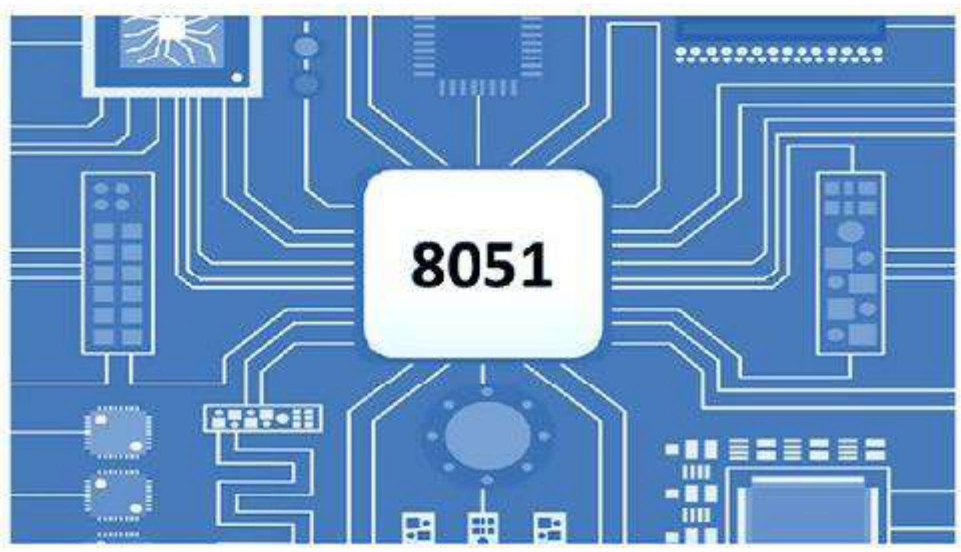
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

SEMESTER -VI

EC334 MICROCONTROLLER LAB



VISION

To nurture the talents of electronics and communication engineers, making them highly competent for growth of the society.

MISSION

- To deliver excellence in teaching - learning process.
- Promote safe, orderly, caring and supportive environment to learners.
- Development of skilled engineers to perform innovative Research for betterment of the society.
- To encourage industry - institute interaction, career advancement, innovation and entrepreneurship development.

PROGRAM EDUCATIONAL OUTCOME (PEO)

PEO1: To acquire a strong foundation in mathematics and scientific fundamentals, to develop an ability to analyze various functional elements of different disciplines of electronics and communication engineering.

PEO2: Develop technical competence to move in pace with rapid changes in technology.

PEO3: Equip learners to strengthen knowledge and soft skills for carrier advancement.

PEO4: Adhere to ethics to contribute for betterment of the society.

PROGRAM SPECIFIC OUTCOMES (PSO)

PSO1. To understand principles and applications of various electronic components/devices and circuits.

PSO2. Enable learners to solve complex problems using modern hardware and software tools.

COURSE OUTCOME

C334.1	Students will be able to understand fundamental programming concepts of microcontrollers.
C334.2	Students will be able to have an in-depth knowledge on interfacing the external devices to the controllers.
C334.3	Students will be able to design a microcontroller based system with the help of the interfacing devices.
C334.4	Students will be able to have an in-depth knowledge of applying the concepts on real- time applications.

CO-PO MAPPING

PO / **CO**	**PO 1**	**PO 2**	**PO 3**	**PO 4**	**PO 5**	**PO 6**	**PO 7**	**PO 8**	**PO 9**	**PO1 0**	**PO1 1**	**PO1 2**	**PSO 1**	**PSO 2**
C334.1	**3**	**-**	**1**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**
C334.2	**3**	**-**	**1**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**2**	**-**
C334.3	**3**	**-**	**3**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**2**	**-**
C334.4	**3**	**-**	**2**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**2**	**-**
C334	**3**	**-**	**1.7**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**1.5**	**-**

SYLLABUS

COURSE CODE	COURSE NAME	L-T-P-C	YEAR OF INTRODUCTION
EC334	**Microcontroller Lab**	**0-0-3-1**	**2016**

Prerequisite: EC305 Microprocessors & Microcontrollers

Course objectives:

1. To understand Assembly Language/embedded C programming of Microcontroller.
2. To interface simple peripheral devices to a Microcontroller.
3. To equip student groups to design and implement simple embedded systems.

List of Experiments:

PART –A(At least 6 experiments are mandatory)

Assembly Language Programming experiments using 8051 Trainer kit.

1. Data transfer/exchange between specified memory locations.
2. Largest/smallest from a series.
3. Sorting (Ascending/Descending) of data.
4. Addition / subtraction / multiplication / division of 8/16 bit data.
5. Sum of a series of 8 bit data.
6. Multiplication by shift and add method.
7. Square / cube / square root of 8 bit data.
8. Matrix addition.
9. LCM and HCF of two 8 bit numbers.
10. Code conversion – Hex to Decimal/ASCII to Decimal and vice versa.

PART –B (At least 4 experiments are mandatory)

Interfacing experiments using 8051 Trainer kit and interfacing modules.

1. Time delay generation and relay interface.
2. Display (LED/Seven segments/LCD) and keyboard interface.
3. ADC interface.
4. DAC interface with wave form generation.
5. Stepper motor and DC motor interface.
6. Realization of Boolean expression through port.
7. Elevator interfacing.

PART –C (At least 2 experiments are mandatory)

Programming / interfacing experiments with IDE for 8051/PIC/MSP/Arduino/Raspberry Pi based interfacing boards/sensor modules (Direct downloading of the pre-written ALP/'C'/Python programs can be used).

1. Relay control
2. Distance measurement.
3. Temperature measurement / Digital Thermometer
4. Txr-Rxr interface.
5. Alphanumeric LCD display interface.
6. Simple project work including multiple interfaces.

Expected outcome:

The students will be able to:

1. Program Micro controllers.
2. Interface various peripheral devices to Micro controller.
3. Function effectively as an individual and in a team to accomplish the given task.

LIST OF EXPERIMENTS

EXPT. NO	NAME OF THE EXPERIMENT
1	DATA TRANSFER
2	LARGEST FROM A SERIES
3	8 BIT ADDITION, SUBTRACTION, MULTICATION AND DIVISION
4	16 BIT ADDITION
5	SQUARE OF A NUMBER
6	SQUARE ROOT OF A NUMBER
7	SUM OF N NUMBER
8	SORTING OF N NUMBERS
9	FACTORIAL OF A NUMBER
10	MATRIX ADDITION
11	FIBONACCI SERIES
12	16 BIT ADDITION/SUBTRACTION
13	8 BIT MULTILICATION /DIVISION
14	INTERFACING LED WITH 8051
15	INTERFACING OF SEVEN SEGMENT LED WITH 8051
16	INTERFACING OF STEPPER MOTOR WITH 8051
17	SQUARE WAVE GENERATION
18	INTERFACING OF LCD WITH 8051
19	INTERFACING OF EM RELAY WITH 8051

INDUX

INTRODUCTION TO 8051 MICROCONTROLLER

8051 microcontroller is designed by Intel in 1981. It is an 8-bit microcontroller. It is built with 40 pins DIP (dual inline package), 4kb of ROM storage and 128 bytes of RAM storage, 2 16-bit timers. It consists of are four parallel 8-bit ports, which are programmable as well as addressable as per the requirement. An on-chip crystal oscillator is integrated in the microcontroller having crystal frequency of 12 MHz.

The 8051 memory is organized in a Harvard Architecture. Both the code memory space and data memory space begin at location 0x00 for internal or external memory which is different from the Princeton Architecture where code and data share same memory space. The advantage of the Harvard Architecture is not only doubling the memory capacity of the microcontroller with same number of address lines but also increases the reliability of the microcontroller, since there are no instructions to write to the code memory which is read only.

1. Features of 8051 Microcontroller

An 8051 microcontroller comes bundled with the following features.

- 64K bytes on-chip program memory (ROM)
- 128 bytes on-chip data memory (RAM)
- Four register banks
- 128 user defined software flags
- 8-bit bidirectional data bus
- 16-bit unidirectional address bus
- 32 general purpose registers each of 8-bit
- 16 bit Timers (usually 2, but may have more or less)
- Three internal and two external Interrupts
- Four 8-bit ports,(short model have two 8-bit ports)
- 16-bit program counter and data pointer

2. Architecture Of 8051 Microcontroller

Figure 1.4 shows the architecture block diagram of 8051.

