LABORATORY MANUAL

MICROPROCESSORS AND MICROCONTROLLERS LABORATORY

III B.Tech II-SEM (ECE)(MR18)

PREPARED BY:

MrsK.ANURADHA,ASSISTANT PROFESSOR

Department of Electronics & Communication Engineering

MALLAREDDY ENGINEERING COLLEGE
(AUTONOMOUS)
Affiliated to JNTUH, Approved by AICTE
Accredited by NAAC with 'A' grade, Reaccredited by NBA
Maisammaguda, Dhulpally, Secunderabad-500100

MALLA REDDY ENGINEERING COLLEGE

(AUTONOMOUS)

III Year B.Tech. ECE-I Sem

MICROPROCESSORS AND MICROCONTROLLERS LAB **Code:** 80417

Course Objectives: To introduce programming skills related to microcontrollers.

List of Experiments:

1 Arithmetic operations of 8-bit numbers using 8085.

2.Logical operations of 8-bit numbers using 8085.

3 a) Binary to BCD code conversions

b)BCD to Binary code conversions using 8085.

4.Arithmetic logical operations of 16 bit numbers using 8086 5.Programming using arithmetic, logical and bit manipulation instructions of 8051.

6.Program to toggle all the bits of Port P1 of 8051 continuously with 250 ms delay.

7.Program to interface seven segment display unit using 8051 8.Program to transmit/receive a message from Microcontroller to PC serially using RS232 using 8051

9.Program to interface Stepper Motor to rotate the motor in clockwise and anticlockwise directions using 8051

10.Program to interface a relay using 8051.

11.Program to interface LCD data pins to port P1 of 8051 and display a message on it.

12.Program for Traffic Light Controller using 8051

Software required: 1. GNU sim8085, MASM, Keil / µVision , Flash Magic

Course Outcomes:

At the end of the course, students will be able to

1.Understand and develop the 8085microprocessor based system 2.Able to program 8086microprocessor.

3.Interface different input &output devices to Microcontroller.

Experiment No: 01

Arithmetic operations of 8-bit numbers using 8085.

1 A ADDITION OF TWO 8 BIT NUMBERS

AIM

To perform addition of two 8 bit numbers using 8085.

ALGORITHM

1) Start the program by loading the first data into Accumulator.
2) Move the data to a register (B register).
3) Get the second data and load into Accumulator.
4) Add the two register contents.
5) Check for carry.
6) Store the value of sum and carry in memory location.
7) Terminate the program.

SOURCE CODE

	MVI	C, 00	Initialize C register to 00
	LDA	4150	Load the value to Accumulator.
	MOV	B, A	Move the content of Accumulator to B register.
	LDA	4151	Load the value to Accumulator.
	ADD	B	Add the value of register B to A
	JNC	LOOP	Jump on no carry.
	INR	C	Increment value of register C
LOOP:	STA	4152	Store the value of Accumulator (SUM).
	MOV	A, C	Move content of register C to Acc.
	STA	4153	Store the value of Accumulator (CARRY)
	HLT		Halt the program.

SAMPLE INPUT & OUTPUT

Input: 80 (4150)
80 (4251)

Output: 00 (4152)
01 (4153)

RESULT: Thus the program to add two 8-bit numbers was executed.

1 B SUBTRACTION OF TWO 8 BIT NUMBERS

AIM

To perform the subtraction of two 8 bit numbers using 8085.

ALGORITHM

1. Start the program by loading the first data into Accumulator.
2. Move the data to a register (B register).
3. Get the second data and load into Accumulator.
4. Subtract the two register contents.
5. Check for carry.
6. If carry is present take 2's complement of Accumulator.
7. Store the value of borrow in memory location.
8. Store the difference value (present in Accumulator) to a memory location
9. Terminate the program.

SOURCE CODE

	MVI	C, 00	Initialize C to 00
	LDA	4150	Load the value to Acc.
	MOV	B, A	Move the content of Acc to B register.
	LDA	4151	Load the value to Acc.
	SUB	B	Subtract the value of register B to A
	JNC	LOOP	Jump on no carry.
	CMA		Complement Accumulator contents.
	INR	A	Increment value in Accumulator.
	INR	C	Increment value in register C
LOOP:	STA	4152	Store the value of A-reg to memory address.
	MOV	A, C	Move contents of register C to Accumulator.
	STA	4153	Store the value of Accumulator memory
	HLT		address. Terminate the program.

SAMPLE INPUT & OUTPUT

Input:	06 (4150)
	02 (4251)
Output:	04 (4152)
	01 (4153)

RESULT : Thus the program to subtract two 8-bit numbers was executed.

1 C MULTIPLICATION OF TWO 8 BIT NUMBERS

AIM

To perform the multiplication of two 8 bit numbers using 8085.

ALGORITHM

1) Start the program by loading HL register pair with address of memory location.
2) Move the data to a register (B register).
3) Get the second data and load into Accumulator.
4) Add the two register contents.
5) Check for carry.
6) Increment the value of carry.
7) Check whether repeated addition is over and store the value of product and carry in memory location.
8) Terminate the program.

SOURCE CODE

	MVI MVI	D, 00 A, 00	Initialize register D to 00 Initialize Accumulator content to 00
	LXI	H, 4150	
	MOV	B, M	Get the first number in B - reg
	INX	H	
	MOV	C, M	Get the second number in C- reg.
LOOP:	ADD	B	Add content of A - reg to register B.
	JNC	NEXT	Jump on no carry to NEXT.
	INR	D	Increment content of register D
NEXT:	DCR	C	Decrement content of register C.
	JNZ	LOOP	Jump on no zero to address
	STA	4152	Store the result in Memory
	MOV	A, D	Move the content of D register to Accumulator
	STA	4153	Store the MSB of result in Memory
	HLT		Terminate the program.

SAMPLE INPUT &OUTPUT

Input: FF (4150)
FF (4151)

Output: 01 (4152)
FE (4153)

RESULT

Thus the program to multiply two 8-bit numbers was executed.

