

SETHU INSTITUTE OF TECHNOLOGY

(An Autonomous Institution | Accredited with 'A' Grade by NAAC)

PULLOOR, KARIAPATTI - 626 115



DEPARTMENT OF INFORMATIO TECHNOLOGY

PRACTICAL RECORD

SEMESTER: V

19UIT509 : MICROPORCESSOR BASED SYSTEM DESIGN LABORATORY

NAME _____

ROLL No. _____

REG. No. _____

DEPT. _____



SETHU INSTITUTE OF TECHNOLOGY

An Autonomous Institution

Pulloor, Kariapatti –Taluk, Virudhunagar Dist-626115.

Department of Information Technology

Accredited By NBA



PRACTICAL RECORD

BONAFIDE CERTIFICATE

Certified that this bonafide record of work done by

_____ in the _____ laboratory

during the year_____.

Staff-in charge

Head of the Department

Register Number

:

**Submitted for the practical
examination held on**

:

Internal Examiner

External Examiner

DEPARTMENT OF INFORMATION TECHNOLOGY

Vision

To promote excellence in producing competent IT Professionals to serve the society through technology and research

Mission

- Producing competent professionals in information and communication technologies.
- Educating the students with the state of art computing environment and pedagogical innovations
- Encouraging entrepreneurship and imparting skills for employability
- Establishing collaboration with IT and allied industries
- Promoting research in information and communication technology to improve the quality of human life
- Offering beneficial service to the society by imparting knowledge and providing IT solutions

CORE VALUES

- Quality
- Responsibility
- Novelty
- Team work
- Courtesy

PROGRAM EDUCATIONAL OBJECTIVES (PEO)

- Exhibit proficiency in analyzing, designing and developing IT based solutions to cater to the needs of the Industry {**Technical Competence**}
- Provide professional expertise to the industry and society with effective communication and ethics {**Professionalism**}
- Engage in lifelong learning for professional development and research {**Life-Long Learning**}

PROGRAM OUTCOMES (PO)

- a. Apply the knowledge of Mathematics, Basic Science, Computer and communication Fundamentals to solve complex problems in Information Technology. [**Engineering Knowledge**]
- b. Identify, formulate, review research literature and analyze complex problems reaching concrete conclusions using principles of mathematics , Engineering sciences and Information Technology[**Problem Analysis**]
- c. Design solution for complex information and communication engineering problems and design system components or processes that meet with realistic constraints for public health and safety, cultural, societal and environment considerations. [**Design/Development of Solutions**]
- d. Conduct investigations of complex Information technology related problems using research based knowledge and research methods including design of experiments,analysis and interpretation of data to provide valid conclusions through synthesis of information.[**Conduct investigations of complex problems**]
- e. Create, select and apply appropriate techniques, resources and modern IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations. [**Modern Tool Usage**]
- f. Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and consequent responsibilities relevant to professional engineering practice .[**The Engineer and Society**]
- g. Understand the impact of professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of and need for sustainable development.[**Environment and sustainability**]
- h. Apply ethical principles and commit to professional ethics and responsibilities through the norms of professional engineering practice .[**Ethics**]
- i. Function effectively as an individual and as a member or leader in diverse teams and in multidisciplinary settings.[**Individual and Team Work**]
- j. Communicate effectively with the engineering community and the society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations and give and receive clear instructions.[**Communication**]
- k. Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member /or leader in a team, to manage projects in multi-disciplinary environment. [**Project Management and Finance**]
- l. Recognize the need for, and have the preparation and ability to engage in independent and Life-long learning in broadest context of technological change. [**Life-long Learning**]

PROGRAM SPECIFIC OUTCOMES (PSO)

1. Design software solutions using programming skills and computing technologies
2. Design and implement data communication system using various IT compon

CO – PO MAPPING

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2
CO1														
CO2	3													3
CO3		3												3
CO4			3											3
CO5				3										3
CO6					3									3

COURSE OUTCOME

Co No.	Course Outcomes
CO1	Assess and solve basic binary math operations using the microprocessor
CO2	Apply knowledge and demonstrate programming proficiency using the various Basic configurations.
CO3	Compare accepted standards and guidelines to select appropriate Microprocessor and Microcontroller to meet specified performance requirements.
CO4	Design electrical circuitry to the Microprocessor I/O ports in order to interface the processor to external devices
CO5	Evaluate assembly language programs and download the machine code that will provide solutions real-world control problems.
CO6	Select appropriate machine tool for a cross assembler utility of a microprocessor and microcontroller.

LIST OF EXPERIMENTS

sno	Date	LISTOF EXPERIMENTS	MARKS	SIGNATURE
1.		Arithmetic Operations of two 8-bit numbers (Addition, Subtraction, Multiplication & Division). 8086 Microprocessor		
2.		Arranging an array of data (ascending order & descending order).		
3.		Code Conversion (BCD to HEX, HEX to BCD,).		
4.		Interfacing traffic light controller		
5.		Arithmetic Operations of two 8-bit numbers (Addition, Subtraction, Multiplication & Division) - 8051 Microcontroller.		
6.		Verify Timer/ Counter.		
7.		Verify Interrupt Handling.		
8.		Interfacing and programming of Stepper motor speed control using 8086		
9.		Arithmetic Operations of two 8-bit numbers (Addition, Subtraction, Multiplication & Division). ARM Processor		
10.		Code Conversion Using 8051 Microcontroller		
11.		Programming / interfacing experiments with IDE for 8051/PIC/MSP/Arduino/Raspberry Pi).		