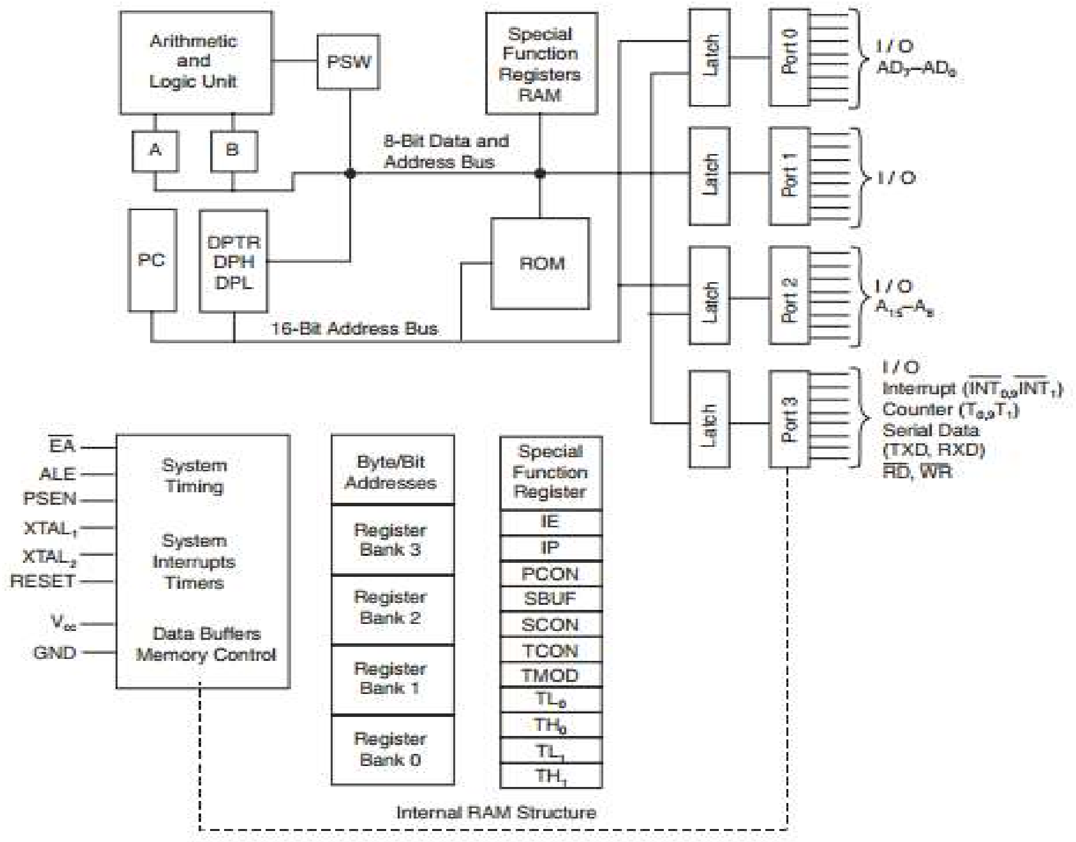


Fig. 1.4 Architectural block diagram of microcontroller 8051

2.1 8051 Memory Organization

8051 microcontroller has an internal program of 4K size and if needed an external memory can be added (by interfacing) of size 60K maximum. So in total 64K size memory is available for 8051 micro controller. By default, the External Access (EA) pin should be connected Vcc so that instructions are fetched from internal memory initially. When the limit of internal memory (4K) is crossed, control will automatically move to external memory to fetch remaining instructions. If the programmer wants to fetch instruction from external memory only (bypassing the internal memory), then he must connect External Access (EA) pin to ground (GND).

2.2 Timers and Counters

Timer means which can give the delay of particular time between some events. For example on or off the lights after every 2 sec. This delay can be provided through some assembly program but in microcontroller two hardware pins are available for delay generation. These hardware pins can be also used for counting some external events. How much times a number is repeated in the given table is calculated by the counter.

In MC8051, two timer pins are available T0 and T1, by these timers we can give the delay of particular time if we use these in timer mode. We can count external pulses at these pins if we use these pins in counter mode. 16 bits timers are available. Means we can generate delay between 0000H to FFFFH. Two special function registers are available. If we want to load T0 with 16 bit data then we can load separate lower 8 bit in TL0 and higher 8 bit in TH0. In the same way for T1. TMOD, TCON registers are used for controlling timer operation.

2.3 Serial Port

There are two pins available for serial communication TXD and RXD. Normally TXD is used for transmitting serial data which is in SBUF register, RXD is used for receiving the serial data. SCON register is used for controlling the operation.

2.4 Input Output Ports

There are four input output ports available P0, P1, P2, P3. Each port is 8 bit wide and has special function register P0, P1, P2, P3 which are bit addressable means each bit can be set or reset by the Bit instructions (SETB for high, CLR for low) independently. The data at any port which is transmitting or receiving is in these registers. The port 0 can perform dual works. It is also used as Lower order address bus (A0 to A7) multiplexed with 8 bit data bus P0.0 to P0.7 is AD0 to AD7 respectively the address bus and data bus is demultiplex by the ALE signal and latch which is further discussed in details. Port 2 can be used as I/O port as well as higher order address bus A8 to A15. Port 3 also have dual functions it can be worked as I/O as well as each pin of P3 has specific function. P3.0 – RXD – {Serial I / P for Asynchronous communication Serial O / P for synchronous communication. P3.1 – TXD – Serial data transmit. P3.2 – INT0 – External Interrupt 0. P3.3 – INT1 – External Interrupt 1. P3.4 – T0 – Clock input for counter 0. P3.5 – T1 – Clock input for counter 1. P3.6 – WR – Signal for writing to external memory. P3.7 – RD – Signal for reading from external memory. When external memory is interfaced with 8051 then P0 and P2 can't be worked as I/O port they works as address bus and data bus, otherwise they can be accessed as I/O ports.

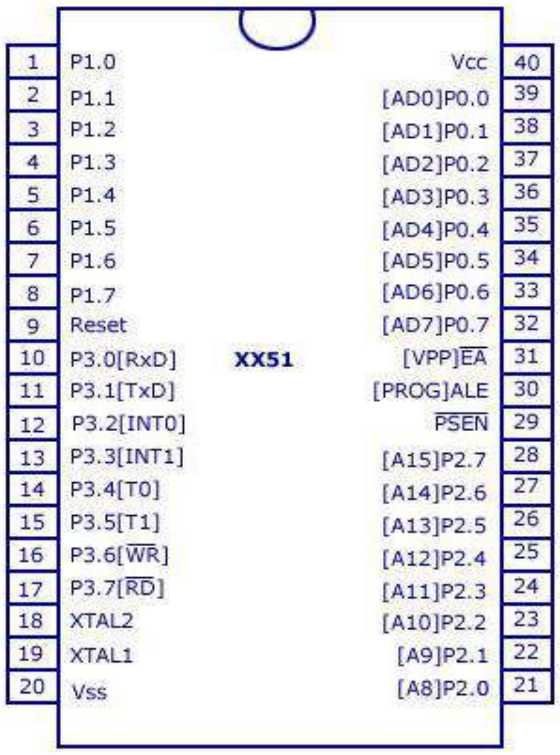
2.5 Oscillator

It is used for providing the clock to MC8051 which decides the speed or baud rate of MC. We use crystal which frequency vary from 4MHz to 30 MHz, normally we use 11.0592 MHz frequency.

2.6 Interrupts

Interrupts are defined as requests because they can be refused (masked) if they are not used, that is when an interrupt is acknowledged. A special set of events or routines are followed to handle the interrupts. These special routines are known as interrupt handler or interrupt service routines (ISR). These are located at a special location in memory. • INT0 and INT1 are the pins for external interrupts.

3. Pin Diagram Of 8051 microcontroller



3.1 Pin Description

The EA' (External Access) pin is used to control the internal or external memory access. The signal 0 is for external memory access and signal 1 for internal memory access. The PSEN' (Program Store Enable) is for reading external code memory when it is low (0) and EA is also 0. The ALE (Address Latch Enable) activates the port 0 joined with port 2 to provide 16 bit external address bus to access the external memory. The ALE multiplexes the P0: 1 for latching address on P0 as A0-A7 in the 16 bit address buss, 0 for latching P0 as data I/O. P0.x is named ADx because P0 is multiplexed for Address bus and Data bus at different clock time. WR' provides the signal to write external data memory RD' provides the signal to read external data and code memory.

- PORT P1 (Pins 1 to 8): The port P1 is a port dedicated for general I/O purpose. The other ports P0, P2 and P3 have dual roles in addition to their basic I/O function.

• PORT P0 (pins 32 to 39): When the external memory access is required then Port P0 is multiplexed for address bus and data bus that can be used to access external memory in conjunction with port P2. P0 acts as A0-A7 in address bus and D0-D7 for port data. It can be used for general purpose I/O if no external memory presents.

• PORT P2 (pins 21 to 28): Similar to P0, the port P2 can also play a role (A8-A15) in the address bus in conjunction with PORT P0 to access external memory.