1 D DIVISION OF TWO 8 BIT NUMBERS

AIM

To perform the division of two 8 bit numbers using 8085

ALGORITHM

1) Start the program by loading HL register pair with address of memory location.
2) Move the data to a register (B register).
3) Get the second data and load into Accumulator.
4) Compare the two numbers to check for carry.
5) Subtract the two numbers.
6) Increment the value of carry.
7) Check whether repeated subtraction is over and store the value of product and carry in memory location.
8) Terminate the program.

SOURCE CODE

	LXI MOV MVI	H, 4150 B, M C, 00	 Get the dividend in B – reg. Clear C – reg for quotient
	INX	H	
	MOV	A, M	Get the divisor in A – reg.
NEXT:	CMP	B	Compare A - reg with register B.
	JC	LOOP	Jump on carry to LOOP
	SUB	B	Subtract A – reg from B- reg.
	INR	C	Increment content of register C.
	JMP	NEXT	Jump to NEXT
LOOP:	STA	4152	Store the remainder in Memory
	MOV	A, C	Move the Content of C register to Accumulator
	STA	4153	Store the quotient in memory
	HLT		Terminate the program.

SAMPLE INPUT & OUTPUT

Input: FF (4150)
FF (4251)

Output: 01 (4152) ---- Remainder
FE (4153)----- Quotient

RESULT

Thus the program to divide two 8-bit numbers was executed.

QUESTIONS RELATED TO THE NEXT EXPERIMENT:

1. What is XCHG instruction?
2. What is DAD instruction?
3. Explain about SBB instruction.
4. Explain about SPHL instruction.
5. Difference between SHLD and STA.

Experiment No: 02

Logical operations of 8-bit numbers using 8085.

AIM

To write an assembly language program on Logical operations of 8-bit numbers using 8085.

LDA 2050	A <- M[2050]
ANI 0F	A <- A (AND) 0F
MOV B, A	B <- A
LDA 2050	A <- M[2050]
ANI F0	A <- A (AND) F0
RLC	Rotate accumulator left by one bit without carry
RLC	Rotate accumulator left by one bit without carry
RLC	Rotate accumulator left by one bit without carry
RLC	Rotate accumulator left by one bit without carry
ANA B	A <- A (AND) B
STA 3050	M[3050] <- A
HLT	END

Experiment No: 03

BINARY TO BCD CODE CONVERSIONS

AIM

To write an assembly language program to convert an 8 bit binary data to BCD using 8085 microprocessor .

ALGORITHM

STEP 1: Start the microprocessor
STEP 2: Clear 'D' and 'E' register to account for hundred's and ten's load the binary data in Accumulator
STEP 3: Compare 'A' with 64 if cy = 01, go step C otherwise next step
STEP 4: Subtract 64 from (64+1) 'A' register
STEP 5: Increment 'E' register
STEP 6: Compare the register 'A' with '0A', if cy=1, go to step 11, otherwise next step
STEP 7: Subtract (0AH) from 'A' register
STEP 8: Increment D register Step 9 : Go to step 7
STEP 10: Combine the units and tens to from 8 bit result
STEP 11: Save the units, tens and hundred's in memory
STEP 12 : Stop the program execution

SOURCE CODE:

	MVI	E,00
	MOV	D,E
	LDA	4200
HUND	CPI	64
	JC	TEN
	SUI	64
	INR	E
	JMP	HUND
TEN	CPI	0A
	JC	UNIT
	SUI	0A
	INR	D
	JMP	TEN
UNIT	MOV	4A
	MOV	A,D
	RLC	
	RLC	
	RLC	
	RLC	
	ADD	
	STA	
	HLT	

SAMPLE INPUTS & OUTPUTS

Input: 4200 : 54

Output: 4250 : 84

RESULT

Thus the binary to BCD conversion was executed successfully

BCD TO BINARY CODE CONVERSIONS

AIM:To write an assembly language program to convert BCD data to Binary data using 8085 microprocessor

ALGORITHM

STEP 1 : Start the microprocessor
STEP 2 : Get the BCD data in accumulator and save it in register 'E'
STEP 3 : Mark the lower nibble of BCD data in accumulator
STEP 4 : Rotate upper nibble to lower nibble and save it in register 'B'
STEP 5 : Clear the accumulator
STEP 6 : Move 0AH to 'C' register
STEP 7 : Add 'A' and 'B' register
STEP 8 : Decrement 'C' register. If zf = 0, go to step 7
STEP 9 : Save the product in 'B'
STEP 10 : Get the BCD data in accumulator from 'E' register and mark the upper nibble
STEP 11 : Add the units (A-ug) to product (B-ug)
STEP 12 : Store the binary value in memory
STEP 13 : End the program

SOURCE CODE

```
LDA 4200
MOV E,A
ANI F0
RLC
RLC
RLC
RLC
MOV B,A
XRA A
MVI C,0A REP
DCR C
JNZ
MOV B,A
MOV A,E
ANI 0F
ADD B
STA 4201
HLT
```

SAMPLE INPUTS & OUTPUTS

Input : 4200 : 84
Output: 4201 : 54

RESULT

Thus the BCD to binary conversion was executed successfully.

QUESTIONS RELATED TO THE NEXT EXPERIMENT:

1. What is a counter?
2. Explain how counters are used in loop instructions?
3. What is meant by time delay?
4. Explain how to calculate execution delay or delay sub-routine?
5. Difference between time delay in loop and nested loop?

Introduction to MASM:

MASM: (Microsoft assembler)

Run command prompt and go to Masm directory

i.e. C:\masm

Type the program by opening an editor using Edit command

i.e. C:\masm\edit filename.asm

After typing the program assemble the program using masm command.

i.e. C:\masm\masm filename.asm;

After assembling, link the file using link command

i.e. C:\masm\link filename.obj;

Finally use debug command to execute the program.

C:\masm\debug filename .exe

-t; for single step execution

-g; for at a time execution

-I; for restarting the program execution

-d; to see the data segment

-q; to quit the execution

C:\masm\afdebug filename .exe

F1; for single step execution

g; for at a time execution

L filename .exe; to reload the program

Quit; to come out of the execute screen

Experiment No: 4

16 BIT ARITHMETIC OPERATIONS

Aim: To write an ALP to 8086 to perform 16-bit arithmetic operations in various Addressing modes.

