOP1
        MOV          A,#FF
LOOP2:  MOVX         @DPTR ,A
        DEC          A
        JNZ          LOOP2
        LJMP         START
```

RESULT: Thus the square, triangular and saw tooth wave form were generated by interfacing DAC with 8051 timer kit.

EXPERIMENT NO. 9

Aim: Develop 8051 assembly language programmes on trainer kit for OR-ing of two 8-bit numbers

Apparatus required: 8051 Simulator / kit, PC, Power supply, connecting wires

Algorithm

1. Clear C reg. for carry
2. Get the data immediately.
3. ORing the two data
4. Store the result in memory pointed by DPTR

Program

```
        ORG 4100
        CLR C
        MOV A # data1
        ORL A # data2
        MOV DPTR # 4500
        MOV X @ DPTR A
HERE:   SJMP HERE
```

Result: Thus the program to perform ORing of two 8 bit numbers using 8051 instruction set was executed

EXPERIMENT NO. 10

Aim: Develop 8051 assembly language programmes on trainer kit for AND-ing of two 8-bit numbers

Apparatus required: 8051 Simulator / kit, PC, Power supply, connecting wires

Algorithm

1. Clear C reg. for carry
2. Get the data immediately.
3. ANDing the two data
4. Store the result in memory pointed by DPTR

Program

```
        ORG 4100
        CLR C
        MOV A # data1
        ANL A # data2
        MOV DPTR # 4500
        MOV X @ DPTR A
HERE:   SJMP HERE
```

Result: Thus the program to perform ANDing of two 8 bit numbers using 8051 instruction set was executed

EXPERIMENT NO. 11

Aim: Develop 8051 assembly language programmes on trainer kit for Inverse AND-ing of a 8-bit Numbers

Apparatus required: 8051 Simulator / kit, PC, Power supply, connecting wires

Algorithm

1. Clear C reg. for carry
2. Get the data immediately.
3. Inverse ANDing the two data
4. Store the result in memory pointed by DPTR

Program

```
        ORG 4100
        CLR C
        MOV A # data1
        ANL A # data2
        CPL A
        MOV DPTR # 4500
        MOV X @ DPTR A
HERE:   SJMP HERE
```

Result: Thus the program to perform Inverse-ANDing of two 8 bit numbers using 8051 instruction set was executed

EXPERIMENT NO. 12

Aim: Develop 8051 assembly language programmes on trainer kit for Inverse EX-ORing of 8-bit numbers

Apparatus required: 8051 Simulator / kit, PC, Power supply, connecting wires

Algorithm

1. Clear C reg. for carry
2. Get the data immediately.
3. Inverse EX-ORing the two data
4. Store the result in memory pointed by DPTR

Program

```
        ORG 4100
        CLR C
        MOV A # data1
        XRL A # data2
        CPL A
        MOV DPTR # 4500
        MOV X @ DPTR A
HERE:   SJMP HERE
```

Result: Thus the program to perform Inverse-EX-ORing of two 8 bit numbers using 8051 instruction set was executed

EXPERIMENT NO. 13

Aim: Develop 8051 assembly language programmes on trainer kit for finding 1's and 2's complements of 8-bit numbers

Apparatus required: 8051 Simulator / kit, PC, Power supply, connecting wires

Algorithm for 1"S complement

1. Clear C reg. for carry
2. Get the data immediately.
3. Complement the data of Accumulator.
4. Store the result in memory pointed by DPTR

Program for 1"S complement

```
        ORG 4100
        CLR C
        MOV A # data1
        CPL A
        MOV DPTR # 4500
        MOV X @ DPTR A
HERE:   SJMP HERE
```

Algorithm for 2"S complement

1. Clear C reg. for carry
2. Get the data immediately.
3. Complement the data of Accumulator.
4. Add 01H data into Accumulator.
5. Store the result in memory pointed by DPTR

Program for 2"S complement

```
        ORG 4100
        CLR C
        MOV A # data1
        CPL A
```

```
        ADD A # 01H
        MOV DPTR # 4500

        MOV X @ DPTR A
HERE:   SJMP HERE
```

Result: Thus the program to perform 1's & 2's complement of 8 bit numbers using 8051 instruction set was executed