- PORT P3 (Pins 10 to 17): In addition to acting as a normal I/O port,
  - P3.0 can be used for serial receive input pin(RXD) • P3.1 can be used for serial transmit output pin(TXD) in a serial port,
  - P3.2 and P3.3 can be used as external interrupt pins(INT0’ and INT1’),
  - P3.4 and P3.5 are used for external counter input pins(T0 and T1),
  - P3.6 and P3.7 can be used as external data memory write and read control signal pins(WR’ and RD’)read and write pins for memory access.

Experiment No:01

DATA TRANSFER

Date:

AIM: Write an assembly language program to transfer data between specified memory location.

PROGRAM:

```
ORG 0000H

MOV R0,#40H

MOV R1,#50H

MOV R7,#0AH

BACK:MOV A,@R0

MOV @R1,A

INC R0

INC R1

DJNZ R7,BACK

END
```

OUTPUT:

	LOCATION	**DATA**
INPUT	**40H**	**01**
	41H	**02**
	42H	**03**
	43H	**04**
	44H	**05**
	45H	**06**
	46H	**07**
	47H	**08**
	48H	**09**
	49H	**10**
OUTPUT	**50H**	**01**
	51H	**02**
	52H	**03**
	53H	**04**
	54H	**05**
	55H	**06**
	56H	**07**
	57H	**08**
	58H	**09**
	59H	**10**

Experiment No:02 **LARGEST FROM A SERIES**

Date:

AIM: Write an assembly language program to find the largest number from a series

PROGRAM:

```
ORG 0000H
MOV A,50H
MOV R2,A
DEC R2
L3:
        MOV A,R2
        MOV R3,A
        MOV R0,#51H
L2:
        MOV A,@R0
        MOV R4,A
        INC R0
        MOV A,@R0
        MOV R5,A
        SUBB A,R4
        JNC L1
        MOV A,R4
        MOV @R0,A
        DEC R0
        MOV A,R5
        MOV @R0,A
        INC R0
L1:
        DJNZ R3,L2
        DJNZ R2,L3
        MOV A,#50H
        ADD A,50H
        MOV 80H,A
END
```

OUTPUT:

	LOCATION	**DATA**
INPUT	**50H**	**05**
	51H	**03**
	52H	**05**
	53H	**02**
	54H	**01**
	55H	**04**
OUTPUT	**50H**	**05**
	51H	**01**
	52H	**02**
	53H	**03**
	54H	**04**
	55H	**05**
	80H	**05**

Experiment No:03 **8 BIT ADDITION, SUBTRACTION, MULTICATION AND DIVISION**

Date:

AIM: Write an assembly language program to find addition, subtraction, multiplication and division

ADDITION

PROGRAM:

```
ORG 000H

MOV A,40H

MOV B,41H

ADD A,B

MOV 42H,A

END
```

OUTPUT:

	LOCATION	DATA
INPUT	**40H** **41H**	**02** **03**
OUTPUT	**42H**	**05**

SUBTRACTION

PROGRAM:

```
ORG 000H

MOV A,40H

MOV B,41H

SUBB A,B

MOV 42H,A

END
```

OUTPUT:

	LOCATION	DATA
INPUT	**40H** **41H**	**05** **03**
OUTPUT	**42H**	**02**

MULTIPLICATION

PROGRAM:

```
ORG 0000H

MOV A,40H

MOV B,41H

MUL AB

MOV 42H,A

MOV A,B

MOV 43H,A

END
```

OUTPUT:

	LOCATION	DATA
INPUT	**40H** **41H**	**02** **02**
OUTPUT	**42H** **43H**	**04** **00**

DIVISION

PROGRAM:

```
ORG 000H

MOV A,40H

MOV B,41H

DIV AB

MOV 42H,A
MOV A,B

MOV 43H,A

END
```

OUTPUT:

	LOCATION	**DATA**
INPUT	**40H** **41H**	**02** **02**
OUTPUT	**42H** **43H**	**01** **00**

Experiment No:4

16 BIT ADDITION

Date:

AIM: Write an assembly language program to find 16 bit addition from internal and external memory

PROGRAM:

16 BIT ADDITION FROM EXTERNAL MEMORY

PROGRAM

```
ORG 0000H
MOV DPTR,#4300H
MOVX A,@DPTR
MOV R0,A
INC DPTR
MOVX A,@DPTR
MOV R1,A
INC DPTR
MOVX A,@DPTR
MOV R2,A
INC DPTR
MOVX A,@DPTR
MOV R3,A
MOV A,R0
ADDC A,R2
MOV DPTR,#4500H
MOVX @DPTR,A
MOV A,R1
ADDC A,R3
```

```
MOV DPTR,#4501H
MOVX @DPTR,A
END
```

OUTPUT:

	LOCATION	**DATA**
INPUT	**4300H** **4301H** **4302H** **4303H**	**05** **10** **15** **20**
OUTPUT	**4500H** **4501H**	**1A** **30**

16 BIT ADDITION FROM INTERNAL MEMORY

PROGRAM

```
ORG 0000H
MOV A,60H
MOV R0,61H
MOV R1,62H
MOV R2,63H
ADDC A,R1
MOV 70H,A
CLR A
MOV A,R0
ADDC A,R2
MOV 71H,A
END
```

OUTPUT:

	LOCATION	**DATA**
INPUT	**60H** **61H** **62H** **63H**	**01** **01** **10** **10**
OUTPUT	**70H** **71H**	**11** **11**

Experiment No:5

SQUARE OF A NUMBER

Date:

AIM: Write an assembly language program to find the square of a number.

PROGRAM:

```
ORG 0000H
MOV A,60H
MOV B,A
MUL AB
MOV 70H,A
MOV 71H,B
END
```

OUTPUT:

	LOCATION	**DATA**
INPUT	**60H**	**03**
OUTPUT	**70H** **71H**	**09** **00**

Experiment No:6

Date:

SQUARE ROOT OF A NUMBER

AIM: Write an assembly language program to find the square root of a number.

PROGRAM:

```
ORG 0000H
MOV A,60H
MOV R1,#00H
MOV R0,#01H
L1:
        SUBB A,R0
        JC L2
        INC R0
        INC R0
        INC R1
        JNZ L1
        MOV A,R1
        SJMP L3
L2:
        MOV A,#0FFH
L3:
        MOV 70H,A
END
```

OUTPUT

	LOCATION	DATA
INPUT	**60H**	**09**
OUTPUT	**70H**	**03**

Experiment No:7 **SUM OF N NUMBER**

Date:

AIM: Write an assembly language program to find the sum of n numbers

PROGRAM:

```
ORG 0000H
MOV A,60H
MOV R1,#00H
MOV R0,#01H
L1:
        SUBB A,R0
        JC L2
        INC R0
        INC R0
        INC R1
        JNZ L1
        MOV A,R1
        SJMP L3
L2:
        MOV A,#0FFH
L3:
        MOV 70H,A
END
```

OUTPUT:

	LOCATION	**DATA**
INPUT	**40H**	**06**
	60H	**01**
	61H	**02**
	62H	**03**
	63H	**04**
	64H	**05**
	65H	**06**
OUTPUT	**80H**	**15**
	81H	**00**

Experiment No:8 **SORTING OF N NUMBERS**

Date:

AIM: Write an assembly language program to sort n numbers.

PROGRAM:

```
ORG 0000H

MOV A,50H

MOV R2,A

DEC R2

L3:

        MOV A,R2

        MOV R3,A

        MOV R0,#51H

L2:

        MOV A,@R0

        MOV R4,A

        INC R0

        MOV A,@R0

        MOV R5,A

        SUBB A,R4

        JNC L1

        MOV A,R4

        MOV @R0,A

        DEC R0

        MOV A,R5
        MOV @R0,A
        INC R0
L1:
        DJNZ R3,L2
        DJNZ R2,L3
END
```

OUTPUT:

	LOCATION	**DATA**
INPUT	**50H** **51H** **52H** **53H** **54H** **55H**	**05** **03** **02** **05** **01** **04**
OUTPUT	**50H** **51H** **52H** **53H** **54H** **55H**	**05** **01** **02** **03** **04** **05**

Experiment No:9

FACTORIAL OF A NUMBER

Date:

AIM: Write an assembly language program to find the factorial of a number using 8051.

PROGRAM:

```
ORG 0000H
MOV DPTR,#4200H
MOV A,@DPTR
MOV R1,A

LOOP:
    DEC R1
    MOV B,R1
    MUL AB
    CJNE R1,#01,LOOP
    MOV DPTR,#4201H
    MOVX @DPTR,A
END
```

Experiment No:10 **MATRIX ADDITION**

Date:

AIM: Write an assembly language program to find the sum of two matices using 8051.

PROGRAM:

```
ORG 0000H
MOV DPTR,#4200H
MOV A ,@DPTR
MOV B,A
INC DPTR
MOVX A,@DPTR
MUL AB
MOV R0,A
MOV DPL,#00H
NEXT:
        MOV DPH,#43H
        MOVX A,@DPTR
        MOV R1,A
        MOV DPH,#44H
        MOVX A,@DPTR
        ADD A,R1
        MOV DPH,#45H
        MOVX @DPTR,A
        INC DPTR
        DJNZ R1,NEXT
HERE:   SJMP HERE
END
```

Experiment No:11 **FIBONACCI SERIES**

Date:

AIM: Write an assembly language program to find the Fibonacci series upto n numbers using 8051.