Tools: PC installed with MASM/TASM

Program:

```
.MODEL SMALL
.STACK 42H
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
OPR1  DW   4269H
OPR2  DW   1000H
ADDRES DW  ?
SUBRES  DW  ?
MULRESLW  DW  ?
MULRESHW  DW  ?
DIVQ  DW  ?
DATA  ENDS
CODE SEGMENT
START     MOV   AX, DATA
          MOV   DS, AX
          MOV   AX, 4269H          ;Immediate addressing mode
          ADD   AX, OPR2           ;Direct addressing mode
          MOV   ADDRES, AX
          MOV   BX, OFFSET OPR1
          MOV   AX, [BX]           ;Register base addresing mode
          SUB   AX, OPR2
          MOV   SUBRES, AX
          MOV   AX, OPR1
          MOV   BX, OPR2
          MUL   BX                 ; Register addresiing mode
```

```
        MOV   MULRESLW, AX      ;Direct addressing mode
        MOV   MULRESHW, DX      ;Direct addressing mode
        MOV   SI, OFFSET DIVQ   ;Indexed addressing mode
        MOV   DX, 0000H
        MOV   AX, OPR1
        MOV   BX, OPR2
        DIV   BX
        MOV   [SI], AX
        MOV   [SI+2], DX
        INT   03H
    CODE ENDS
    END START
END
```

Result:

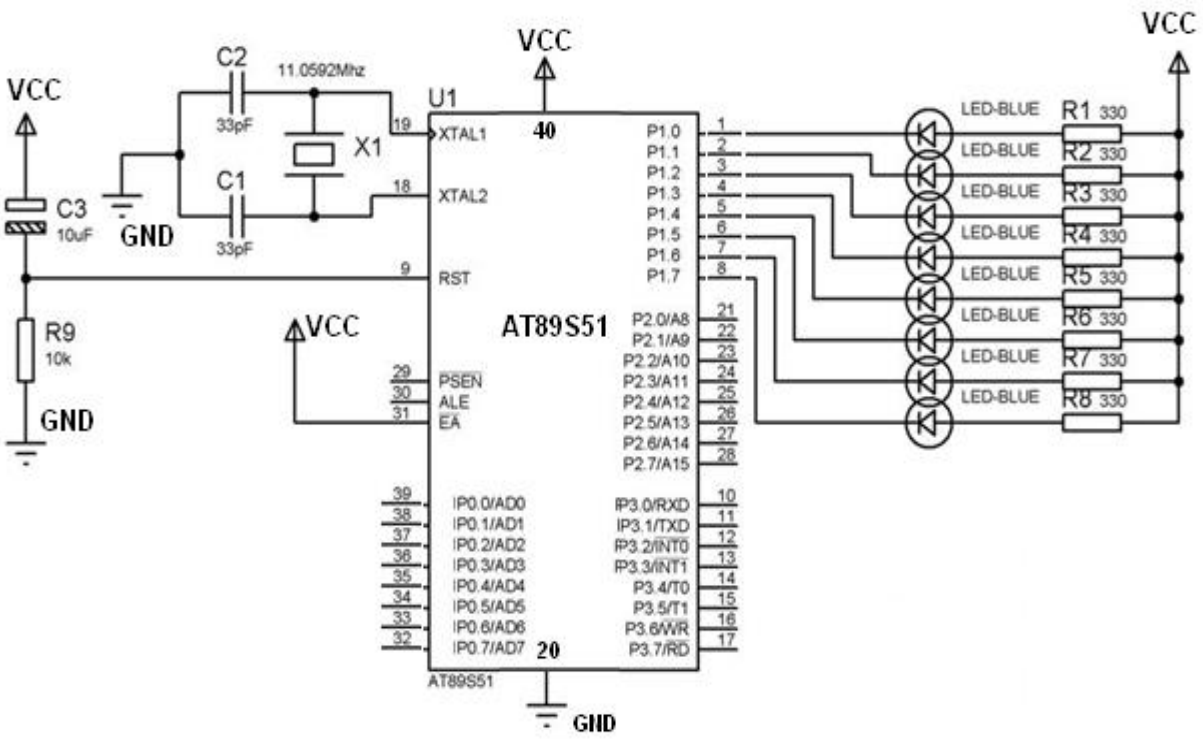
Experiment No: 5

5 Programming using arithmetic, logical and bit manipulation instructions of 8051

```
ORG 00H
MOV A,#56H
MOV B,#32H
ADD A,B
MOV R0,A
MOV A,#56H
SUBB A,B
MOV R1,A
MOV A,#56H
MUL AB
MOV R2,A
MOV R3,B
MOV A,#56H
DIV AB
MOV R4,A
MOV R5,B
CPL A
MOV R6,A
ANL A,B
MOV R7,A
ORL A,B
END
```

6 .Program to toggle all the bits of Port P1 of 8051 continuously with 250 ms delay.

```
ORG 0
MOV A,#00H                ;load A with 00H
BACK: MOV P1,A            ;send 00H to port1
ACALL DELAY               ;time delay
CPL A                     ;complement reg A
SJMP BACK                 ;keep doing this indefinitely
END                        ;end of asm file


DELAY:
        MOV  R5,#11
H3:     MOV  R4,#248
H2:     MOV  R3,#255
H1:     DJNZ R3,H1
        DJNZ R4,H2
        DJNZ R5,H3
        RET
        END
```