Ex. No : 1

Date:

**ARITHMETIC OPERATIONS OF TWO 8-BIT NUMBERS (8086 Microprocessor)
(ADDITION, SUBTRACTION, MULTIPLICATION & DIVISION).**

Aim

To perform arithmetic operations (Addition, Subtraction, Multiplication & Division) of two 8 bit numbers using 8086

Algorithm

Source Code

OUTPUT
ADDITION

Reg	H	L
A	00	00
B	00	00
C	00	00
D	00	00

Reg	H	L
A	00	04
B	00	00
C	00	00
D	00	00

Reg	H	L
A	00	04
B	00	03
C	00	00
D	00	00

Reg	H	L
A	00	07
B	00	03
C	00	00
D	00	00

SUBTRACTION

Reg	H	L
A	00	00
B	00	00
C	00	00
D	00	00

Reg	H	L
A	00	04
B	00	00
C	00	00
D	00	00

Reg	H	L
A	00	04
B	00	03
C	00	00
D	00	00

Reg	H	L
A	00	01
B	00	03
C	00	00
D	00	00

MULTIPLICATION

Reg	H	L
A	00	00
B	00	00
C	00	00
D	00	00

Reg	H	L
A	00	06
B	00	00
C	00	00
D	00	00

Reg	H	L
A	00	0c
B	02	00
C	00	00
D	00	00

Reg	H	L
A	00	06
B	02	00
C	00	00
D	00	00

DIVISION

Reg	H	L
A	00	00
B	00	00
C	00	00
D	00	00

Reg	H	L
A	00	06
B	00	00
C	00	00
D	00	00

Reg	H	L
A	00	03
B	02	00
C	00	00
D	00	00

Reg	H	L
A	00	06
B	02	00
C	00	00
D	00	00

RESULT

Thus the program for arithmetic operations of two 8-bit numbers (8086 microprocessor) was executed and completed successfully

Ex. No : 2

Date:

**ARRANGING AN ARRAY OF DATA IN ASCENDING ORDER AND
DESCENDING ORDER**

Aim

To find the largest number and in an array of data using 8086 instruction set.

Algorithm

SOURCE CODE

ARRANGING AN ARRAY OF DATA IN ASCENDING ORDER IN 8086

```
MOV SI,1100H
MOV CL,[SI]
DEC CL
REPEAT;
MOV SI,1100H
MOV CH,[SI]
DEC CH
INC SI
REPCOM;
MOV AL,[SI]
INC SI
CMP AL,[SI]
JC AHEAD
XCHG AL,[SI]
XCHG AL,[SI-1]
AHEAD
DEC CH
JNZ REPCOM
DEC CL
JNZ REPEAT
HLT
```

ARRANGING AN ARRAY OF DATA IN DESCENDING ORDER IN 8086

```
MOV SI,1100h
MOV CL,[SI]
DEC CL
REPEAT:
MOV SI,1100h
MOV CH,[SI]
DEC CH
INC SI
REPCOM:
MOV AL,[SI]
INC SI
CMP AL,[SI]
JNC AHEAD
XCHG AL,[SI]
XCHG AL,[SI-1]
AHEAD:
DEC CH
JNZ REPCOM
DEC CL
JNZ REPEAT
HLT
```

OUTPUT

Random Access Memory																	
0100:1100		update		<input checked="" type="radio"/> table		<input type="radio"/> list											
0100:1100	07	AA	AA	22	11	44	BB	00-00	00	00	00	00	00	00	00	00	הדפסה
0100:1110	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	00
0100:1120	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	00
0100:1130	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	00
0100:1140	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	00
0100:1150	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	00
0100:1160	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	00
0100:1170	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	00

Random Access Memory																	
0100:1100		update		<input checked="" type="radio"/> table		<input type="radio"/> list											
0100:1100	07	11	22	44	AA	AA	BB	00-00	00	00	00	00	00	00	00	00	הדפסה
0100:1110	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	00
0100:1120	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	00
0100:1130	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	00
0100:1140	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	00
0100:1150	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	00
0100:1160	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	00
0100:1170	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	00

Result

Thus the program for arranging an array of data in ascending order and descending order was executed and verified successfully

Ex. No :3

Date:

**CODE CONVERSION (8086 Microprocessor)
(BCD TO HEX, HEX TO BCD).**

Aim

To convert two BCD numbers in memory to the equivalent HEX number using 8086 instruction set

Algorithm

Source Code

OUTPUT

BCD TO HEX

Input: 4150 : 02 (MSD)

4151 : 09 (LSD)

Output: 4152 : 1D H

HEX TO BCD

Input: 4150: FF

Output: 4151: 55 (LSB)

4152: 02 (MSB)

Result

Thus the program to convert BCD data to HEX data was executed and verified successfully

Ex. No :4

Date:

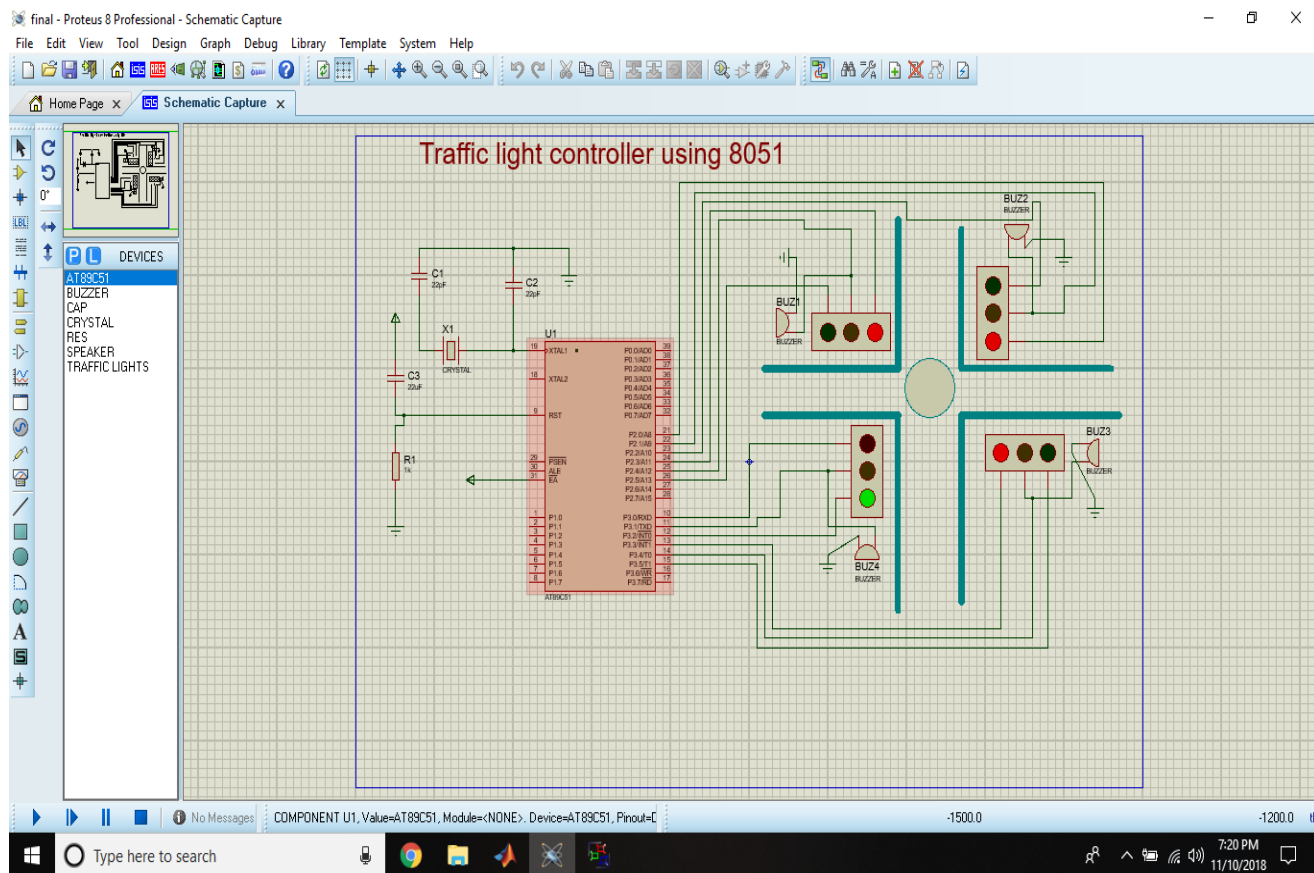
INTERFACING TRAFFIC LIGHT CONTROLLER

Aim

To control the traffic light controller using 8051 microcontroller

Algorithm

CIRCUIT DIAGRAM



Source code:

```

ORG 00H
MOV P2,#00H
MOV P3,#00H
MAIN:
    SETB P2.2
    SETB P3.2
    SETB P2.3
    SETB P3.3
    ACALL DELAY1
    SETB P2.4
    SETB P3.4
    CLR P2.3
    CLR P3.3
    ACALL DELAY2
    MOV P2,#00H
    MOV P3,#00H

    SETB P2.5
    SETB P3.5
    SETB P2.0
    SETB P3.0
    ACALL DELAY1
    SETB P2.1
    SETB P3.1
    CLR P2.0
    CLR P3.0
    ACALL DELAY2
    MOV P2,#00H
    MOV P3,#00H
    SJMP MAIN
DELAY1: MOV R0,#255D
H1: MOV R1,#255D
H2: MOV R2,#71D
H3: DJNZ R2,H3
    DJNZ R1,H2
    DJNZ R0,H1
    RET
DELAY2: MOV R0,#255D
H4: MOV R1,#142D
H5: MOV R2,#51D
H6: DJNZ R2,H6
    DJNZ R1, H5
    DJNZ R0, H4
    RET
END

```

Result

Thus the circuit diagram for Interfacing Traffic Light Controller was drawn and connected in proteous software and build successfully

Ex. No :5

Date:

**ARITHMETIC OPERATIONS OF TWO 8-BIT NUMBERS (8051 – Microcontroller)
(ADDITION, SUBTRACTION, MULTIPLICATION & DIVISION).**

AIM

To perform arithmetic operations (Addition, Subtraction, Multiplication & Division) of two 8 bit numbers using 8051 controller

ALGORITHM

SOURCE CODE

```
ORG 0000H  
MOV R1,#30H  
MOV R2,#20H  
MOV A,R1
```

```
ADD A,R2  
MOV R4,A  
CLR C
```


```
MOV A,R1  
SUBB A,R2  
MOV R5,A
```

```
MOV A,R1  
MOV B,R2  
MUL AB  
MOV R6,A
```


```
MOV A,R1  
MOV B,R2  
DIV AB  
MOV R7,A  
END
```


OUTPUT

Address:



C:0x0006:	FC	C3	E9	9A	FD	E9	8A	F0	A4	FE	E9	8A	F0	84	FF	00	00	00	00	00	00	00	00	00	00	00	00
C:0x001E:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C:0x0036:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C:0x004E:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C:0x0066:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

 Call Stack + Locals

 Memory 1

RESULT

Thus the Program for arithmetic operations of two 8-bit numbers (8051 – microcontroller) (addition, subtraction, multiplication & division) was executed successfully

Ex. No :6

Date:

VERIFY TIMER/ COUNTER.