PROGRAM:

```
ORG 0000H
MOV DPTR,#0000H
MOVX A,@DPTR
MOV R0,A
MOV A,#00H
INC DPTR
MOVX @DPTR,A
MOV A,#01H
MOV B,#00H
LOOP:
        INC DPTR
        MOV R1,A
        MOVX @DPTR,A
        ADD A,B
        MOV B,R1
        DEC R0
        CJNE R0,#01H,LOOP
END
```

Experiment No:12 **<u>16 BIT ADDITION/SUBTRACTION</u>**

Date:

AIM: Write an assembly language program to find the sum and difference of two 16 bit numbers using 8051.

PROGRAM:

```
ORG 0000H
MOV DPTR,#4200H
MOVX A,@DPTR
MOV B,A
INC DPTR
MOVX A,@DPTR
ADD A,B
INC DPTR
MOVX @DPTR,A
MOV DPTR,#4200H
MOVX A,@DPTR
MOV B,A
INC DPTR
MOVX A,@DPTR
SUBB A,B
MOV DPTR,#4203H
MOVX @DPTR,A
END
```

Experiment No:13

8 BIT MULTILICATION /DIVISION

Date:

AIM: Write an assembly language program to find the product and division of two 8 bit numbers using 8051.

PROGRAM:

```
ORG 0000H
MOV DPTR,#4200H
MOVX A,@DPTR
MOV B,A
INC DPTR
MOVX A,@DPTR
MUL AB
INC DPTR
MOVX @DPTR,A
MOV DPTR,#4200H
MOVX A,@DPTR
MOV B,A
INC DPTR
MOVX A,@DPTR
DIV AB
MOV DPTR,#4203H
MOVX @DPTR,A
END
```

Experiment No:14 **INTERFACING LED WITH 8051**

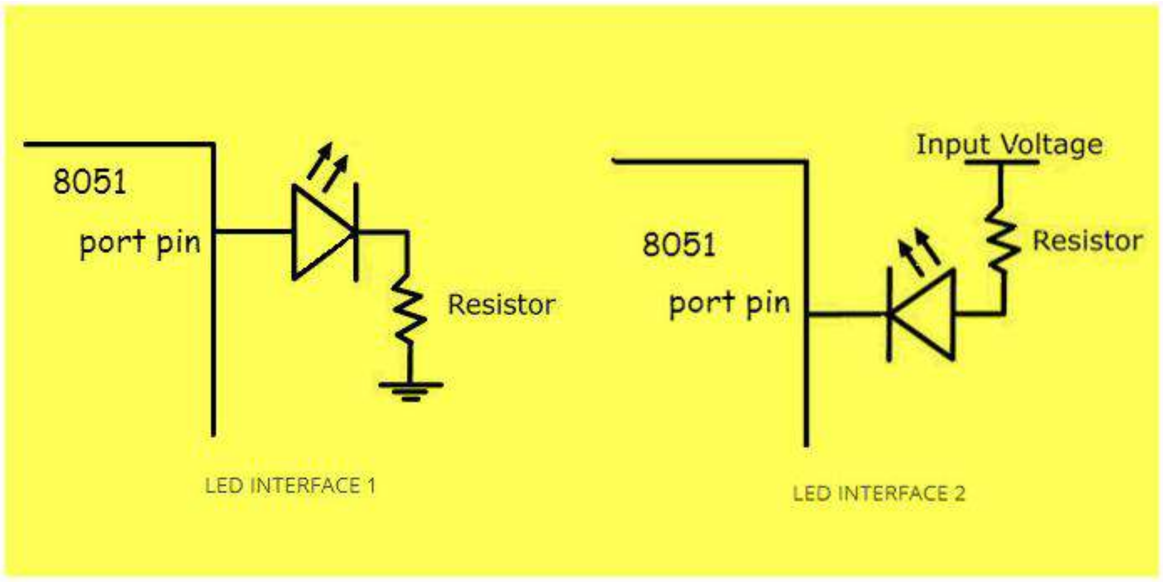
Date:

AIM: Write an assembly language program to for toggling the LED connected to one of the port pins of 8051.

THEORY:

Interfacing comprises of hardware (Interface device) and Software (source code to communicate, also called as the Driver). Simply, to use an LED as the output device, LED should be connected to Microcontroller port and the MC has to be programmed inside make LED ON or OFF or blink or dim. This program is called as the driver/firmware. The driver software can be developed using any programming language like Assembly, C etc.

There are two ways which we can interface LED to the Microcontroller 8051. But the connections and programming techniques will be different. This article provides the information on LED interfacing with 8051 and LED blinking code for AT89C52/ AT89C51 Microcontroller.



Interfacing LED to 8051 Methods

Observe carefully the interface LED 2 is in forward biased because the input voltage of 5v connected to the positive terminal of the LED, So here the Microcontroller pin should be at LOW level and vice versa with the interface 1 connections.

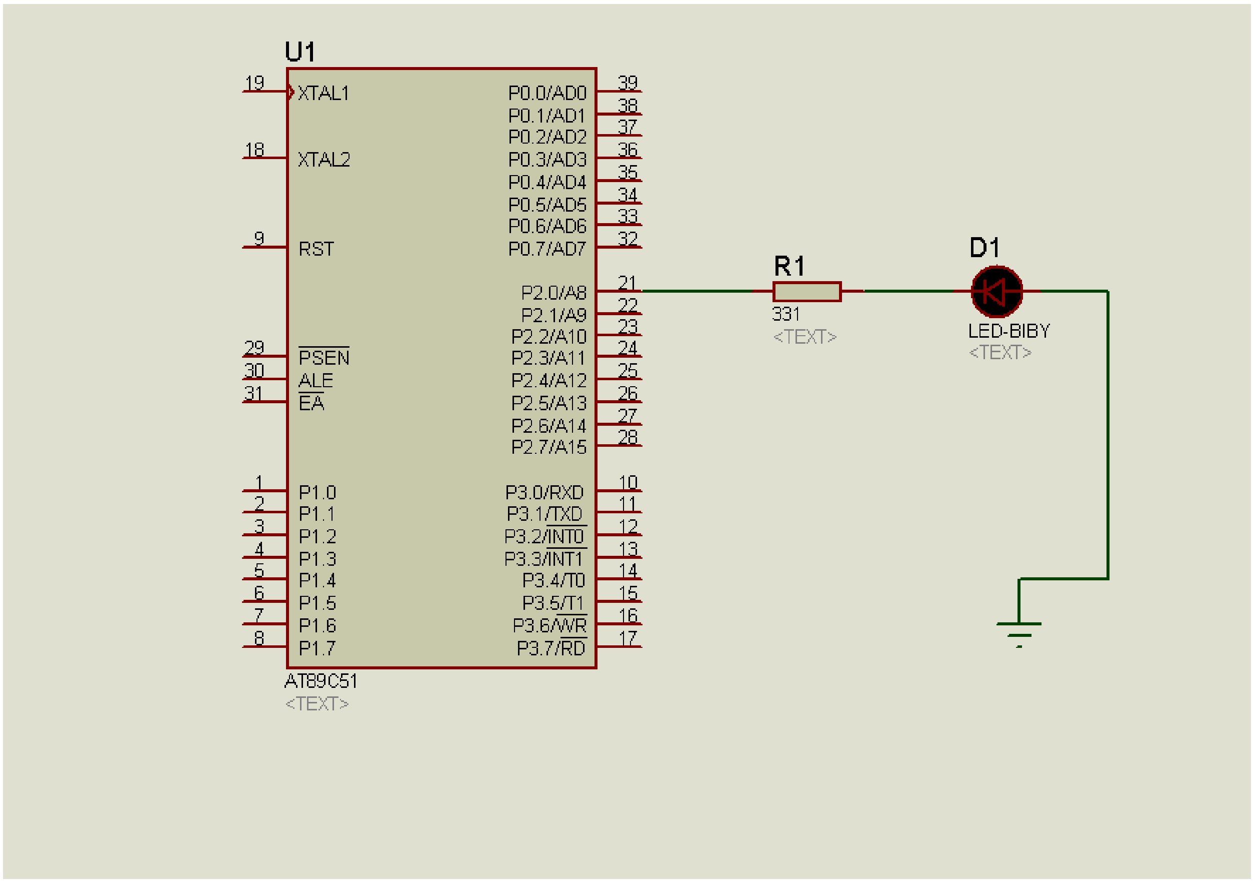
The resistor is important in LED interfacing to limit the flowing current and avoid damaging the LED and/or MCU.