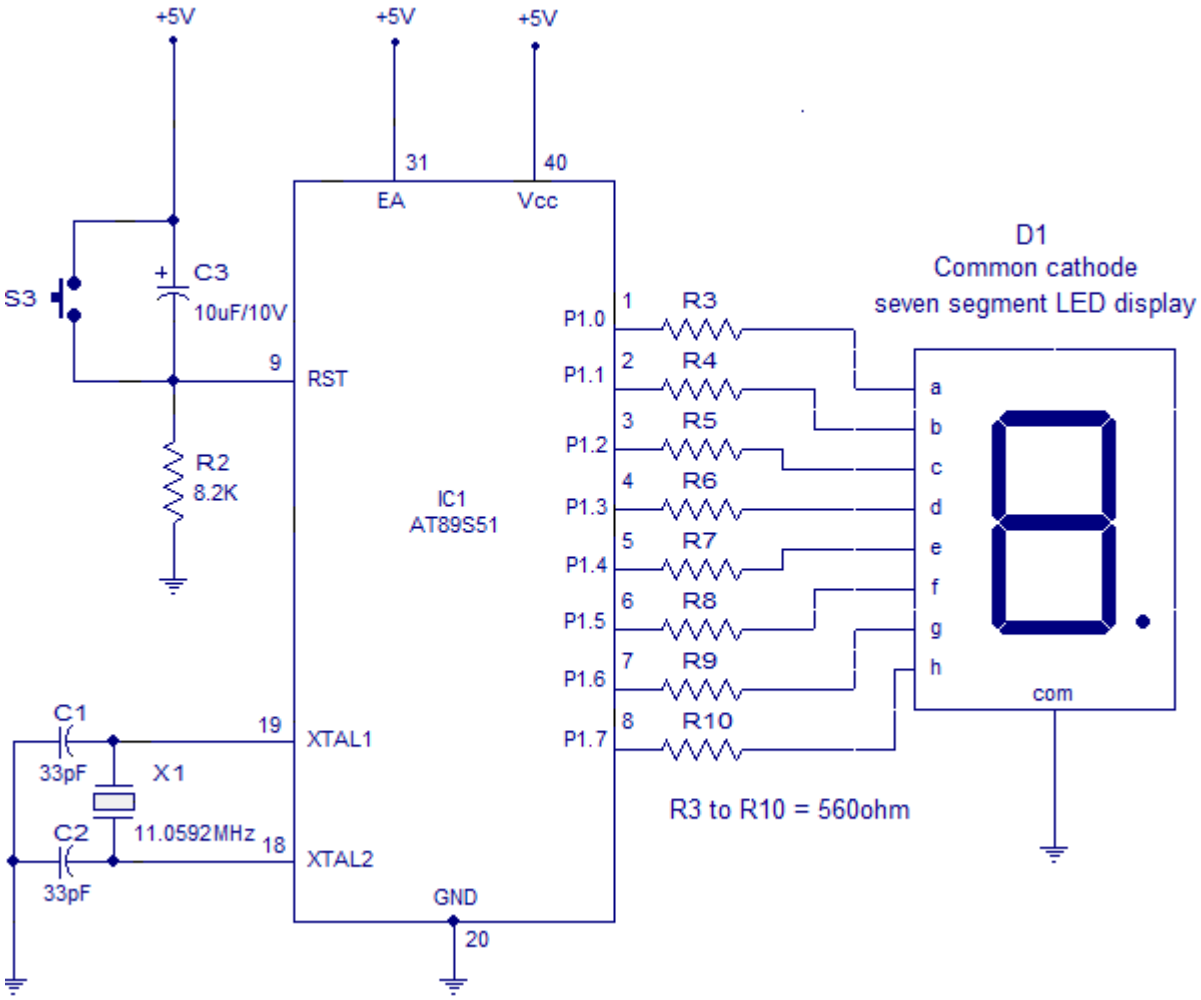


7 Program to interface seven segment display unit using 8051

```
ORG 000H                  //initial starting address
START: MOV A,#00001001B   // initial value of accumulator
MOV B,A
```

```
MOV R0,#0AH                 //Register R0 initialized as counter which counts from 10 to 0
LABEL: MOV A,B
INC A
MOV B,A
MOVC A,@A+PC                // adds the byte in A to the program counters address
MOV P1,A
ACALL DELAY                 // calls the delay of the timer
DEC R0                      //Counter R0 decremented by 1
MOV A,R0                    // R0 moved to accumulator to check if it is zero in next
instruction.
JZ START                    //Checks accumulator for zero and jumps to START. Done to
check if counting has       been finished.
SJMP LABEL
DB 3FH                      // digit drive pattern for 0
DB 06H                      // digit drive pattern for 1
DB 5BH                      // digit drive pattern for 2
DB 4FH                      // digit drive pattern for 3
DB 66H                      // digit drive pattern for 4
DB 6DH                      // digit drive pattern for 5
DB 7DH                      // digit drive pattern for 6
DB 07H                      // digit drive pattern for 7
DB 7FH                      // digit drive pattern for 8
DB 6FH                      // digit drive pattern for 9


DELAY: MOV R4,#05H          // subroutine for delay
WAIT1: MOV R3,#00H
WAIT2: MOV R2,#00H
WAIT3: DJNZ R2,WAIT3
DJNZ R3,WAIT2
DJNZ R4,WAIT1
RET
END
```

Interfacing seven segment display to 8051.



8. Program to transmit/receive a message from Microcontroller to PC serially using RS232 using 8051

Steps to send data serially:

1.Set baud rate by loading TMOD register with the value 20H, this indicating timer 1 in mode 2 (8-bit auto-reload) to set baud rate

2. The TH1 is loaded with proper values to set baud rate for serial data transfer

3. The SCON register is loaded with the value 50H, indicating serial mode 1, where an 8-bit data is framed with start and stop bits

4. TR1 is set to 1 to start timer 1

5. TI is cleared by CLR TI instruction

6. The character byte to be transferred serially is written into SBUF register

7. The TI flag bit is monitored with the use of instruction JNB TI,xx to see if the character has been transferred completely

8. To transfer the next byte, go to step 5

Program to transfer letter "D" serially at 9800baud, continuously:

```
MOV TMOD,#20H          ; timer 1,mode 2(auto reload)
MOV TH1, #-3           ; 9600 baud rate
MOV SCON, #50H         ; 8-bit, 1 stop,REN enabled
SETB TR1               ; start timer 1AGAIN:
MOV SBUF, #"D"         ; letter "D" to transfer
HERE: JNB TI, HERE     ; wait for the last bit
CLR TI                 ;clear TI for next char
SJMP AGAIN             ; keep sending A
```