Aim

To Write an ALP using 8051 for Timer / Counter and to Find the value for TMOD if we want to program Timer 0 in mode 2, use 8051 XTAL for the clock source, and use instructions to start and stop the timer.

Solution

TMOD= 0000 0010 Timer 0, mode 2,
 C/T = 0 to use XTAL clock source, and
 gate = 0 to use internal (software)
 start and stop method.

The role of the TMOD register, we will look at the timer's modes and how they are programmed to create a time delay.

Because modes 1 and 2 are so widely used, we describe each of them in detail.

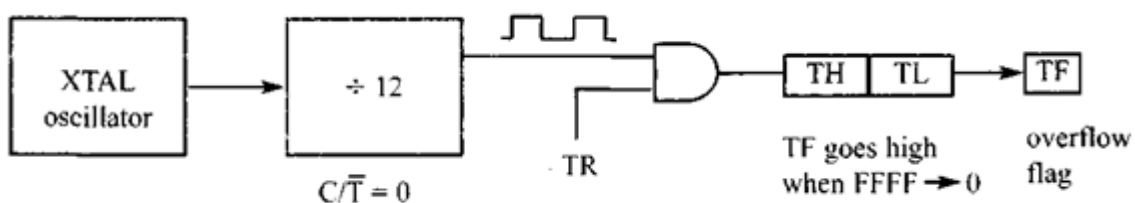
Mode 1 programming

The following are the characteristics and operations of mode 1:

1. It is a 16-bit timer; therefore, it allows values of 0000 to FFFFH to be loaded into the timer's registers TH and TL.
2. After TH and TL are loaded with a 16-bit initial value, the timer must be started. This is done by "SETB TRO" for Timer 0 and "SETB TR1" for Timer 1.
3. After the timer is started, it starts to count up. It counts up until it reaches its limit of FFFFH.
4. When it rolls over from FFFFH to 0000, it sets high a flag bit called TF (timer flag). This timer flag can be monitored. When this timer flag is raised, one option would be to stop the timer with the instructions "CLR TRO" or "CLR TR1", for Timer 0 and Timer 1, respectively.

Again, it must be noted that each timer has its own timer flag: TFO for Timer 0, and TF1 for Timer 1.

1. After the timer reaches its limit and rolls over, in order to repeat the process the registers
2. TH and TL must be reloaded with the original value, and TF must be reset to 0.



Steps to program in mode 1

To generate a time delay, using the timer's mode 1, the following steps are taken.

- Load the TMOD value register indicating which timer (Timer 0 or Timer 1) is to be used and which timer mode (0 or 1) is selected.
 1. Load registers TL and TH with initial count values.
 2. Start the timer.

- Keep monitoring the timer flag (TF) with the “JNB TFx, target” instruction to see if it is raised. Get out of the loop when TF becomes high.

3. Stop the timer.

4. Clear the TF flag for the next round.

5. Go back to Step 2 to load TH and TL again.

To calculate the exact time delay and the square wave frequency generated on pin PI .5, we need to know the XTAL frequency. See Example 9-5.

Develop a formula for delay calculations using mode 1 (16-bit) of the timer for a crystal frequency of XTAL = 11.0592 MHz. This is given in Figure 9-4. The scientific calculator in the Accessories directory of Microsoft Windows can help you to find the TH, TL values.

This calculator supports decimal, hex, and binary calculations.

(a) in hex

$(FFFF - YYXX + 1) \times 1.085 \text{ us}$ where YYXX are TH, TL initial values respectively.

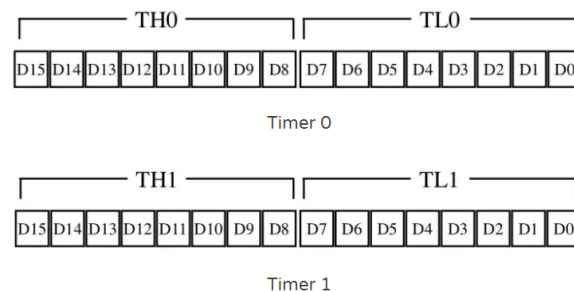
Notice that values YYXX are in hex.

(b) in decimal

Convert YYXX values of the TH,TL register to decimal to get a NNNNN decimal number, then $(65536 - NNNNN) \times 1.085 \text{ microsec}$

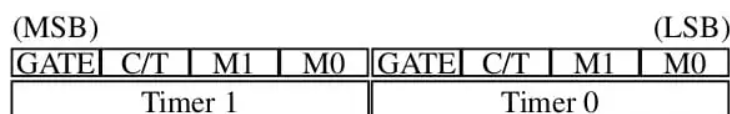
Timer Delay Calculation for XTAL = 11.0592 MHz

Timer



The 8051 uses two registers to handle and control the working of the Timer registers. One of these registers is TMOD. The other is TCON.