- Interface 1 will glow LED, only if the PIN value of the MC is HIGH as current flows towards the ground.
- Interface 2 will glow LED, only if the PIN value of the MC is LOW as current flows towards PIN due to its lower potential.

PROGRAM:

```
ORG 0000H
LOOP: SETB P1.0
ACALL DELAY
CLR P1.0
ACALL DELAY
SJMP LOOP
DELAY: MOV R0,#100
AGAIN: MOV R1,#200
BACK:  DJNZ R1,BACK
DJNZ R0,AGAIN
RET
END
```

SCHEMATIC DIAGRAM:



INFERENCE:

Experiment No:15 **INTERFACING OF 7 SEGMENT DISPLAY WITH 8051**

Date:

AIM: Write an assembly language program for displaying the decimal numbers in 7 Segment display.

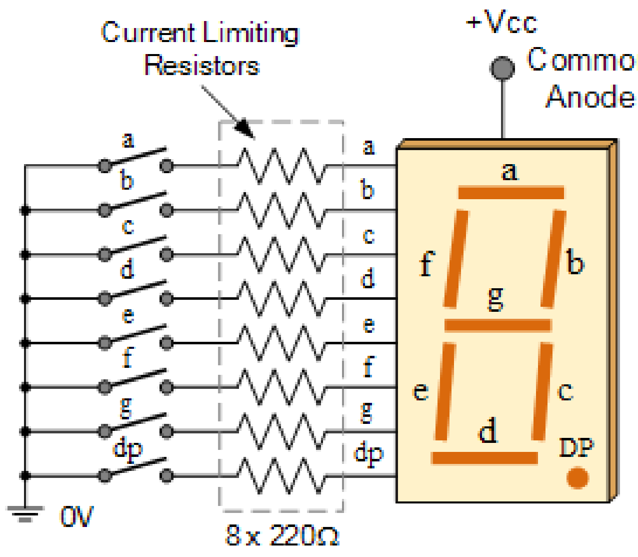
THEORY:

The *7-segment display*, also written as "seven segment display", consists of seven LEDs (hence its name) arranged in a rectangular fashion as shown. Each of the seven LEDs is called a segment because when illuminated the segment forms part of a numerical digit (both Decimal and Hex) to be displayed. An additional 8th LED is sometimes used within the same package thus allowing the indication of a decimal point, (DP) when two or more 7-segment displays are connected together to display numbers greater than ten.

Each one of the seven LEDs in the display is given a positional segment with one of its connection pins being brought straight out of the rectangular plastic package. These individually LED pins are labelled from a through to g representing each individual LED. The other LED pins are connected together and wired to form a common pin.So by forward biasing the appropriate pins of the LED segments in a particular order, some segments will be light and others will be dark allowing the desired character pattern of the number to be generated on the display. This then allows us to display each of the ten decimal digits 0 through to 9 on the same 7-segment display.

The displays common pin is generally used to identify which type of 7-segment display it is. As each LED has two connecting pins, one called the "Anode" and the other called the "Cathode", there are therefore two types of LED 7-segment display called: **Common Cathode** (CC) and **Common Anode** (CA).

The difference between the two displays, as their name suggests, is that the common cathode has all the cathodes of the 7-segments connected directly together and the common anode has all the anodes of the 7-segments connected together and is illuminated as follows.The Common Cathode (CC) – In the common cathode display, all the cathode connections of the LED segments are joined together to logic "0" or ground. The individual segments are illuminated by application of a “HIGH”, or logic “1” signal via a current limiting resistor to forward bias the individual Anode terminals (a-g).



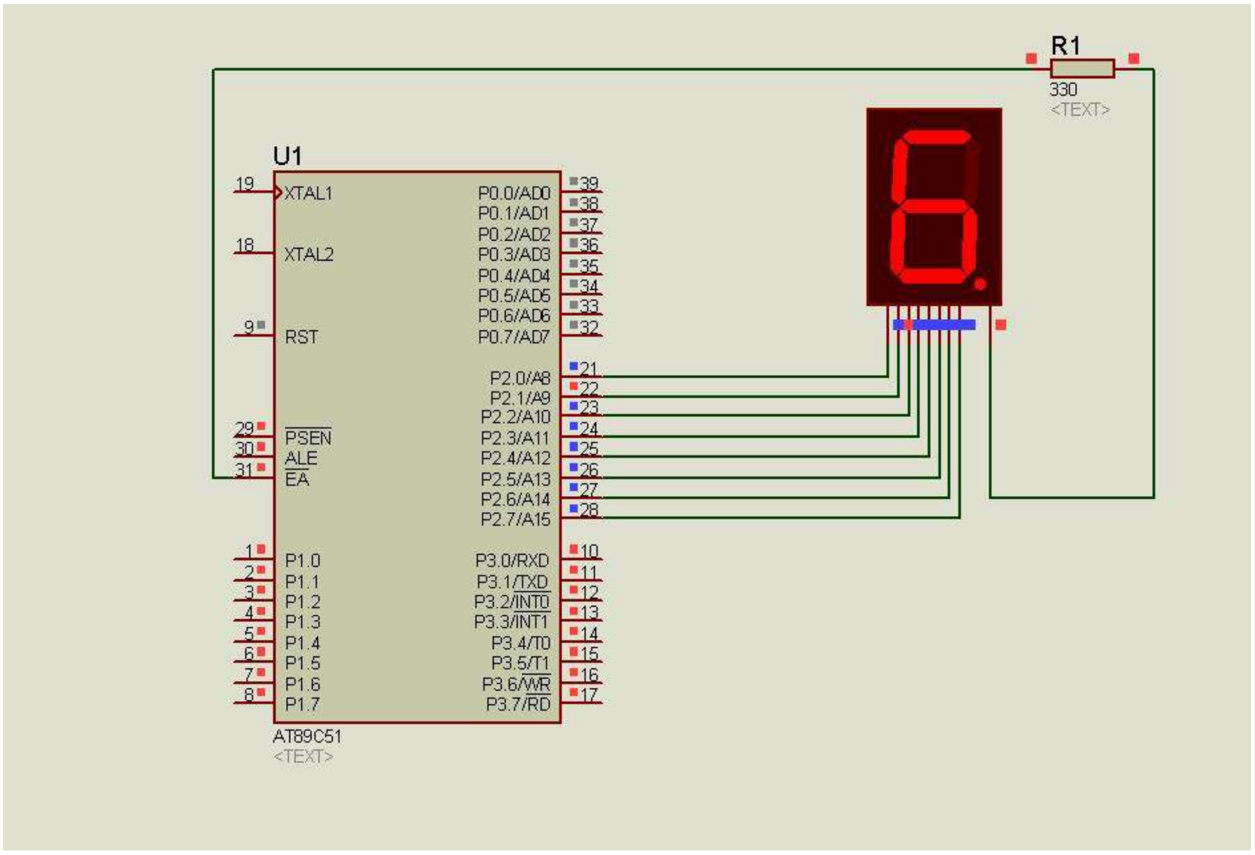
PROGRAM:

```
ORG 0000H
LOOP:MOV DPTR,#0100H
        MOV R1,#0AH
BACK:CLR A
        MOVC A,@A+DPTR
        MOV P2,A
        ACALL DELAY
        INC DPTR
        DJNZ R1,BACK
        SJMP LOOP

       DELAY:MOV R5,#05
       BACK2: MOV R3,#255
```

```
        BACK1:  MOV R4,#255
        AGAIN: DJNZ R4,AGAIN
               DJNZ R3,BACK1
               DJNZ R5,BACK2
               RET
ORG 0100H
DB 40H,79H,24H,30H,19H,12H,02H,58H,00H,10H
END
```

SCHEMATIC DIAGRAM:



INFERENCE:

Experiment No:16 **INTERFACING OF STEPPER MOTOR WITH 8051**

Date:

AIM: Write an assembly language program for interfacing stepper motor with 8051.

THEORY:

A **Stepper Motor** or a **step motor** is a brushless, synchronous motor which divides a full rotation into a number of steps. Unlike a brushless DC motor which rotates continuously when a fixed DC voltage is applied to it, a step motor rotates in discrete step angles. The **Stepper Motors** therefore are manufactured with steps per revolution of 12, 24, 72, 144, 180, and 200, resulting in stepping angles of 30, 15, 5, 2.5, 2, and 1.8 degrees per step. The stepper motor can be controlled with or without feedback. Stepper motors work on the principle of electromagnetism. There is a soft iron or magnetic rotor shaft surrounded by the electromagnetic stators. The rotor and stator have poles which may be teethed or not depending upon the type of stepper. When the stators are energized the rotor moves to align itself along with the stator (in case of a permanent magnet type stepper) or moves to have a minimum gap with the stator (in case of a variable reluctance stepper). This way the stators are energized in a sequence to rotate the stepper motor..