Steps to receive data serially:

1. Set baud rate by loading TMOD register with the value 20H, this indicating timer 1 in mode 2 (8-bit auto-reload) to set baud rate

2. The TH1 is loaded with proper values to set baud rate

3. The SCON register is loaded with the value 50H, indicating serial mode 1, where an 8-bit data is framed with start and stop bits

4. TR1 is set to 1 to start timer 1

5. RI is cleared by CLR RI instruction

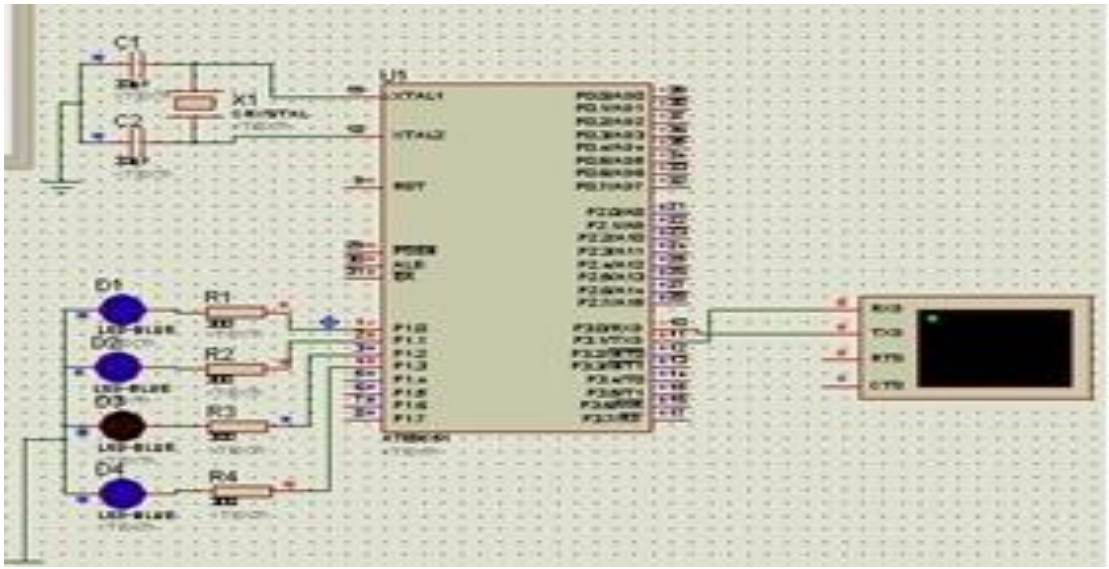
6. The RI flag bit is monitored with the use of instruction JNB RI,xx to see if an entire character has been received yet

7. When RI is raised, SBUF has the byte; its contents are moved into a safe place

8. To receive next character, go to step 5

Program to receive bytes of data serially, and put them in P2, set the baud rate at 9600, 8-bit data, and 1 stop bit:

```
MOV TMOD, #20H          ; timer 1,mode 2(auto reload)
MOV TH1, #-3            ; 9600 baud rate
MOV SCON, #50H          ; 8-bit, 1 stop, REN enabled
SETB TR1                ; start timer 1
HERE: JNB RI, HERE ;    wait for char to come in
MOV A, SBUF              ; saving incoming byte in A
MOV P2, A               ; send to port 2
CLR RI                  ; get ready to receive next byte
SJMP HERE               ; keep getting data
```



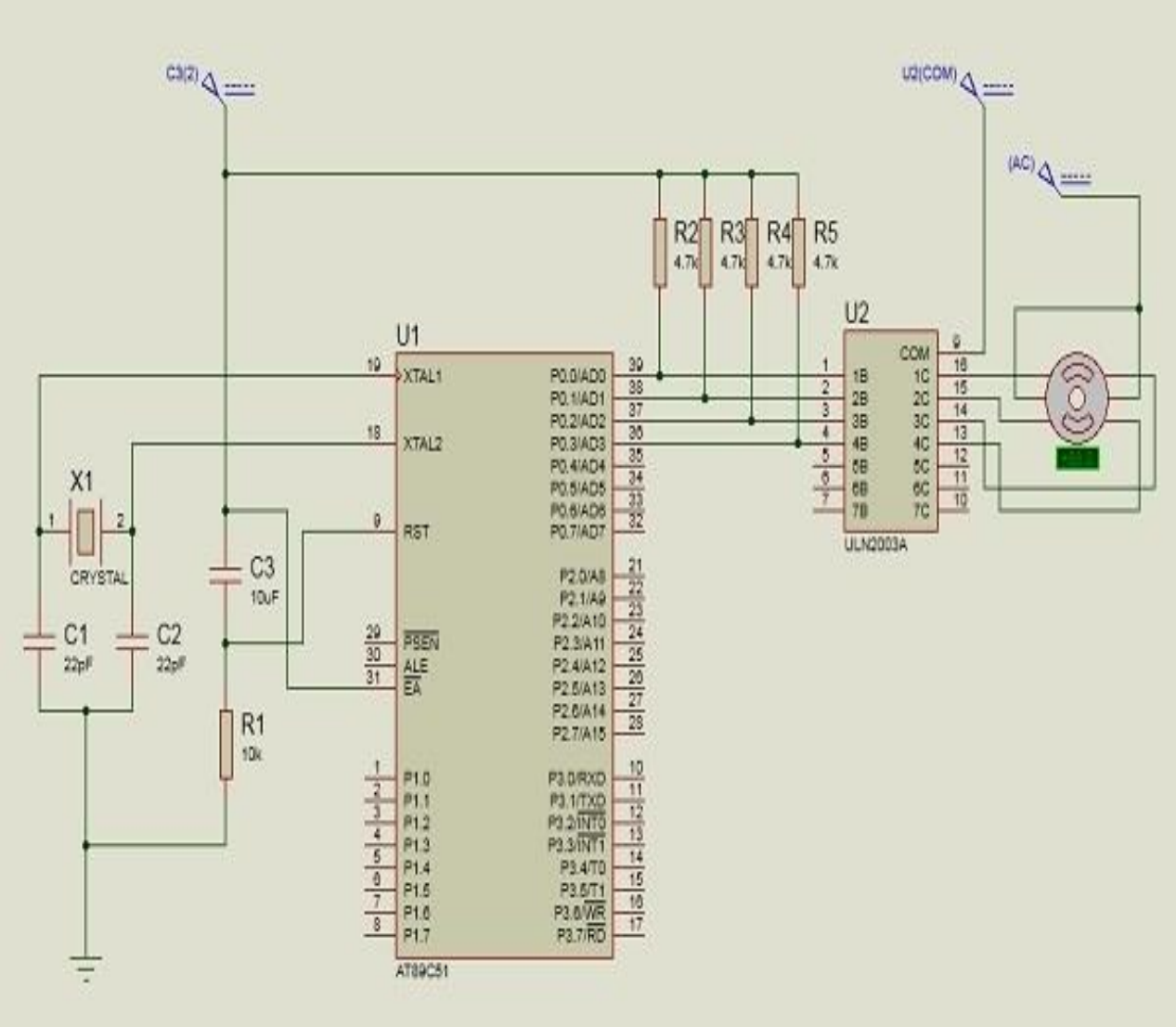
9 Program to interface Stepper Motor to rotate the motor in clockwise and anticlockwise directions using 8051