- **TMOD (Timer mode register)** – The timer registers can operate in four different modes. We’ll take a look at all these modes in a jiffy. But the point is, the selection of the functioning mode is done in this register. Additionally, the overall function of the timer registers, whether to work as timers or counters, is done here too.



Structure of TMOD register (Located at 89H-SFR space)

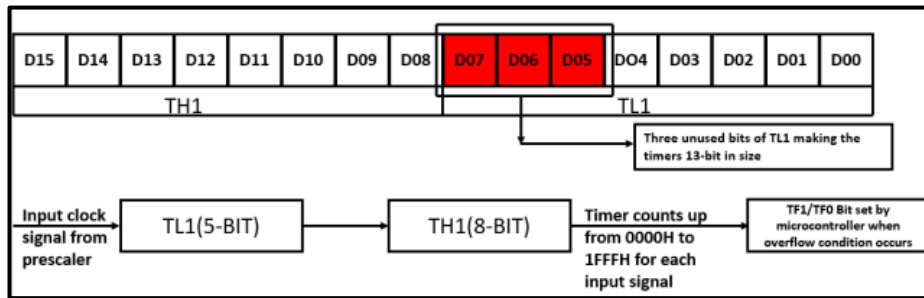
- **TCON (Timer Control Register)** – At the risk of a slight oversimplification that we will address later, we only use half of the TCON register for timing and counting purposes. This register is used to initialize the counting and to indicate when the timers have reached their counting limit.



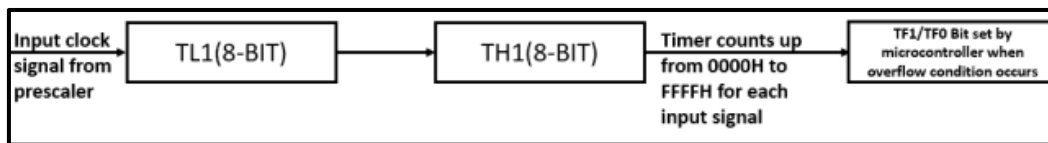
Structure of TCON register (Located at 88H-SFR space)

Timer modes in 8051 microcontrollers

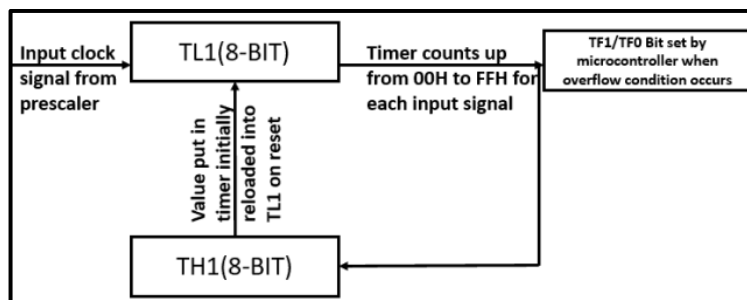
Mode 0:



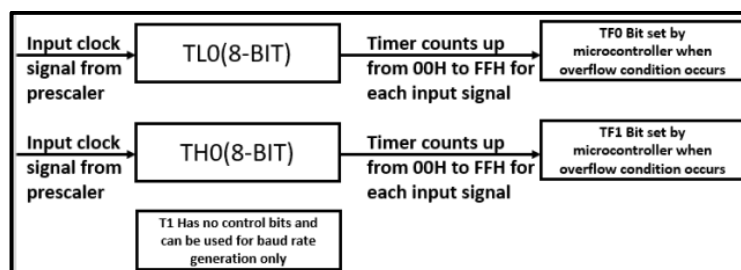
Mode 1:



Mode 2:



Mode 3:



RESULT

Thus the ALP using 8051 for Timer / Counter was completed and executed successfully

Ex. No :7

Date:

VERIFY INTERRUPT HANDLING

Aim

Write an ALP using 8051 for Interrupt Handling and to Find the amount of time delay in the DELAY subroutine generated by the timer. Assume that XTAL = 11.0592 MHz

Solution

```
                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
```

- Interrupts are the events that temporarily suspend the main program, pass the control to the external sources and execute their task. It then passes the control to the main program where it had left off.
- 8051 has 5 interrupt signals, i.e. INT0, TFO, INT1, TF1, RI/TI.
Each interrupt can be enabled or disabled by setting bits of the IE register and the whole interrupt system can be disabled by clearing the EA bit of the same register.

IE (INTERRUPT ENABLE) REGISTER

This register is responsible for enabling and disabling the interrupt.

EA register is set to one for enabling interrupts and set to 0 for disabling the interrupts.

Its bit sequence and their meanings are shown in the following figure.

EA	-	-	ES	ET1	EX1	ET0	EX0

EA	IE.7	It disables all interrupts. When EA = 0 no interrupt will be acknowledged and EA = 1 enables the interrupt individually.
-	IE.6	Reserved for future use.
-	IE.5	Reserved for future use.
ES	IE.4	Enables/disables serial port interrupt.
ET1	IE.3	Enables/disables timer1 overflow interrupt.
EX1	IE.2	Enables/disables external interrupt1.
ET0	IE.1	Enables/disables timer0 overflow interrupt.
EX0	IE.0	Enables/disables external interrupt0.