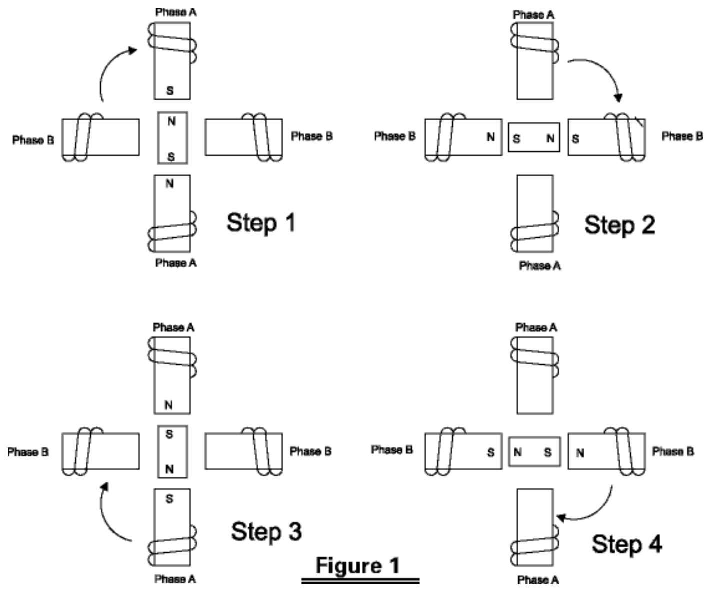
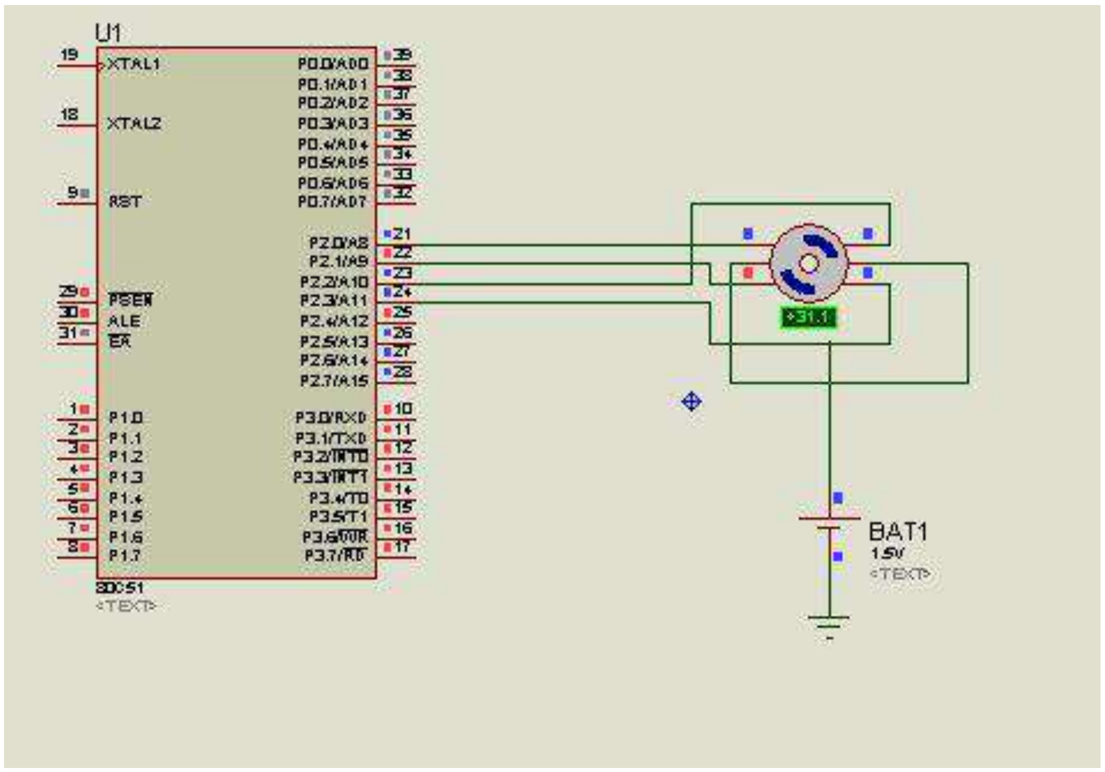


Figure 1

PROGRAM:

```
ORG 0000H
MOV A,#66H
LOOP:MOV P2,A
ACALL DELAY
RR A
SJMP LOOP
DELAY:MOV R5,#0AH
AGAIN:MOV R3,#0FFH
BACK:DJNZ R3,BACK
DJNZ R5,AGAIN
RET
END
```

SCHEMATIC DIAGRAM:



INFERENCE:

Experiment No:17 **SQUARE WAVE GENERATION**

Date:

AIM: Write an assembly language program to generate a square wave using 8051.

THEORY:

A timer can be generalized as a multi-bit counter which increments/decrements itself on receiving a clock signal and produces an interrupt signal up on roll over. When the counter is running on the processor's clock , it is called a "Timer", which counts a predefined number of processor clock pulses and generates a programmable delay. When the counter is running on an external clock source (may be a periodic or aperiodic external signal) it is called a "Counter" itself and it can be used for counting external events.

In 8051, the oscillator output is divided by 12 using a divide by 12 network and then fed to the Timer as the clock signal. That means for an 8051 running at 12MHz, the timer clock input will be 1MHz. That means the the timer advances once in every 1uS and the maximum time delay possible using a single 8051 timer is (2^16) x (1µS) = 65536µS. Delays longer than this can be implemented by writing up a basic delay program using timer and then looping it for a required number of time. We will see all these in detail in next sections of this article.
Designing a delay program using 8051 timers.

While designing delay programs in 8051, calculating the initial value that has to be loaded in to TH and TL registers forms a very important thing. Let us see how it is done.

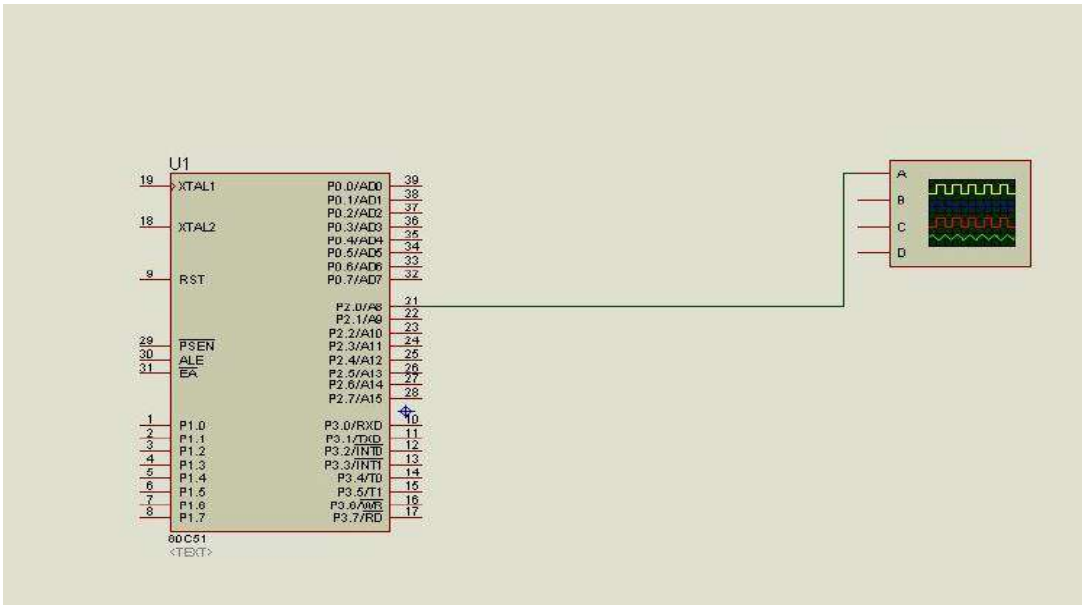
- Assume the processor is clocked by a 12MHz crystal.
- That means, the timer clock input will be 12MHz/12 = 1MHz
- That means, the time taken for the timer to make one increment = 1/1MHz = 1uS
- For a time delay of "X" uS the timer has to make "X" increments.
- 2^16 = 65536 is the maximim number of counts possible for a 16 bit timer.
- Let TH be the value value that has to be loaded to TH registed and TL be the value that has to be loaded to TL register.