A **Stepper Motor to rotate the motor in clockwise directions**

```
                ORG 0000H

                MOV A, #66H          ; LOAD THE STEP SEQUENCE

BACK :          MOV P0, A            ; LOAD SEQUENCE TO PORT
                RR A                 ; CHANGE SEQUENCE ROTATE CLOCKWISE
                ACALL DELAY          ;WAIT FOR IT
                SJMP BACK            ; NOW KEEP GOING


DELAY:          MOV R2, #100

H1 :            MOV R3, #255

H2 :            DJNZ R3, H2

                DJNZ R2, H1
                RET
                END
```

B Stepper Motor to rotate the motor in anticlockwise directions

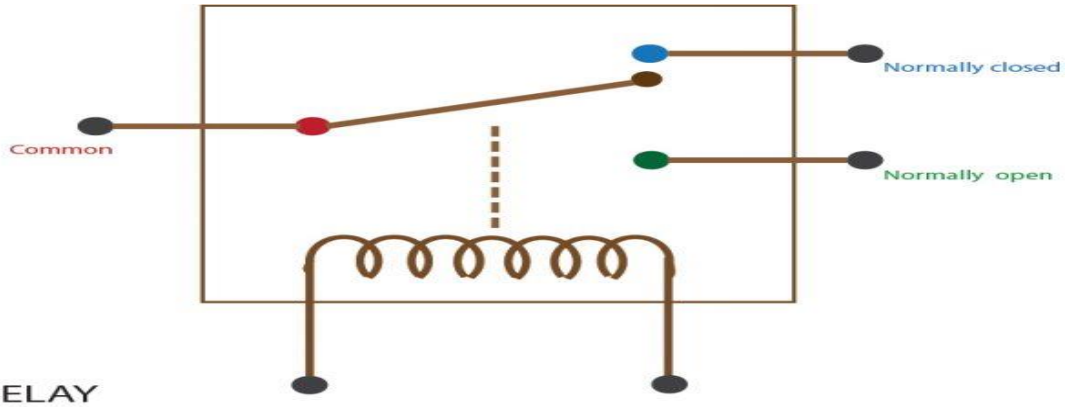
```
        ORG 0000H
                    MOV A, #66H         ; LOAD THE STEP SEQUENCE
BACK :              MOV P0, A           ; LOAD SEQUENCE TO PORT
                    RL A                ; CHANGE SEQUENCE ROTATE
        ANTICLOCKWISE
                    ACALL DELAY         ;WAIT FOR IT
                    SJMP BACK           ; NOW KEEP GOING

DELAY:              MOV R2, #100

H1 :                MOV R3, #255

H2 :                DJNZ R3, H2

                    DJNZ R2, H1
                    RET
                    END
```



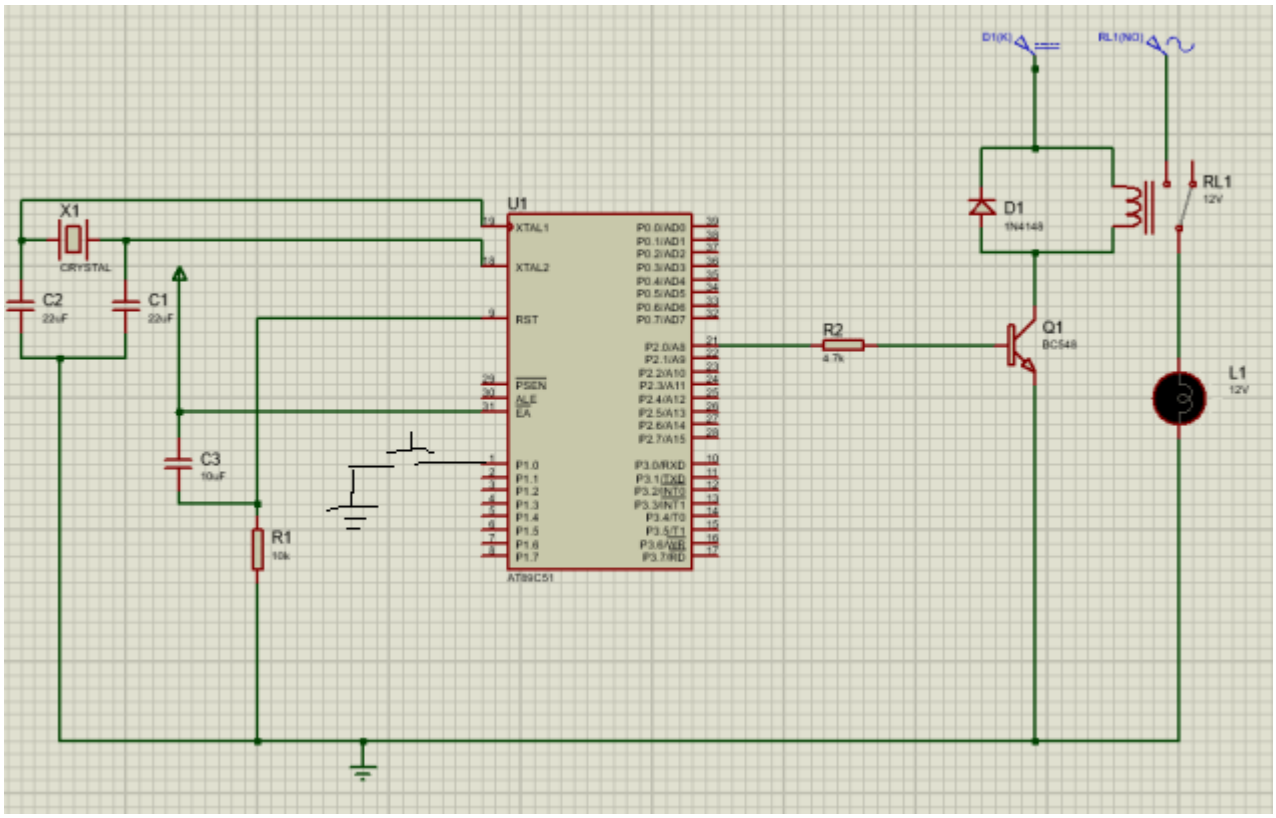
10 Program to interface a relay using 8051

```
RELAY     EQU   P2.0
SW        EQU   P1.0

ORG 0000H

MAIN:CLR RELAY                    ;Configure inp and outp
SETB SW
UP:  JNB SW,ON                    ;wait for switch to be pressed
CLR RELAY
ACALL DELAY
SJMP UP
ON:  SETB RELAY                   ;Turn ON relay
ACALL DELAY
ACALL DELAY
HERE:JB SW,HERE                   ;wait for switch to be released
CLR RELAY                 ;Turn OFF relay
ACALL DELAY
ACALL DELAY
SJMP UP                    ;Loop

DELAY:MOV R7,#0FFH    ;delay subroutine
AGAIN:MOV R6,#0FFH
DJNZ R6,$
DJNZ R7,AGAIN
RET
END
```

An electromechanical relay consists of three terminals namely common (COM), normally closed (NC) and normally opened (NO) contacts. These can either get opened or closed when the relay is in operation.



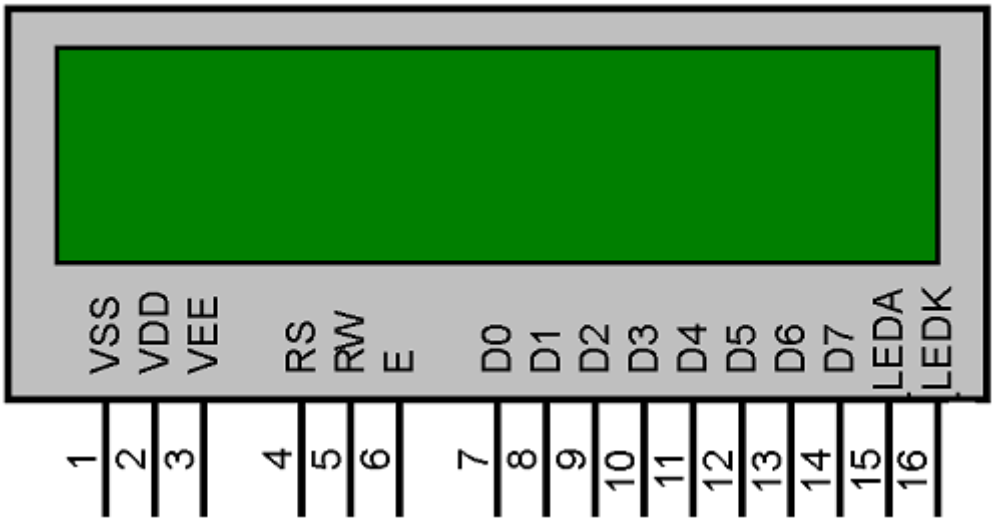
RELAY



11 Program to interface LCD data pins to port P! of 8051 and display a message on it

```
MOV A,#38H          // Use 2 lines and 5x7 matrix
ACALL CMND
MOV A,#0FH          // LCD ON, cursor ON, cursor blinking ON
ACALL CMND
MOV A,#01H          //Clear screen
ACALL CMND
MOV A,#06H          //Increment cursor
ACALL CMND
MOV A,#82H          //Cursor line one , position 2
ACALL CMND
MOV A,#3CH          //Activate second line
ACALL CMND
MOV A,#49D
ACALL DISP
MOV A,#54D
ACALL DISP
MOV A,#88D
ACALL DISP
MOV A,#50D
ACALL DISP
```

```
MOV A,#0C1H              //Jump to second line, position 1
ACALL CMND

MOV A,#67D
ACALL DISP
MOV A,#73D
ACALL DISP
MOV A,#82D
ACALL DISP

HERE: SJMP HERE

CMND: MOV P1,A
CLR P3.5
CLR P3.4
SETB P3.3
CLR P3.3
ACALL DELY
RET

DISP:MOV P1,A
SETB P3.5
CLR P3.4
SETB P3.3
CLR P3.3
ACALL DELY
RET

DELY: CLR P3.3
CLR P3.5
SETB P3.4
MOV P1,#0FFh
SETB P3.3
MOV A,P1
JB ACC.7,DELY

CLR P3.3
CLR P3.4
RET

END
```



Pin No:	Pin Name:	Description
1	Vss (Ground)	Ground pin connected to system ground
2	Vdd (+5 Volt)	Powers the LCD with +5V (4.7V – 5.3V)
3	VE (Contrast V)	Decides the contrast level of display. Grounded to get maximum contrast.
4	Register Select	Connected to Microcontroller to shift between command/data register
5	Read/Write	Used to read or write data. Normally grounded to write data to LCD
6	Enable	Connected to Microcontroller Pin and toggled between 1 and 0 for data acknowledgement
7	Data Pin 0	Data pins 0 to 7 forms a 8-bit data line. They can be connected to Microcontroller to send 8-bit data. These LCD's can also operate on 4-bit mode in such case Data pin 4,5,6 and 7 will be left free.
8	Data Pin 1	
9	Data Pin 2	
10	Data Pin 3	
11	Data Pin 4	
12	Data Pin 5	
13	Data Pin 6	
14	Data Pin 7	
15	LED Positive	Backlight LED pin positive terminal
16	LED Negative	Backlight LED pin negative terminal

RS (Register select)

A 16X2 LCD has two registers, namely, command and data. The register select is used to switch from one register to other. RS=0 for command register, whereas RS=1 for data register.

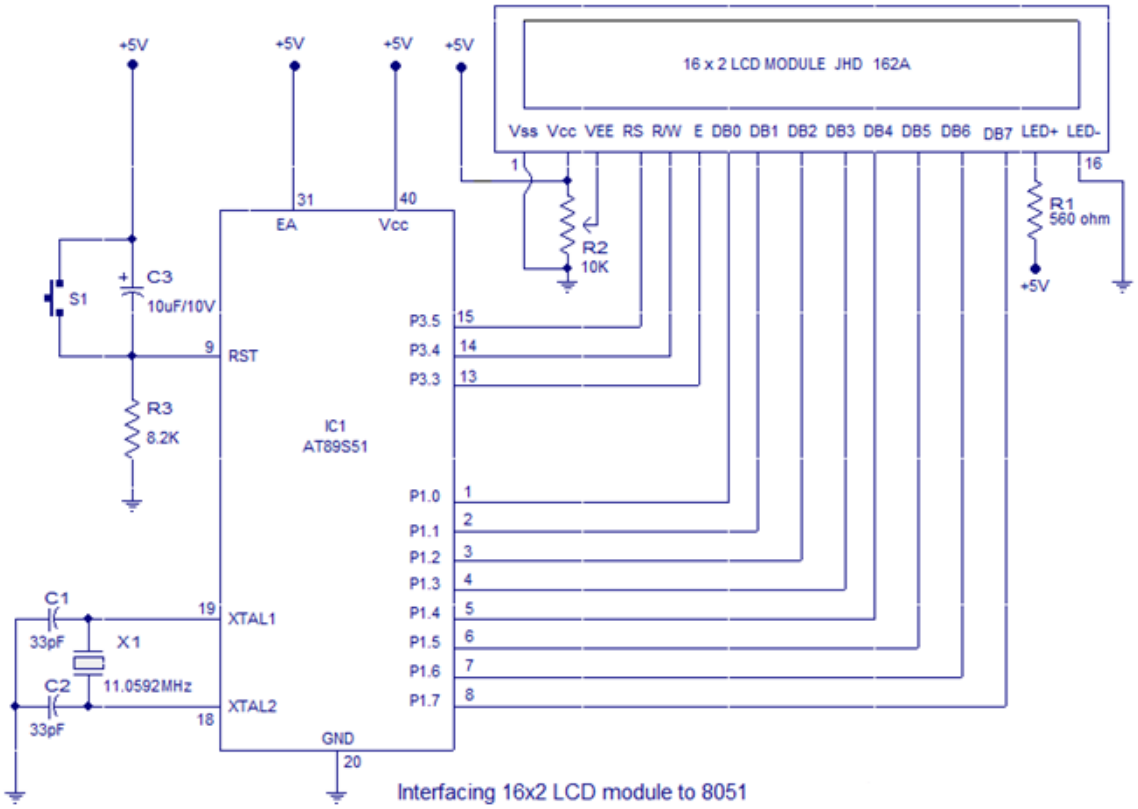
Command Register: The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task. Examples like:

- initializing it
- clearing its screen
- setting the cursor position
- controlling display etc.

Processing for commands happens in the command register.

Data Register: The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. When we send data to LCD it goes to the data register and is processed there. When RS=1, data register is selected.

Sr.No.	Hex Code	Command to LCD instruction Register
1	01	Clear display screen
2	02	Return home
3	04	Decrement cursor (shift cursor to left)
4	06	Increment cursor (shift cursor to right)
5	05	Shift display right
6	07	Shift display left
7	08	Display off, cursor off
8	0A	Display off, cursor on
9	0C	Display on, cursor off
10	0E	Display on, cursor blinking
11	0F	Display on, cursor blinking
12	10	Shift cursor position to left
13	14	Shift cursor position to right
14	18	Shift the entire display to the left
15	1C	Shift the entire display to the right
16	80	Force cursor to beginning (1st line)
17	C0	Force cursor to beginning (2nd line)
18	38	2 lines and 5×7 matrix



Interfacing 16x2 LCD module to 8051

12 Program for Traffic Light Controller using 8051

```
ORG OOH
```

```
HERE: MOV P0,#02H
      ACALL DELAY
      MOV P1,#01H
      MOV P2,#01H
      MOV  P3,#01H

      MOV P0,#04H
      ACALL DELAY1

      MOV P1,#02H
      ACALL DELAY
      MOV P0,#01H
      MOV P2,#01H
      MOV  P3,#01H

      MOV P1,#04H
      ACALL DELAY1

      MOV P2,#02H
      ACALL DELAY
      MOV P0,#01H
      MOV P1,#01H
      MOV  P3,#01H

      MOV P2,#04H
      ACALL DELAY1

      MOV P3,#02H
      ACALL DELAY
      MOV P0,#01H
      MOV P1,#01H
      MOV  P2,#01H
      MOV P3,#04H
      ACALL DELAY1
      SJMP HERE
      END

DELAY:
            MOV   R5,#11
      H3:   MOV   R4,#248
      H2:   MOV   R3,#255
      H1:   DJNZ  R3,H1
            DJNZ  R4,H2
            DJNZ  R5,H3
            RET
            END

DELAY1:
            MOV   R5,#02
      H2:   MOV   R3,#255
      H1:   DJNZ  R3,H1
```

```
DJNZ  R5,H2
RET
END
```