IP (INTERRUPT PRIORITY) REGISTER

We can change the priority levels of the interrupts by changing the corresponding bit in the

Interrupt Priority (IP) register as shown in the following figure.

- A low priority interrupt can only be interrupted by the high priority interrupt, but not interrupted by another low priority interrupt.
- If two interrupts of different priority levels are received simultaneously, the request of higher priority level is served.
- If the requests of the same priority levels are received simultaneously, then the internal polling sequence determines which request is to be serviced.

-	-	PT2	PS	PT1	PX1	PT0	PX0
bit7	bit6	bit5	bit4	bit3	bit2	bit1	

-	IP.6	Reserved for future use.
-	IP.5	Reserved for future use.
PS	IP.4	It defines the serial port interrupt priority level.
PT1	IP.3	It defines the timer interrupt of 1 priority.
PX1	IP.2	It defines the external interrupt priority level.
PT0	IP.1	It defines the timer0 interrupt priority level.
PX0	IP.0	It defines the external interrupt of 0 priority level.

TCON Register **TCON** register specifies the type of external interrupt to the microcontroller.

Result

Thus the program for verify interrupt handling was completed and executed successfully

Ex. No :8

Date:

INTERFACING AND PROGRAMMING OF STEPPER MOTOR SPEED CONTROL USING 8086

Aim

To Write an ALP using 8051 for Stepper motor interface using 8051 microcontroller

ALGORITHM

- A motor, in general, is a device that converts electrical energy into mechanical energy.
- As the name suggests, a stepper motor is a device that does the same task as above but, in steps.
- It is a brushless, synchronous electric motor that can divide a complete rotation into a number of steps. Stepper motors generally have a permanent magnet shaft (rotor), and it is surrounded by a stator.
- The best feature of this type of motor is that the motor's angular position can be accurately controlled without any feedback mechanism, as long as the motor is not oversized. (From a designer's point of view the sizing of the motor is essential, if the stepper motor is undersized then it will not be able to withstand load and if it is oversized then it becomes too expensive for its purpose.).
- Therefore, it works in a simple accurate open-loop system, where the output is directly dependent on the input.
- A stepper motor rotates at small angles to complete 360 degrees rotation, these small angles are called steps, hence the name Stepper Motor. Typically, a stepper motor consists of 200 steps.

$$200 \text{ Steps} = 360 \text{ degrees}$$

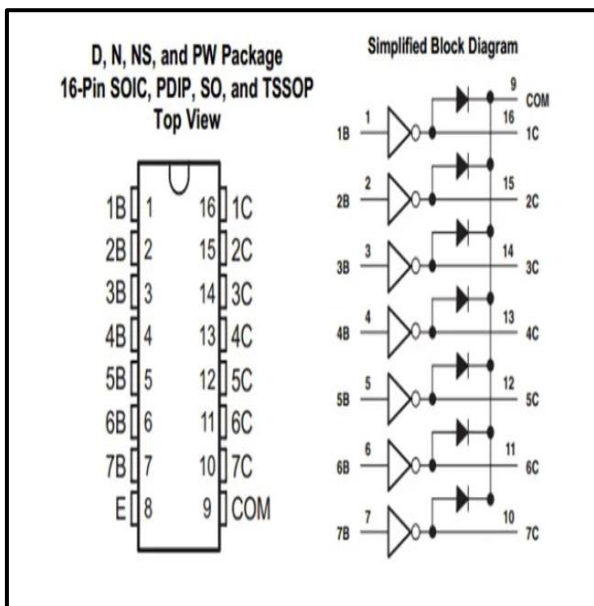
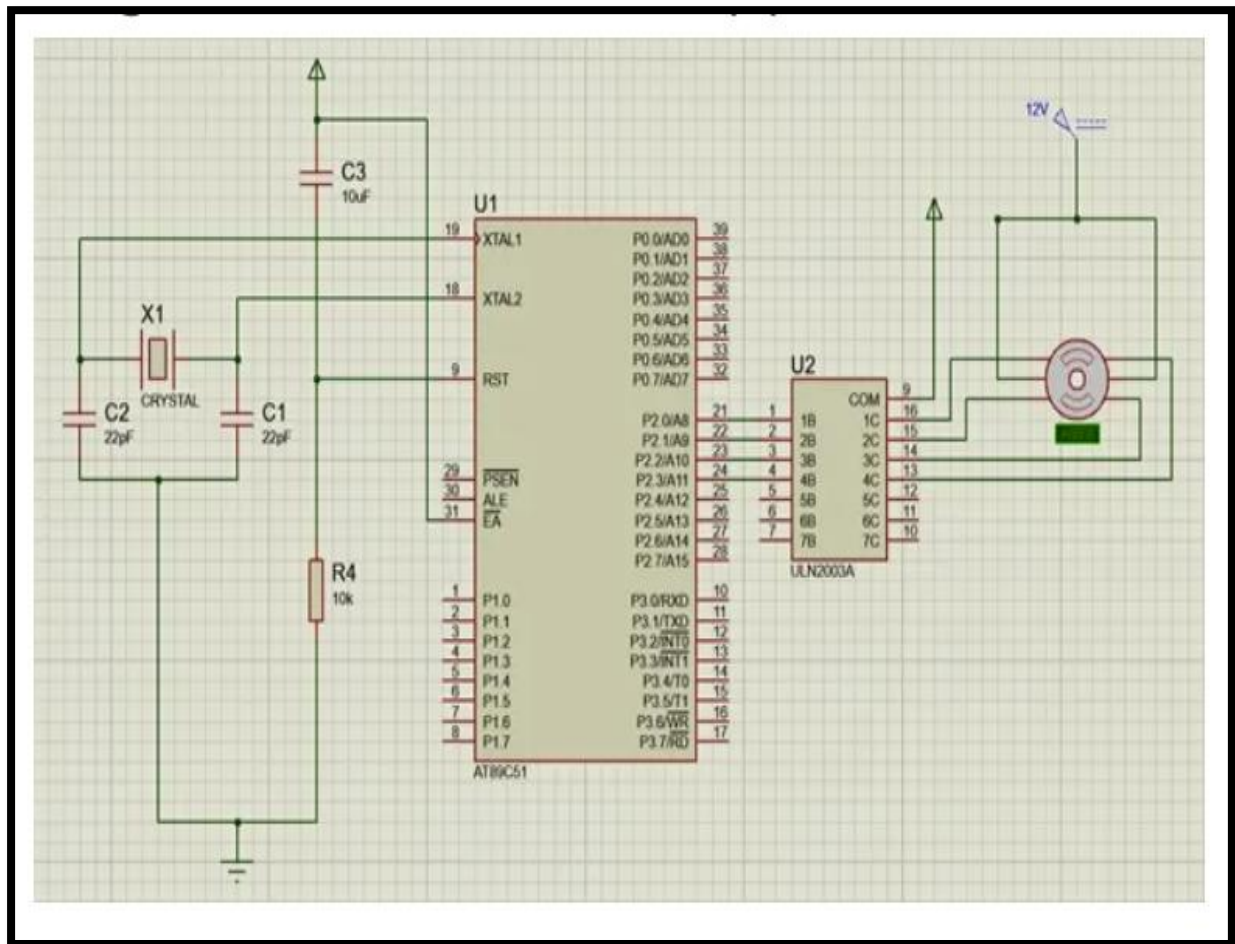
$$1 \text{ Step} = 'x' \text{ degrees}$$

$$x = 360 / 200 = 1.8 \text{ degree}$$

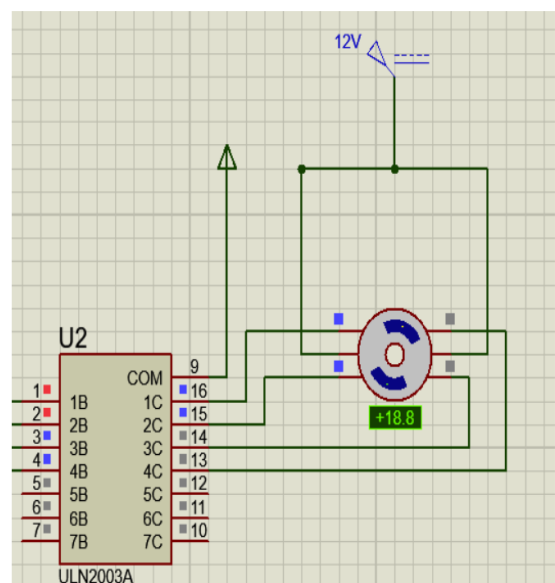
- Therefore, every step is 1.8 degrees.
- The high and low pulses are provided to the stator coil of the stepper motor.
- Here we're using a 4-coil stator. The angle of each step is decided by the steps in the rotor and its alignment with the stator.
- In the case of a 200-steps motor, the step angle is $360 / 200 = 1.8$ degrees, and in the case of an 8-steps rotor, the step angle is $360 / 8 = 45$ degrees. Here's a simulation.

COMPONENTS REQUIRED

Sr no.	Components Required	Description	Nos.
1	8051 Microcontroller	AT89S51/ AT89C51/Any other compatible variants	1
2	Stepper Motor	12V	1
3	Current Driver IC	ULN2003A	1
4	DC Power Supply	12V	1
5	Oscillator Crystal	12Mhz	1
6	Capacitor (1)	22pF	2
7	Capacitor (2)	10uF	1
8	Resistor/ Pot	10kohm	1
9	Connecting wires	As per the requirement	



200 Steps Rotor: Step angle = 1.8 degrees



SOURCE CODE

RESULT

Thus the circuit diagram for stepper motor was drawn and connected in proteous software and build successfully

Ex. No :9

Date:

**ARITHMETIC OPERATIONS OF TWO 8-BIT NUMBERS (ARM controller)
(ADDITION, SUBTRACTION, MULTIPLICATION, DIVISION).**

AIM

To perform arithmetic operations (Addition, Subtraction, Multiplication & Division) of two 8 bit numbers using ARM controller

ALGORITHM

SOURCE CODE

```
ENTER
    LDR R0,=0xFFFFFFFF0
    MOV R1,#0X00
    LDRB R1,[R0]
LOOP CMP R1,#0X64
    BLT NEXT
    SUB R1,R1,#0X64
    ADD R2,R2,#1
    B LOOP
NEXT CMP R1,#0X0A
    BLT LOOP1
    SUB R1,R1,#0X0A
    ADD R3,R3,#1
    B NEXT
LOOP1 ADD R1,R3,LSL#4
    STRB R1,[R0,#1]
    STRB R2,[R0,#2]
STOP B STOP
END
```

OUTPUT

RESULT

Ex. No :10

Date:

CODE CONVERSION USING 8051 MICROCONTROLLER

AIM

ALGORITHM

SOURCE CODE

```
        ENTER
        LDR R0,=0xFFFFFFFF
        MOV R1,#0X00
        LDRB R1,[R0]
LOOP    CMP R1,#0X64
        BLT NEXT
        SUB R1,R1,#0X64
        ADD R2,R2,#1
        B LOOP
NEXT    CMP R1,#0X0A
        BLT LOOP1
        SUB R1,R1,#0X0A
        ADD R3,R3,#1
        B NEXT
LOOP1   ADD R1,R3,LSL#4
        STRB R1,[R0,#1]
        STRB R2,[R0,#2]
STOP    B STOP
        END
```

OUTPUT

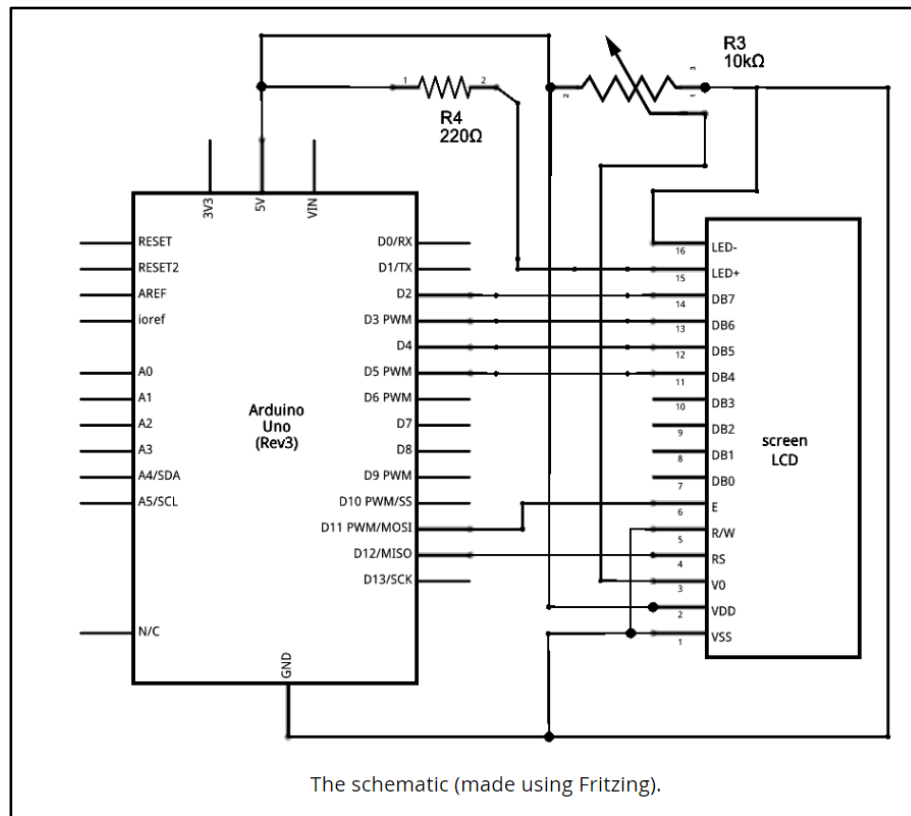
RESULT

AIM**SOLUTION**

- A **register select (RS) pin** that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.
- A **Read/Write (R/W) pin** that selects reading mode or writing mode
- An **Enable pin** that enables writing to the registers
- **8 data pins (D0 -D7)**. The states of these pins (high or low) are the bits that you're writing to a register when you write, or the values you're reading when you read.
- There's also a **display contrast pin (Vo)**, **power supply pins (+5V and GND)** and **LED Backlight (Bklt+ and Bklt-)** pins that you can use to power the LCD, control the display contrast, and turn on and off the LED backlight, respectively.
- The process of controlling the display involves putting the data that form the image of what you want to display into the data registers, then putting instructions in the instruction register. The LiquidCrystal Library simplifies this for you so you don't need to know the low-level instructions.
- The Hitachi-compatible LCDs can be controlled in two modes: 4-bit or 8-bit. The 4-bit mode requires seven I/O pins from the Arduino, while the 8-bit mode requires 11 pins. For displaying text on the screen, you can do most everything in 4-bit mode, so example shows how to control a 16x2 LCD in 4-bit mode.

HARDWARE REQUIRED

- Arduino Board
- LCD Screen (compatible with Hitachi HD44780 driver)
- pin headers to solder to the LCD display pins
- 10k ohm potentiometer
- 220 ohm resistor
- hook-up wires
- breadboard



SOURCE CODE

Hello World Example

This example sketch prints Hello World! to the LCD and shows the time in seconds since the Arduino was reset.

```
#include <LiquidCrystal.h>
```

```
// initialize the library by associating any needed LCD interface pin
```

```
// with the arduino pin number it is connected to
```

```
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
```

```
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
```

```
void setup() {
```

```
  // set up the LCD's number of columns and rows:
```

```
  lcd.begin(16, 2);
```

```
  // Print a message to the LCD.
```

```
  lcd.print("hello, world!");
```

```
}
```

```
void loop() {
```

```
  // set the cursor to column 0, line 1
```

```
  // (note: line 1 is the second row, since counting begins with 0):
```

```
  lcd.setCursor(0, 1);
```

```
  // print the number of seconds since reset:
```

```
  lcd.print(millis() / 1000);
```

```
}
```

RESULT