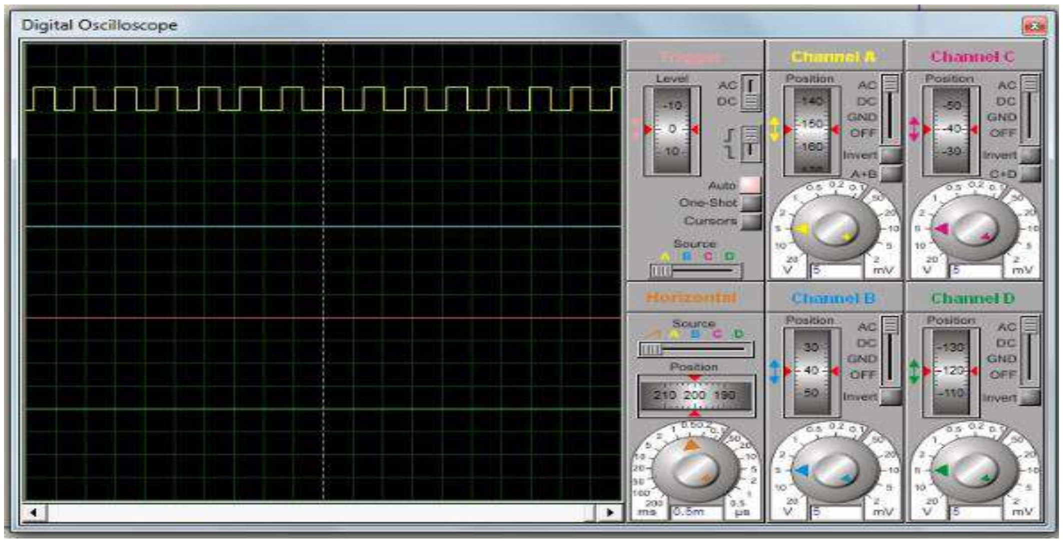
- Then, THTL = Hexadecimal equivalent of (65536-X) where (65536-X) is considered in decimal.

PROGRAM:

```
ORG 0000H
MOV TMOD,#20H
MOV TH1,#1AH
MAIN:CPL P1.3
ACALL DELAY
SJMP MAIN
DELAY:SETB TR1
AGAIN:JNB TF1,AGAIN
CLR TR1
CLR TF1
RET
END
```

SCHEMATIC DIAGRAM:




Digital Oscilloscope
Trigger
Channel A
Channel C
Horizontal
Channel B
Channel D
Auto
One-Shot
Cursors
Source
Position

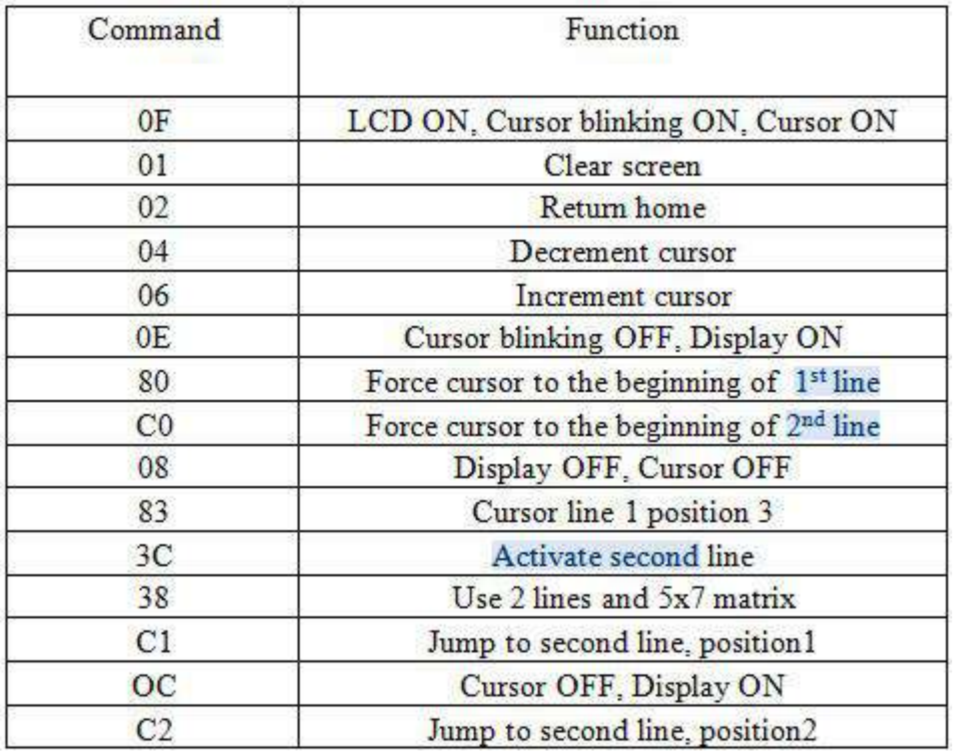
INFERENCE:

Experiment No:18 **<u>INTERFACING OF LCD WITH 8051</u>**
Date:

AIM: Write an assembly language program to display a message in LCD display

THEORY:

16×2 LCD module is a very common type of LCD module that is used in 8051 based embedded projects. It consists of 16 rows and 2 columns of 5×7 or 5×8 LCD dot matrices. The module were are talking about here is type number JHD162A which is a very popular one . It is available in a 16 pin package with back light ,contrast adjustment function and each dot matrix has 5×8 dot resolution.



Command	Function
0F	LCD ON, Cursor blinking ON, Cursor ON
01	Clear screen
02	Return home
04	Decrement cursor
06	Increment cursor
0E	Cursor blinking OFF, Display ON
80	Force cursor to the beginning of 1st line
C0	Force cursor to the beginning of 2nd line
08	Display OFF, Cursor OFF
83	Cursor line 1 position 3
3C	Activate second line
38	Use 2 lines and 5x7 matrix
C1	Jump to second line, position1
OC	Cursor OFF, Display ON
C2	Jump to second line, position2

LCD initialization

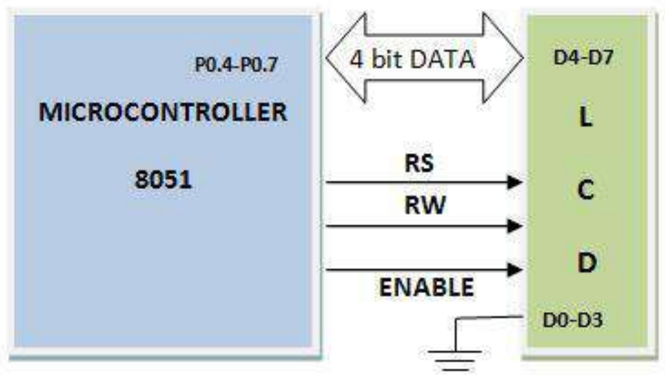
The steps that has to be done for initializing the LCD display is given below and these steps are common for almost all applications.

- Send 38H to the 8 bit data line for initialization
- Send 0FH for making LCD ON, cursor ON and cursor blinking ON.
- Send 06H for incrementing cursor position.
- Send 01H for clearing the display and return the cursor.

Sending data to the LCD

The steps for sending data to the LCD module is given below. I have already said that the LCD module has pins namely RS, R/W and E. It is the logic state of these pins that make the module to determine whether a given data input is a command or data to be displayed.

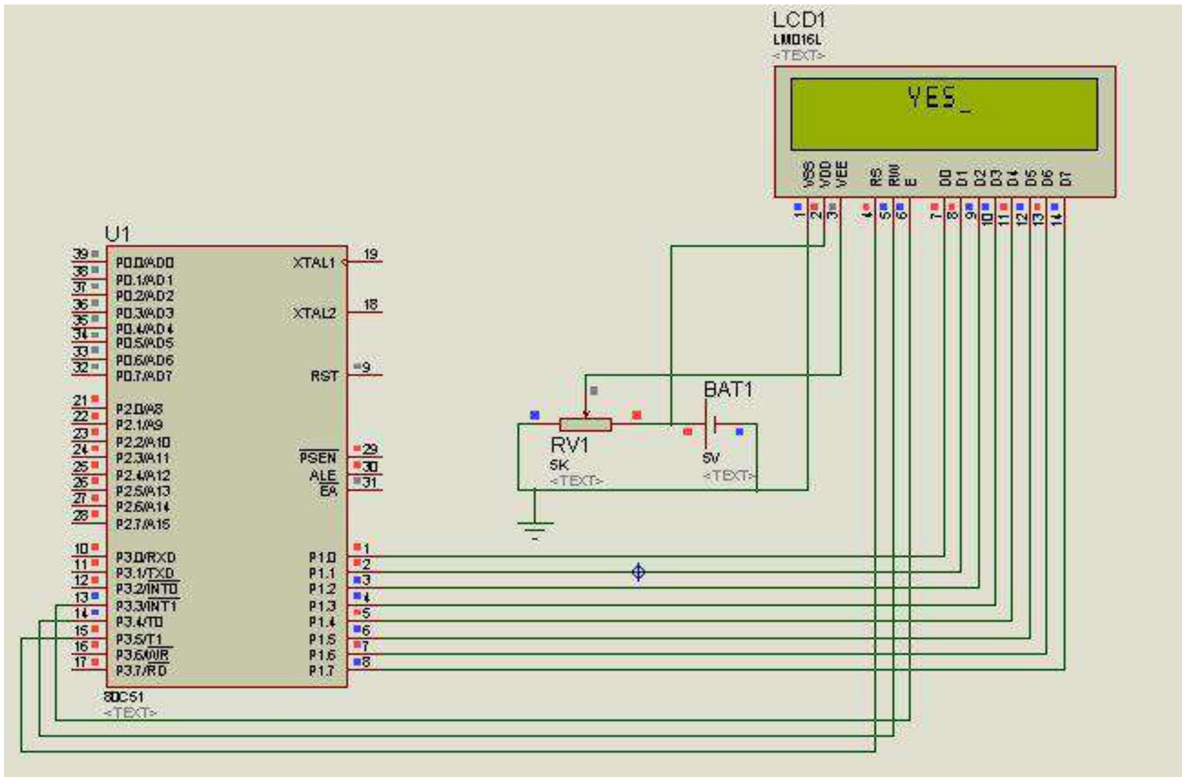
- Make R/W low.
- Make RS=0 if data byte is a command and make RS=1 if the data byte is a data to be displayed.
- Place data byte on the data register.
- Pulse E from high to low.
- Repeat above steps for sending another data.



PROGRAM:

```
ORG 0000H
MOV A,#38H
ACALL COMNWRT
ACALL DELAY
MOV A,#0EH
ACALL COMNWRT
ACALL DELAY
MOV A,#01
ACALL COMNWRT
ACALL DELAY
MOV A,#06H
ACALL COMNWRT
ACALL DELAY
MOV A,#86H
ACALL COMNWRT
ACALL DELAY
MOV A,#'Y'
ACALL DATAWRT
ACALL DELAY
MOV A,#'E'
 ACALL DATAWRT
ACALL DELAY
MOV A,#'S'
ACALL DATAWRT
AGAIN:SJMP AGAIN
COMNWRT:
        MOV P1,A
        CLR P3.5
        CLR P3.4
        SETB P3.3
        ACALL DELAY
        CLR P3.3
        RET
DATAWRT:
        MOV P1,A
        SETB P3.5
        CLR P3.4
        SETB P3.3
        ACALL DELAY
        CLR P3.3
        RET
```

SCHEMATIC DIAGRAM:



INFERENCE:

Experiment No:19 **INTERFACING OF EM RELAY WITH 8051**

Date:

AIM: Write an assembly language program to interface EM relay with 8051.

THEORY:

Relays are devices which allow low power circuits to switch a relatively high Current/Voltage ON/OFF. For a relay to operate a suitable pull-in & holding current should be passed through its coil. Generally relay coils are designed to operate from a particular voltage often its 5V or 12V. Also relays of various current ratings are available in market most commonly used relays is called as Cube relay with 12V coil and 5 Ampere rating. Also some relays with current rating of 30 Ampere are also available.

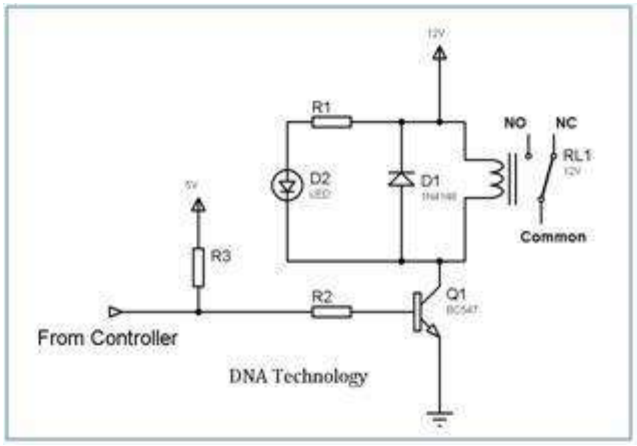


Figure shows the basic relay driver circuit. As you can see an NPN transistor BC547 is being used to control the relay. The transistor is driven into saturation (turned ON) when a LOGIC 1 is written on the PORT PIN thus turning ON the relay. The relay is turned OFF by writing LOGIC 0 on the port pin. A diode (1N4007/1N4148) is connected across the relay coil; this is done so as to protect the transistor from damage due to the BACK EMF generated in the relay's inductive coil when the transistor is turned OFF. When the transistor is switched OFF the energy stored in the inductor is dissipated through the diode & the internal resistance of the relay coil.

The LED is used to indicate that the RELAY has been turned ON. The resistor R1 defines the current flowing through the LED thereby defining the LED’s intensity.

Resistor R2 is used as a Series Base Resistor to set the base current. When working with 8051 controllers I have noted that it's not compulsory to use this resistor as the controller has internal 10k resistor which acts as a base resistor.

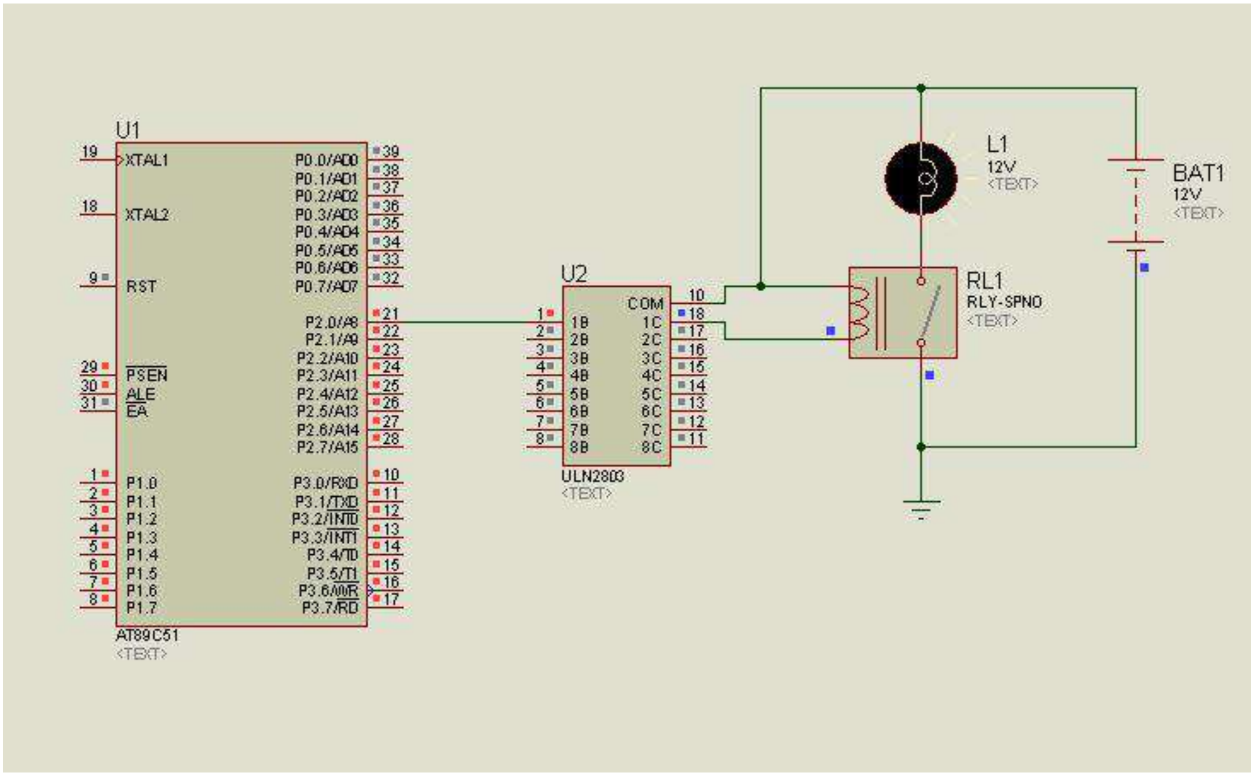
Microcontrollers have internal pull up resistors hence when a port pin is HIGH the output current flows through this internal pull up resistor. 8051 microcontrollers have an internal pull up of 10KΩ. Hence the maximum output current will be 5v/10k = 0.5ma. This current is not sufficient to drive the transistor into saturation and turn ON the relay. Hence an external pull up resistor R3 is used.

Let us now calculate the value of R3. Normally a relay requires a pull in current of 70ma to be turned ON. So our BC547 transistor will require enough base current to make sure it remains saturated and provide the necessary collector current i.e. 70ma. The gain (h_{fe}) of BC547 is 100 so we need to provide at least 70ma/100 = 0.7ma of base current. In practice you require roughly double the value of this current so we will calculate for 1.4ma of base current.

PROGRAM:

```
ORG 0000H
LOOP: SETB P2.0
ACALL DELAY
CLR P2.0
ACALL DELAY
 SJMP LOOP
DELAY: MOV R0,#100
AGAIN: MOV R1,#200
BACK:  DJNZ R1,BACK
DJNZ R0,AGAIN
RET
END
```

SCHEMATIC DIAGRAM:



INFERENCE: