1st Edition

As Per C-16

MICROCONTROLLERS LAB STUDY MATERIAL

For

D.E.C.E Third Year (Fifth semester)

B.JAGADEESH M.TECH

LECTURER IN ECE

Department of Technical Education Andhra Pradesh

Price Rs: 75

Maanya's M.G.B **Publications**

Hyderabad: **Cell: 9290 4295 49**

Tirupati: **Cell: 9000 3050 79**

Microcontrollers Laboratory Study Material

First Edition: January 2019

FOR COPIES PLEASE CONTACT

Maanya's M.G.B Publications

Cell: 9000305079

Also Available at All Leading Book Shops

PREFACE

This book is intended as a laboratory study material for third year (5th Semester) students of Diploma in Electronics and communication Engineering and is written as per the Latest Syllabus (C-16) framed by State Board of Technical and Education and Training, Andhra Pradesh. This book consists of 2 Chapters; each chapter is systematic and well planned. This book offers a balanced exposition of "8051 instructions proper utilisation and interfacing with different input/output devices"

In the introduction, a step wise procedure is explained on how to use simulation software like PROTEUS and KEIL.

Chapter 1 related with group of programs which consists of Data transfer, Arithmetic, Logical instructions in addition to the utilisation of 8051 internal timers in producing required time delay.

Chapter 2 related with interfacing circuits like LED, Switch, 7segmentdisplays, Keypad, LCD display, and a DC motor.

Every program contains comments for each instruction for better understanding of the program.

I wish to thank **Sri P. SRINIVAS,** Head of Electronics &Communication Engineering, Govt.Polytechnic, Amadalavasa for his support and encouragement in presenting this book and his valuable suggestions enhanced the quality of the book.

I wish to thank **Dr.K.Narayana Rao**,Principal, Govt.Polytechnic, Amadalavasa for his support and encouragement in presenting this book. I acknowledge my sincere thanks to all my colleagues. Without their support this book would not have been possible.

I express my sincere thanks **Sri N.DHANANJAYA**, (Maanya's M.G.B Publications), for bringing out this book in a short time and pricing it moderately inspite of heavy cost of paper and printing.

We shall feel satisfied if the book meets the needs of the students for whom it is meant. Efforts have been made to present this book with error free text. Any suggestions or criticism are welcomed, and will be reflected in the next edition, if they are worth.

- B JAGADEESH

SYLLABUS

LIST OF EXPERIMENTS

I Familiarization with Microcontroller Kit & Simulators

1. To Work with microcontroller kits and Simulators
   a) Familiarize with 8051 Microcontroller Kit
   b) Familiarize with 8051 simulator KEIL (similar)

II . 8051 Instruction set

2. To Practice Arithmetic instructions of 8051
   a) Write an ALP to demonstrate Addition , subtraction , division and multiplication of 8 bit numbers .
   b) Write an ALP to Add and Subtract 16 bit numbers
   c) Write an ALP to find LCM of given 2 decimal numbers
3. To Practice Data transfer instructions
   a) Write an ALP to Block move - 10bytes of data from 0X30-0X39 to 0X40-0X49
   b) Write an ALP to Block exchange – 10bytes of data between 0X30-0X39 to 0X40-0X49
4. To Practice Data Manipulation
   a) To find Smallest/Largest number in 10bytes of data from 0X30-0X39 (R3 – should store the smallest/largest number and R4 – should store address of the smallest/largest number)
5. To Practice Boolean & Logical instructions :
   a) To Find 2's complement of a number using (CPL) instruction
   b) To Convert Packed to Unpacked BCD (bit Masking) Using (ANL) Instruction
   c) To convert Unpacked BCD to ASCII Using (ORL) instruction.

III. To implement Counters ,Timers

6. To implement a HEX up/down counter - (Program should check value @R0=0X30, if 0X30=0 then up counter else down counter)

7. To Implement Delays and Timers

To write a program in assembly language to produce required time delay a) by Using instructions only b) by Using Timers

IV .To practice Interfacing Techniques

8. Micro controller interfacing

a) Interfacing Switches and LEDS to 8051

i) To make an LED connected to port pin P1.5, light up for specific time on pressing a switch connected to port pin P2.3

ii) To Write a Program to make an LED connected to pin P1.7 to blink at a specific rate

9. To Interface 3-digit 7SEGMENT LED DISPLAY

a) To Interface a Single DOTMATRIX DISPLAY and display the given number

10. To Interface a (4x4 matrix) Key Board to 8051

11. To control the direction of rotation of a small DC motor

12. To burn executable code into flash memory for 89C51

LIST OF EXPERIMENTS	PAGE NO
1. Introduction	1
2. Steps to create and compile keil µvision-3/4 project	2
3. Steps to create and exexute hardware simulation on proteus	4
I. PROGRAMMING	
4. Write an ALP to demonstrate Addition, subtraction, division and multiplication of 8 bitnumbers.	10
5. Write an ALP to Add and Subtract 16 bit numbers	14
6. Write an ALP to find LCM of given 2 decimal numbers	19
7. Write an ALP to Block move - 10bytes of data from 0X30-0X39 to 0X40-0X49	23
8. Write an ALP to Block exchange – 10bytes of data between 0X30-0X39 to 0X40-0X49	27
9. Write an ALP to find Smallest/Largest number in 10bytes of data from 0X30-0X39 (R3 – should store the smallest/largest number and R4 – should store address of the smallest/largest number)	31
10. Write an ALP To Find 2‘s complement of a number using (CPL) instruction	36
11. Write an ALP to Convert Packed to Unpacked BCD (bit Masking) Using (ANL) Instruction	38
12. Write an ALP to convert Unpacked BCD to ASCII Using (ORL) instruction.	40
13. Write an ALP to produce required time delay a) Byusing instructions only b) By Using Timers	42
II. INTERFACING	
14. Interfacing Switches and LEDS to 8051 **a)** To make an LED connected to port pin P1.5, light up for specific time on pressing a switch connected to port pin P2.3 b) To Write a Program to make an LED connected to pin P1.7 to blink at a specific rate	44
15. Interface 4-digit 7SEGMENT LED DISPLAY and display given number	52
16. To Interface a Single DOTMATRIX DISPLAY and display the given number	56
17. Interface 16x2 LCD and display “WELCOME TO GPT MADALAVALASA”	60
18. Interface a (4x4 matrix) Key Board to 8051 and display the number pressed on a 63LCD display	66
19. Interface a small DC motor control the direction of rotation	73

1. Introduction

Earlier to Microcontrollers, Microprocessors were greatly used for each and every purpose. Microprocessors were containing ALU, general purpose register, stack pointer, program counter, clock counter and so many other features which the today's Microcontroller also possesses. But the difference between them exists with respect to the number of instructions, access times, size, reliability, PCB size and so on. Microprocessor contains large instruction set called as CISC processor whereas Microcontroller contains less number of instructions and is called as RISC processor. The access time is less in case of microcontrollers compared to microprocessors and the PCB size reduces in case of microcontrollers.

There are many versions of microcontrollers 8051, 80528751, AT8951 from Atmel Corporation and many more. In this manual we will study about the 8051 architecture, its features, programming and interfacing**.**

MCS 8051 is an 8-bit single chip microcontroller with many built-in functions and is the core for all MCS-51 devices.

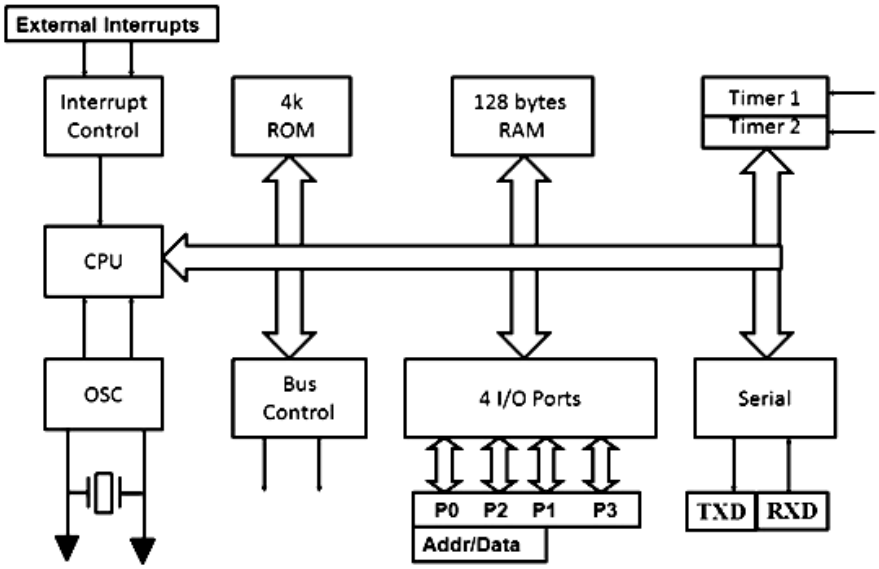


Fig.1. ***General Block Diagram of 8051 Microcontroller Architecture***

2. The main features of the 8051 core

- ♣ Operates with single Power Supply +5V.
- ♣ 8-bit CPU optimized for control applications.
- ♣ 16-bit program counter (PC) and 16-bit data pointer (DPTR).
- ♣ 8-bit program status word (PSW).
- ♣ 8-bit stack pointer (SP).
- ♣ 4K Bytes of On-Chip Program Memory (Internal ROM or EPROM).

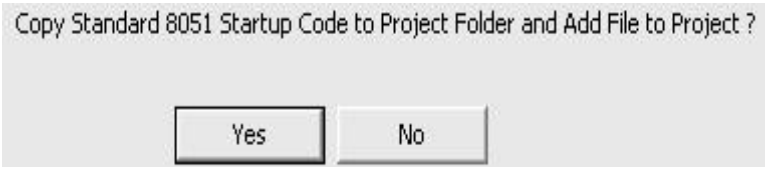
- 128 bytes of On-Chip Data Memory (Internal RAM):
  - Four Register Banks, each containing 8 registers (R0 to R7) [Total 32 reg]
  - 16-bytes of bit addressable memory.
  - 80 bytes of general-purpose data memory (Scratch Pad Area).
- Special Function Registers (SFR) to configure/operate microcontroller.
- 32 bit bi-directional I/O Lines (4 ports P0 to P3).
- Two 16-bit timers/counters (T0 and T1).
- Full duplex UART (Universal Asynchronous Receiver/Transmitter).
- On-Chip oscillator and clock circuitry.

3. Steps to create and Compile KEIL µVISION-3/4 Project

Step 1: Double Click on the µ Vision3/4 icon on the desktop. 

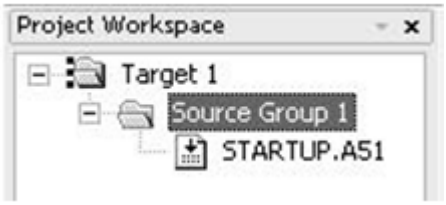
Step 2: Close any previous projects that were opened using – **Project -> Close**.

Step 3: Start **Project – New Project,** and select the CPU from the device database (Database-Atmel- AT89C51ED2 or AT89C51RD2 as per the board).On clicking **'OK'**, the following option is displayed. Choose **'No'**.

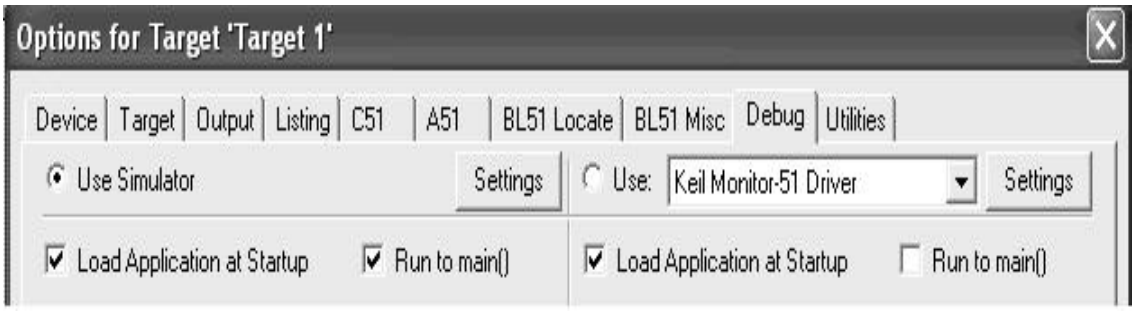


Step 4: Create a source file (using **File->New**), type in the assembly or C program and save this (filename**.asm**/filename**.c**) and add this source file to the project using either one of the following two methods. (i) **Project->Manage->Components, Environment Books->addfiles->** browse to the required file **-> OK**

"OR" ii) right click on the Source Group in the Project Window and the **Add Files to Group** option.

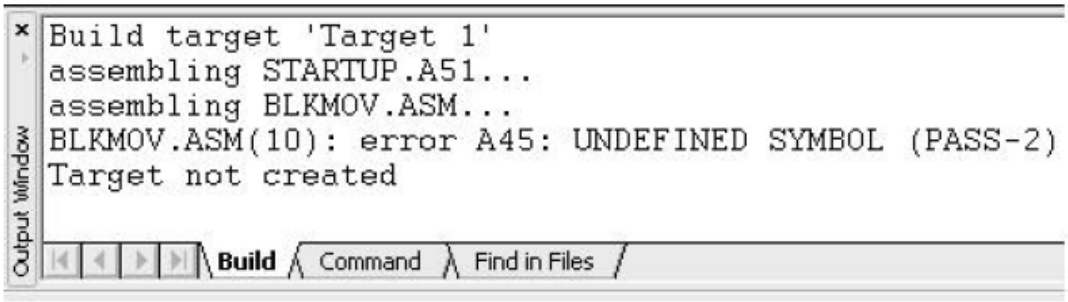


Step 5: Set the Target optionsusing -> **Project – Options for Target** opens the µ Vision2 **Options for Target – Target** configuration dialog. Set the **Xtal** (Crystal frequency) frequency as 11.0592 MHz, and also the **Options for Target – Debug – use either Simulator / Keil Monitor- 51 driver.B,**if you want to generate HEX file click on output tab in the below shown window and select create hex file.



Step 6: If **Keil Monitor- 51 driver is used click on Settings** -> COM Port settings select the COM Port to which the board is connected and select the baud rate as 19200 or 9600 (recommended). Enable **Serial Interrupt** option if the user application is not using on-chip UART, to stop program execution.

Step 7: Build the project; using **Project -> Build Project**. μVision translates all the user application and links. Any errors in the code are indicated by – "Target not created" in the Build window, along with the error line. Debug the errors. After an error free, to build go to Debug mode.



Step 8: Now user can enter into **Debug** mode with **Debug- Start / Stop Debug session** dialog. Or by clicking in the  icon.

Step 9: The program is run using the **Debug-Run** command & halted using **Debug-Stop Running.** Also the  (reset, run, halt) icons can be used. Additional icons are  (step, step over, and step into, run till cursor).

Step 10: If it is an interface program the outputs can be seen on the LCD, CRO, motor, led status, etc. If it is a other than interface program, the appropriate memory window is opened using View -> memory window (for data RAM & XRAM locations), Watch window (for timer program), serial window, etc.

Step 11: **Note:** To access data RAM area type address as D: 0020h. Similarly to access the DPTR region (XRAM-present on chip in AT89C51ED2) say 9000h location type in X: 09000H.

4. Steps to create and exexute hardware simulation on Proteus

Content:

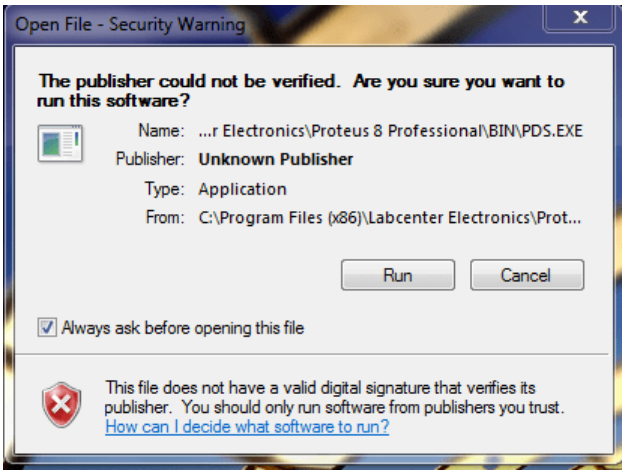
1: Start Proteus 8

2: Circuit Design

3: Circuit Simulation

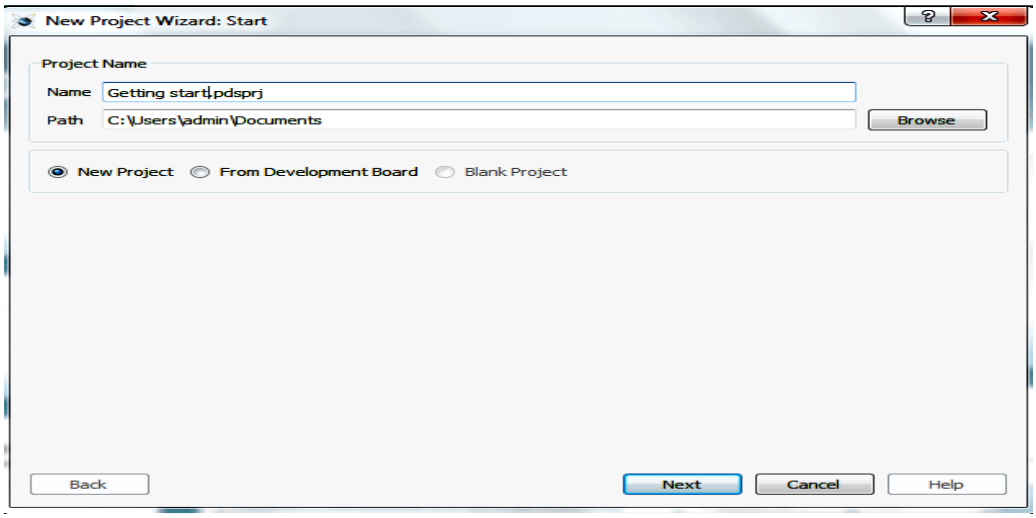
4.1. Start Proteus 8

- **Step 1**: Run the Proteus 8 professional software

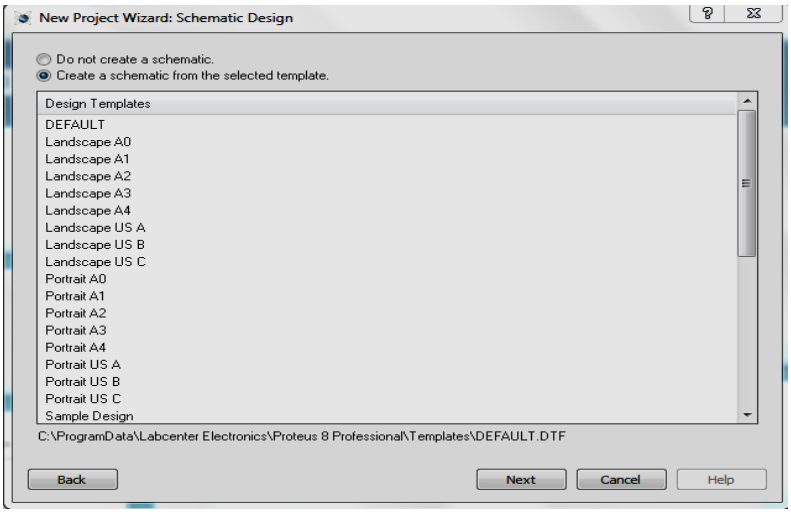


- **Step 2:** Next you see proteus design suit window click on new project option

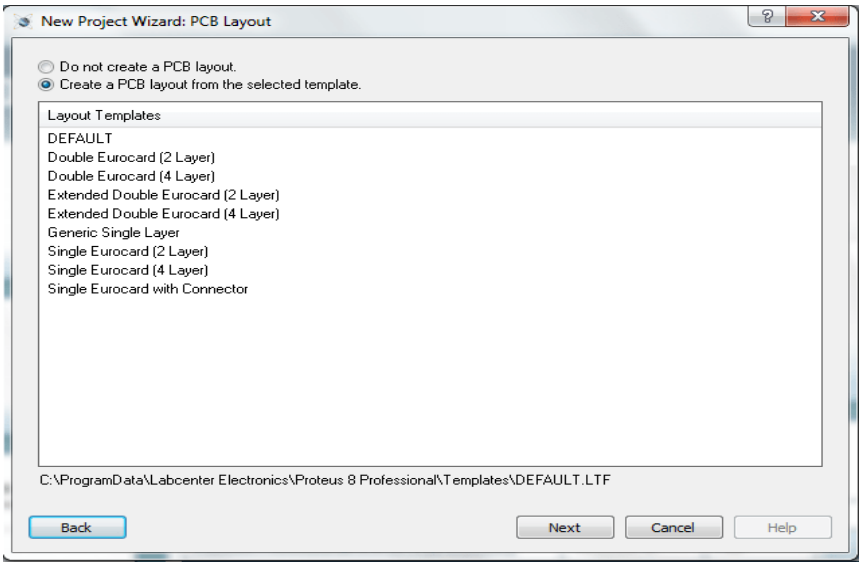
- Write project name and also select path for save the project file then click on next



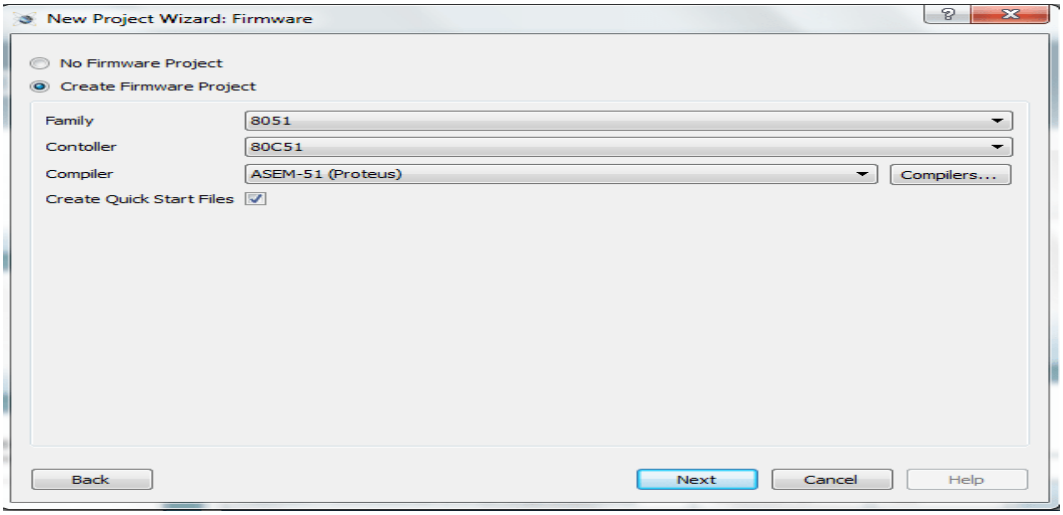
- **Step 3:** Next click on schematic from the selected templates if you don't want to schematic then click on 1st option
- Select default option and click on Next

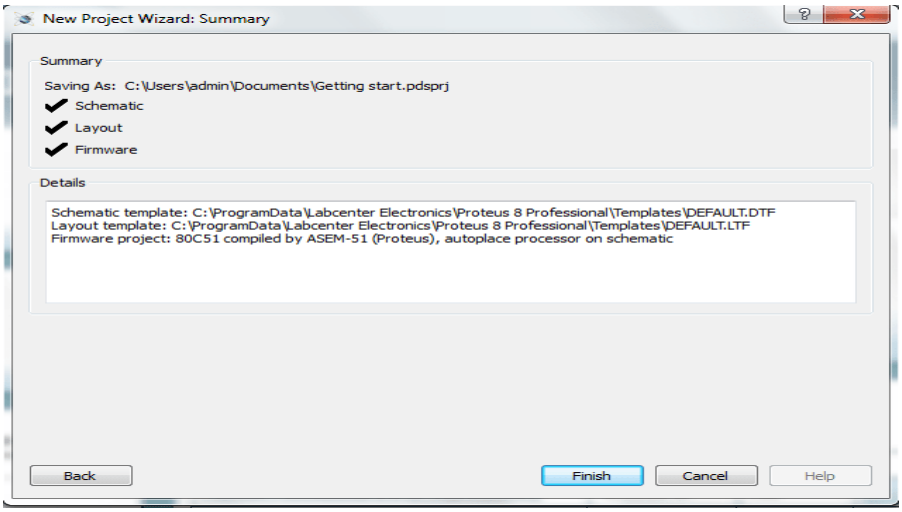


- **Step 4:** Next select create a PCB layout from selected templates.



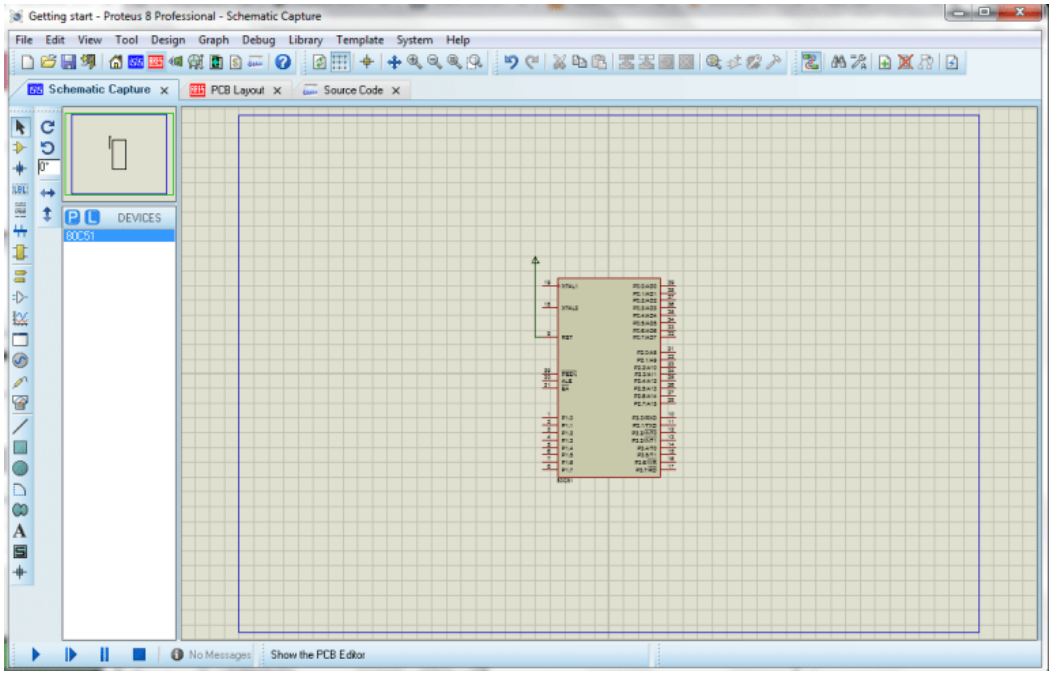
- If you don't want to create PCB then Select another option then select default and click on Next
- **Step 5:** If your Project design on microcontroller then select create firmware and also select microcontroller. else select 1 st option no firmware create. then click on Next finally created Schematic, layout and firmware click on finish.



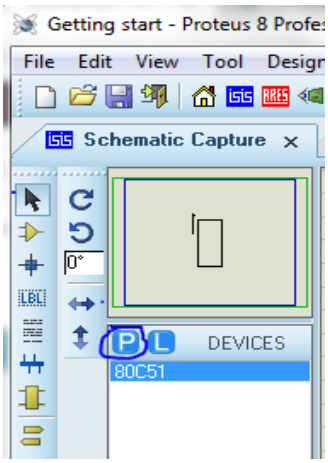


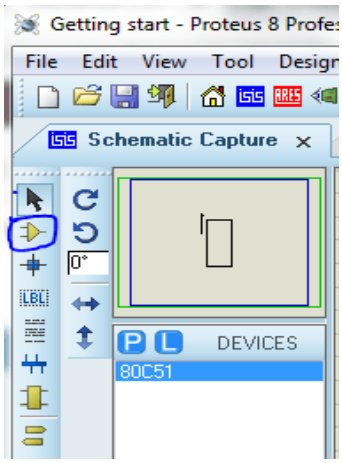
2. Circuit Design

- **Step 6:** Next you will be seen created schematic, layout and microcontroller firmware

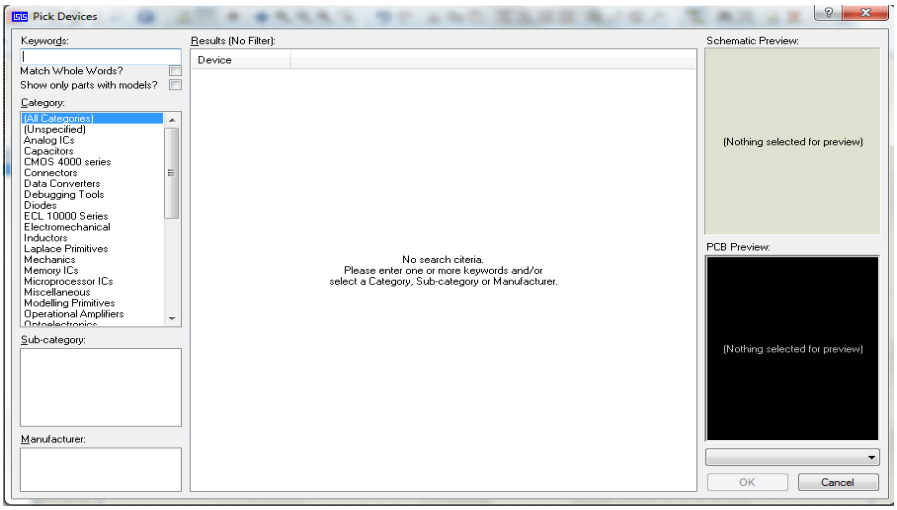


- **Step 7:** Next select the Components mode from left toolbar.
- Click on p(Pick from libraries)

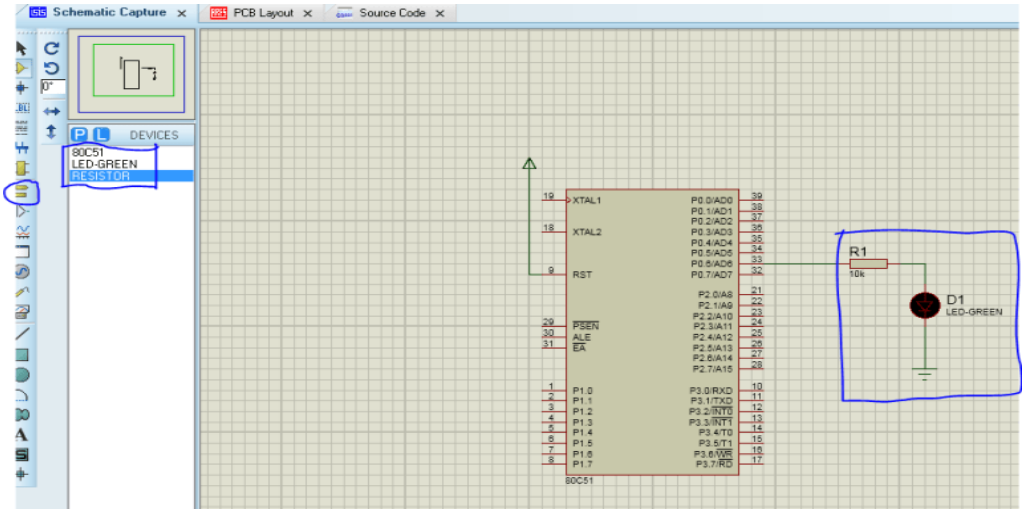




- **Step 8:** Add all required components

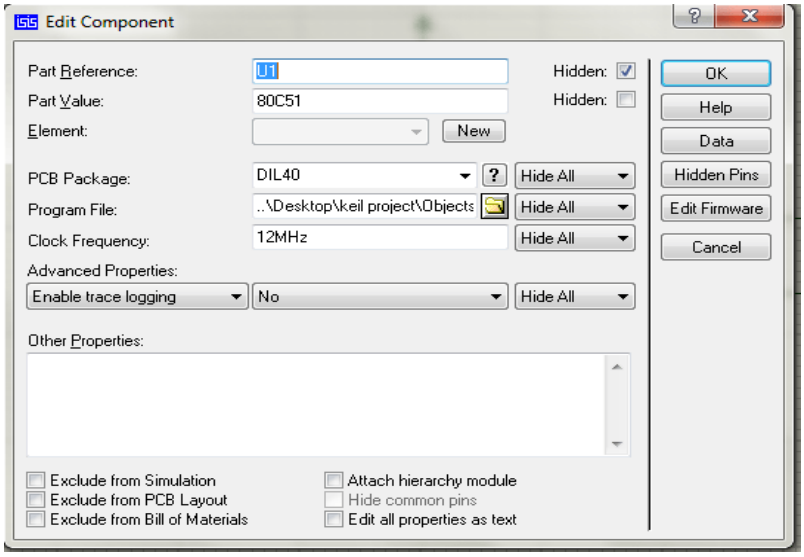


- **Step 9:** Next Place the components on Workplace
- Then wire up the circuit



3. Circuit Simulation

- **Step 10:** you are used microcontroller firmware then you burn the hex file()
- Double click on microcontroller and insert hex file.



- **Step 11:** Then click on play button
- Otherwise you directly click on play button on the bottom of left to start simulation

EXPERIMENT NO. 1

ADDITION, SUBTRACTION, MULTIPLICATION AND DIVISON OF TWO 8BIT NUMBERS

AIM: To add, subtract, multiply and division of two 8bit numbers by using 8051 microcontroller.

APPARATUS:
1. PC
2. KEIL SOFTWARE

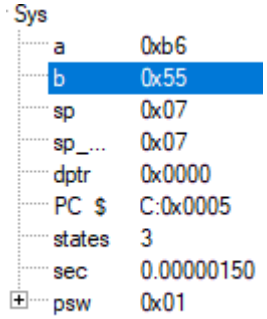
PROGRAM CODE;

ADDITION	SUBTRACTION	MULTIPLICATION	DIVISON
ORG 0000	ORG 0000	ORG 0000	ORG 0000
MOV A,#24H	MOV A,#44H	MOV A,#22H	MOV A,#22H
MOV B,#42H	MOV B,#37H	MOV B,#11H	MOV B,#11H
ADD A,B	CLR C	MUL AB	DIV AB
	SUBB A,B		
END	END	END	END

EXAMPLE

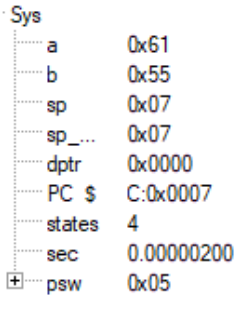
Let A = B6H B = 55H

Before Operation execution Register view in KEIL

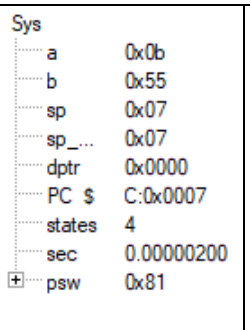


After Operation execution Register view in KEIL

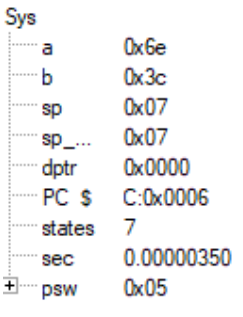
ADD



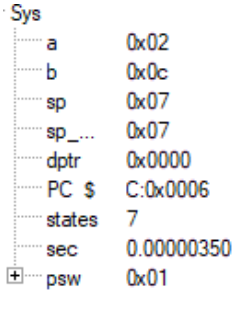
SUB



MUL



DIV



RESULT

INPUT DATA	***INPUT DATA***	***INPUT DATA***	***INPUT DATA***
A	A	A	A
B	B	B	B
OUTPUT DATA	***OUTPUT DATA***	***OUTPUT DATA***	***OUTPUT DATA***
A	A	A	A
		B	B

DRAW THE FLOW CHART:

NOTES:

EXPERIMENT NO. 2

ADDITION AND SUBTRACTION OF TWO 16BIT NUMBERS

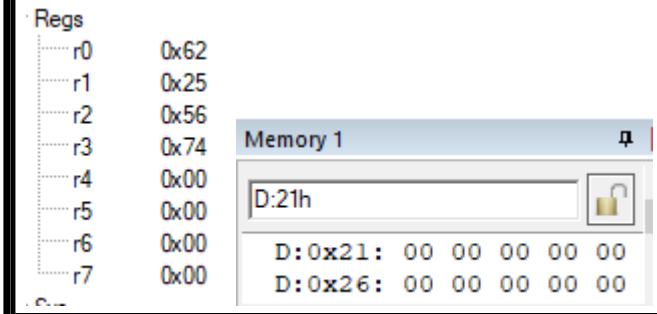
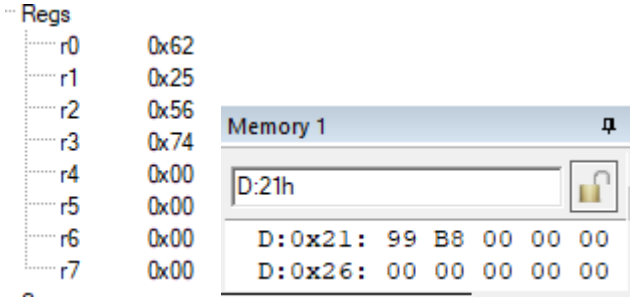
AIM: To perform the addition and subtraction of two 16-bit numbers.

APPARATUS: 1. PC
2. KEIL SOFTWARE

PROGRAM CODE FOR ADDITION;

ADDITION	COMMENTS
ORG 0000	
MOV R0,#34H	//lower byte of No.1
MOV R1,#12H	//higher byte of No.1
MOV R2,#0DCH	//lower byte of No.2
MOV R3,#0FEH	//higher byte of No.2
CLR C	
MOV A,R0	
ADD A,R2	Addition of lower bytes of two numbers
MOV 22H,A	store lower byte of result in 22h of iRAM
MOV A,R1	
ADDC A,R3	Addition of higher bytes of two numbers with carry
MOV 21H,A	store higher byte of result in 22h of iRAM
END	

EXAMPLE: let 1st 16 bit number : 2562H
2nd 16 bit number : 7456H

Before execution of addition Registers and memory view in KEIL	After execution of addition Registers and memory status
	

RESULT:

Input data: **Output data:**

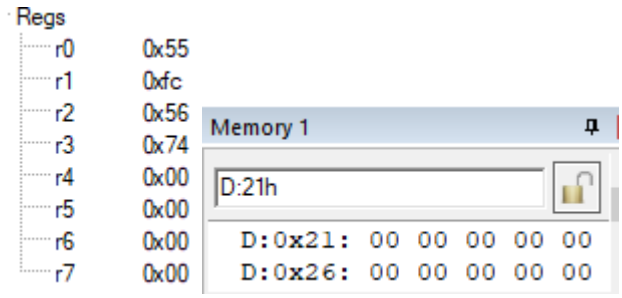
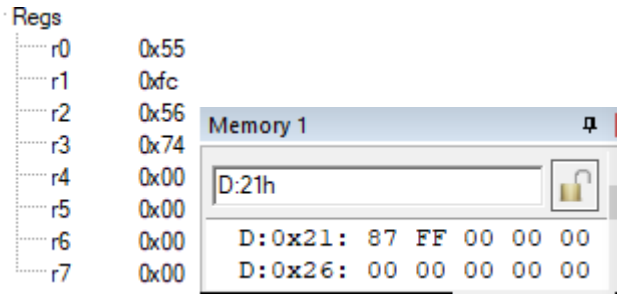
No.1: **1234 H** D:21H D:22H

No.2: **FEDC**H

PROGRAM CODE FOR SUBTRACTION;

SUBTRACTION	COMMENTS
ORG 0000	
MOV R0,#0DH	//lower byte of No.1
MOV R1,#0FH	//higher byte of No.1
MOV R2,#34H	//lower byte of No.2
MOV R3,#12H	//higher byte of No.2
CLR C	
MOV A,R0	
SUBB A,R2	subtraction of lower bytes of two numbers
MOV 22H,A	store lower byte of result in 22h of iRAM
MOV A,R1	
SUBB A,R3	subtraction of higher bytes of two numbers with barrow
MOV 21H,A	store higher byte of result in 22h of iRAM
END	

EXAMPLE: let 1st 16 bit number : FC55H
2nd 16 bit number : 7456H

Before execution of addition Registers and memory view in KEIL	After execution of addition Registers and memory status
	

RESULT:

Input data:

No.1: **0F0DH**
No.2: **1234 H**

Output data:

D:21H D:22H

DRAW THE FLOW CHART:

EXPERIMENT NO. 3

LCM OF GIVEN TWO DECIMAL NUMBERS

AIM: To find LCM of two decimal numbers which are stored in internal RAM locations 30h&31h,and store the LCM 32h

APPARATUS: 1. PC
2. KEIL SOFTWARE

PROGRAM CODE;

LABEL	INSTRUCTIONS	COMMENTS
	ORG 0000H	
	MOV R0,30H	Copy 1st number to R0
	MOV R1,31H	Copy 2nd number to R1
	MOV R2,#01H	Load R2 with 01 for multiplication with smallest number
	MOV A,R0	Copy the 1st number to accumulator
	MOV B,R1	Copy 2nd number to regB
	CJNE A,B,**LOOP1**	Check for smallest among the two numbers if not equal goto LOOP1
	MOV 32H,A	If both are equal store the LCM in 32H
	SJMP **LAST**	
LOOP1:	JNC **LOOP2**	Check for smallest number among two numbers and keep smallest number in R0
LOOP3:	MOV A,R0	Move smallest number to Accumulator
	MOV B,R2	
	MUL AB	
	MOV R3,A	Store the multiplication result in R3
	MOV B,R1	Move the biggest number from R1 to regB
	DIV AB	Divide multiplication result with biggest number
	MOV R4,B	Move reminder to R4
	INC R2	Increment R2 for next multiplication

	CJNE R4,#00H,**LOOP3**	Check the reminder for 0
	MOV 32H,R3	Store LCM in 32H
	SJMP **LAST**	
LOOP2:	MOV R0,B	Exchange R0 and R1 to keep smallest among two numbers in R0
	MOV R1,A	Move biggest number in R1
	SJMP **LOOP3**	
LAST:	SJMP **LAST**	
	END	

Logic: let the two numbers : **07d, 09d**

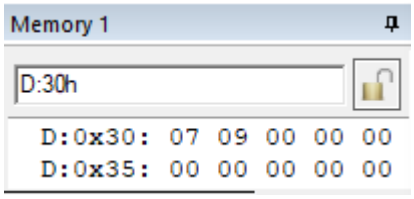
Step1: Find the smallest among the two**: 07d**

Step2: 07x01=07 Step3: 07/09 result will be 07
07x02=14 14/09 result will be 05
07x03=21 21/09 result will be 03

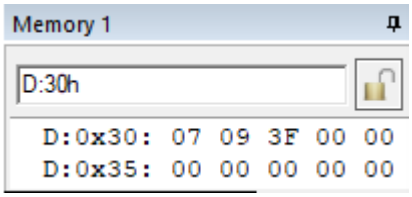
Above step repeated still we got result 00 at that point the multiplication result will be the required LCM of the given two number.

For the above example LCM is **63d=3Fh**

Before execution memory view



After execution memory view



RESULT:

Input data:

D: 30H

D: 31H

Output data:

D:32H

DRAW THE FLOW CHART

EXPERIMENT NO. 4

MOVE BLOCK OF 10BYTES IN INTERNAL RAM

AIM: To Block move - 10bytes of data from 0x30-0x39 to 0x40-0x49 of internal RAM

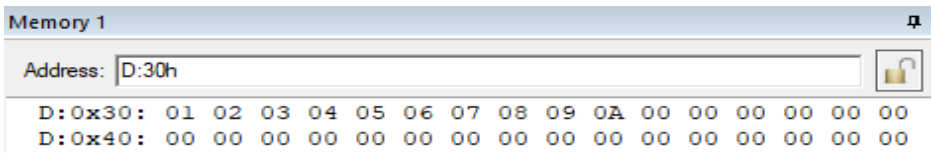
APPARATUS:
1. PC
2. KEIL SOFTWARE

PROGRAM CODE;

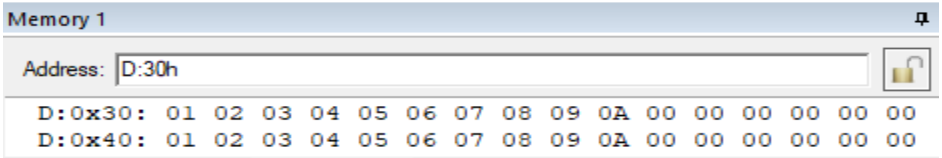
LABEL	INSTRUCIONS	COMMENTS
	ORG 0000H	
	MOV R0,#30H	R0 works as source address pointer
	MOV R1,#40H	R1 works as destination address pointer
	MOV R2,#10D	R2 works as counter
NEXT:	MOV A,@R0	Moving byte by byte from source to destination
	MOV @R1,A	
	INC R0	Increment source and destination address pointer
	INC R1	
	DJNZ R2,**NEXT**	Decrement counter
	END	

EXAMPLE:

Before execution memory view



After execution memory view



RESULT:

Input data:		**Output data:**	
D: 30H		D: 40H	
D: 31H		D: 41H	
D: 32H		D: 42H	
D: 33H		D: 43H	
D: 34H		D: 44H	
D: 35H		D: 45H	
D: 36H		D: 46H	
D: 37H		D: 47H	
D: 38H		D: 48H	
D: 39H		D: 49H	

DRAW THE FLOW CHART

NOTES

EXPERIMENT NO. 5

EXCHANGE BLOCK OF 10BYTES IN INTERNAL RAM

AIM: To Block exchange - 10bytes of data between 0x30-0x39 to 0x40-0x49 of internal RAM

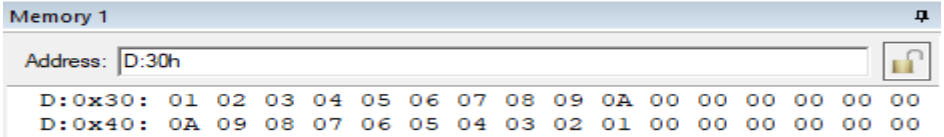
APPARATUS:
1. PC
2. KEIL SOFTWARE

PROGRAM CODE;

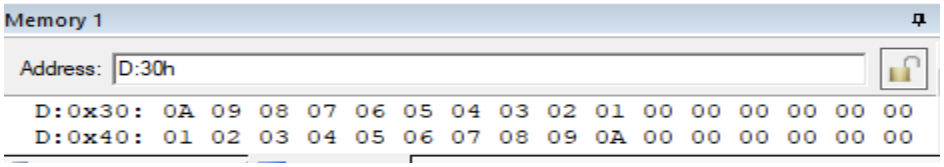
LABEL	INSTRUCIONS	COMMENTS
	ORG 0000H	
	MOV R0,#30H	R0 works as first source block address pointer
	MOV R1,#40H	R1 works as second source block address pointer
	MOV R2,#10D	R2 works as counter
NEXT:	MOV A,@R0	Exchange bytes between locations through registers
	MOV B,@R1	
	MOV @R0,B	
	MOV @R1,A	
	INC R0	Increment both block address pointers
	INC R1	
	DJNZ R2,NEXT	Decrement counter
	END	

EXAMPLE:

Before execution memory view



After execution memory view



RESULT:

Input data:

D:30H		D:40 H	
D:31H		D:41H	
D:32H		D:42H	
D:33H		D:43H	
D:34H		D:44H	
D:35H		D: 45H	
D:36H		D:46H	
D:37H		D: 47H	
D:38H		D: 48H	
D:39H		D: 49H	

Output data:

D:30H		D:40H	
D:31H		D:41H	
D:32H		D:42H	
D:33H		D:43H	
D:34H		D:44H	
D:35H		D:45H	
D:36H		D:46H	
D:37H		D:47H	
D:38H		D:48H	
D:39H		D:49H	

DRAW THE FLOW CHART

NOTES:

EXPERIMENT NO. 6

FINDING SMALLEST/LARGEST NUMBER IN 10 BYTES OF DATA

AIM: To find Smallest/Largest number in 10bytes of data from 0X30-0X39 (R3 –store the smallest/largest number and R4 –store address of the smallest/largest number)

APPARATUS: 1. PC
2. KEIL SOFTWARE

PROGRAM CODE FOR SMALLEST NUMBER;

SMALLEST		
LABEL	**INSTRUCTIONS**	**COMMENTS**
	ORG 0000	
	MOV R0,#30H	R0 used as source address pointer
	MOV R2,#09H	R2 used as counter
	MOV A,@R0	Copy the data from iRAM to Accumulator
	MOV B,A	Register B is used to store **smallest** number
	MOV A,R0	
	MOV R4,A	R4 is used to store **smallest** number address
UP:	INC R0	Address pointer incremented to next address
	MOV A,@R0	Next number copied to Accumulator
	CJNE A,B,**DOWN**	Compare A,B values
	SJMP ***NEXT***	If both are equal next number will loaded to A
DOWN:	JNC ***NEXT***	Jump takes place when A>B
	MOV B,A	Move **smallest** number to Reg.B
	MOV A,R0	
	MOV R4,A	Copy of smallest number address copied to R4
NEXT:	DJNZ R2,***UP***	Counter decremented for each comparison
	MOV A,B	Move **smallest** number to A
	MOV R3,A	Store **smallest** number in R3
	END	

RESULT:

Input data:

D:30H	
D:31H	
D:32H	
D:33H	
D:34H	
D:35H	
D:36H	
D:37H	
D:38H	
D:39H	

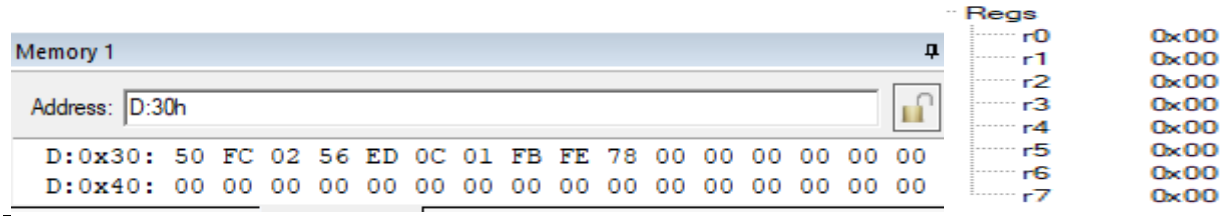
Output data:

R3 [ ]

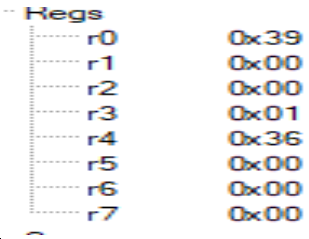
R4 [ ]

EXAMPLE:

Before execution memory and register view



After execution register view



DRAW THE FLOW CHART

PROGRAM CODE FOR SMALLEST NUMBER;

LARGEST		
LABEL	**INSTRUCTIONS**	**COMMENTS**
	ORG 0000	
	MOV R0,#30H	R0 used as source address pointer
	MOV R2,#09H	R2used as counter
	MOV A,@R0	Copy the data from iRAM to Accumulator
	MOV B,A	Register B is used to store **largest** number
	MOVA,R0	R4 is used to store **largest** number address
	MOV R4,A	
UP:	INC R0	Address pointer incremented to next address
	MOV A,@R0	Next number copied to Accumulator
	CJNE A,B,***DOWN***	Compare A,B values
	SJMP ***NEXT***	If both are equal next number will loaded to A
DOWN:	JC ***NEXT***	Jump takes place when A<B
	MOV B,A	Move **largest** number to Reg.B
	MOV A,R0	Copy of largest number address copied to R4
	MOV R4,A	
NEXT:	DJNZ R2,***UP***	Counter decremented for each comparison
	MOV A,B	Move **largest** number to A
	MOV R3,A	Store **largest** number in R3
	END	

RESULT:

Input data:

D:30H	
D:31H	
D:32H	
D:33H	
D:34H	
D:35H	
D:36H	
D:37H	
D:38H	
D:39H	

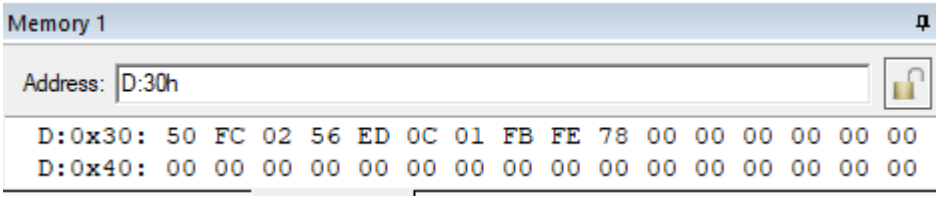
Output data:

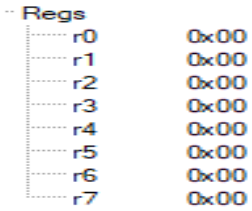
R3 [ ]

R4 [ ]

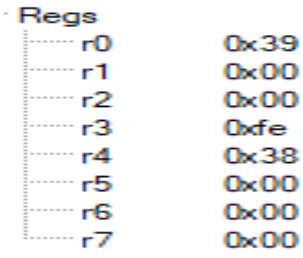
EXAMPLE:

Before execution memory and register view





After execution register view



DRAW THE FLOW CHART

EXPERIMENT NO. 7

2'S COMPLEMENT OF A NUMBER

AIM: To Find 2's complement of a number which is stored in 30H using (CPL) instruction, and store result in 31H

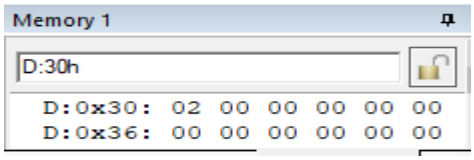
APPARATUS:
1. PC
2. KEIL SOFTWARE

PROGRAM CODE;

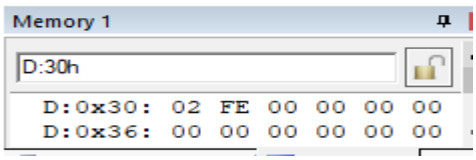
INSTRUCTIONS	COMMENTS
ORG 0000H	
MOV R0,#30H	r0 used as address pointer
MOV A,@R0	
CPL A	1's compliment of given data
INC A	adding one to data
INC R0	
MOV @R0,A	store the result at 31h
END	

EXAMPLE:

Before execution memory view



After execution memory view



RESULT:

Input data: **Output data:**

D:30H ☐ **D:**31H ☐

DRAW THE FLOW CHART

EXPERIMENT NO. 8

PACKED BCD TO UNPACKED BCD

AIM: To Convert Packed to Unpacked BCD of a number stored at 40H and store the result in 41H& 42H Using ANL (bit Masking) Instruction

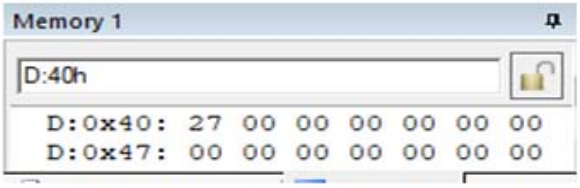
APPARATUS: 1. PC
2. KEIL SOFTWARE

PROGRAM CODE;

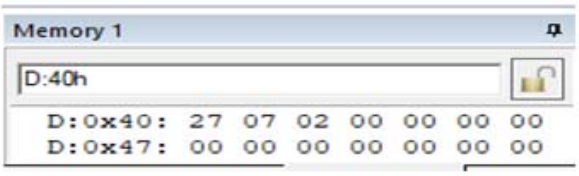
INSTRUCTIONS	COMMENTS
ORG 0000H	
MOV R0,#40H	R0 used as address pointer
MOV A,@R0	
MOV R1,A	copy data to R1
ANL A,#0FH	masking higher nibble
INC R0	
MOV @R0,A	store the lower nibble at 31H
MOV A,R1	
ANL A,#0F0H	masking lower nibble
SWAP A	
INC R0	
MOV @R0,A	Store the higher nibble at 32H
END	

EXAMPLE:

Before execution memory view



After execution memory view



RESULT:

Input data:

D:40H

Output data:

D:41H

D:42H

DRAW THE FLOW CHART

EXPERIMENT NO. 9

UNPACKED BCD TO ASCII CONVERSION

AIM: To convert Unpacked BCD number, which is stored in 40H to ASCII Using ORL instruction and store it in 41H

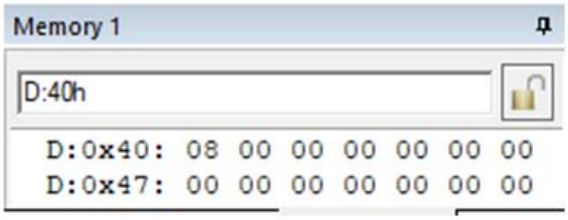
APPARATUS:
1. PC
2. KEIL SOFTWARE

PROGRAM CODE;

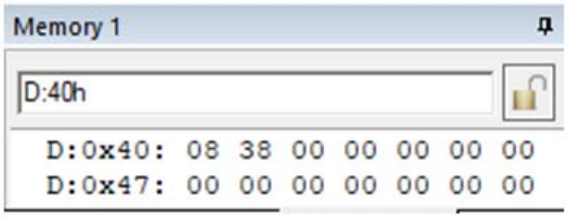
INSTRUCTIONS	COMMENTS
ORG 0000H	
MOV R0,#40H	R0 Used As Address Pointer
MOV A,@R0	
ORL A,#30H	
INC R0	
MOV @R0,A	Store The Result At 41H
END	

EXAMPLE:

Before execution memory view



After execution memory view



RESULT:

Inputdata: **Outputdata:**

D:40H ☐ D:41H ☐

EXPERIMENT NO. 10

PRODUCING REQUIRED TIME DELAY

AIM: to produce required time delay

a) By using instructions only

b) By Using Timers

APPARATUS: 1. PC

2. KEIL SOFTWARE

a)Producing 1ms time delay by using instructions only

In an 8051 microcontroller, it requires 12 cycles of the processor clock for executing a single instruction cycle. For an 8051 microcontroller clocked by a 12MHz crystal, the time taken for executing one instruction cycle is 1µS and it is according to the equation, *Time for 1 instruction cycle= 12 /12MHz = 1µS*. The instruction DJNZ Rn,*LABEL* is a two cycle instruction and it will take 2µS to execute. So repeating this instruction 500 times will generate a delay of 500 x 2µS = 1ms

PROGRAM CODE:

LABEL	INSTRUCTIONS	COMMENTS
	ORG 0000H	
	MOV R1,#250D	R1 used as counter for 250counts
	MOV R2,#250D	R2 used as counter for 250 counts
LABEL1:	DJNZ R1,LABEL1	loop runs for 250 times
LABEL2:	DJNZ R2,LABEL2	loop runs for 250 times
	END	

b)Producing 1ms time delay by using timers only

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

1. Assume the processor is clocked by a 12MHz crystal.
2. Timer to increment once it needs one instruction cycle i.e. 12 clock pulses
3. That means, the time taken for the timer to make one increment = 12/12MHz = 1uS
4. For a time delay of “X” uS the timer has to make “X” increments.
5. 2^16 = 65536 is the maximum number of counts possible for a 16 bit timer.

6. Let TH be the value value that has to be loaded to TH register and TL be the value that has to be loaded to TL register of particular T0 orT1
7. Then, THTL = Hexadecimal equivalent of (65536-X) where (65536-X) is considered in decimal.

Required delay be 1000uS (i.e.; 1mS).

- That means X = 1000
- 65536 – X = 65536 – 1000 = 64536.
- 64536 is considered in decimal and converting it t0 hexadecimal gives FC18
- That means THTL = FC18

LABEL	INSTRUCTIONS	COMMENTS
	ORG 0000H	
	MOV TMOD,#01H	Set Timer0 to Mode1
	MOV TH0,#FCH	
	MOV TL0,#18H	
	SETB TR0	Start Timer0
HERE	JNB TF0,HERE	Check the status of TF0 for timer to over flow
	CLR TR0	stop the timer
	CLR TF0	Clear the timer flag
	END	

EXPERIMENT NO. 11(A)

BLINKING AN LED AT SPECIFIC RATE

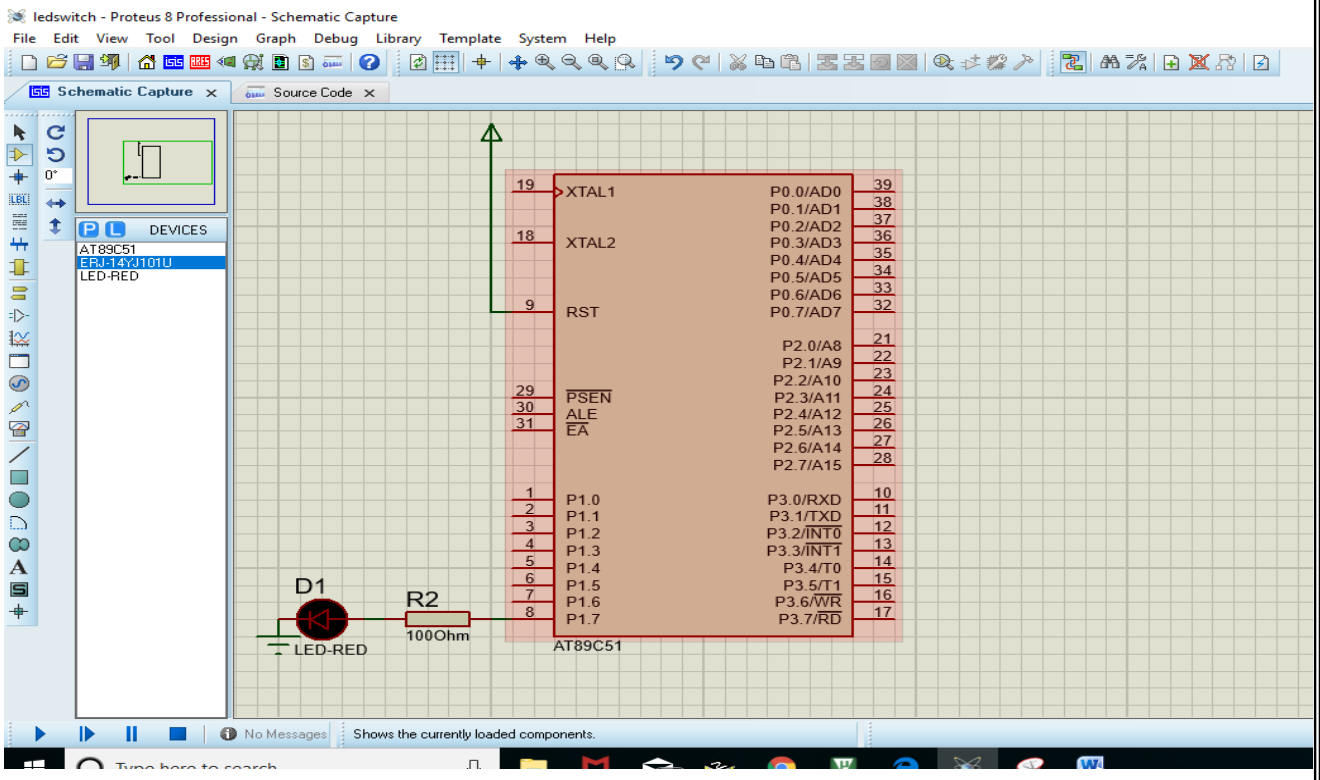
AIM: To make an LED connected to pin P1.7 to blink at a specific rate

APPARATUS:

1. PC
2. KEIL SOFTW
3. PROTEUS SOFTWARE

Components in proteus

PROTEUS CIRCUIT SIMULATION



DRAW INTERFACE DIAGRAM

PROGRAM CODE:

LABEL	INSTRUCTIONS	COMMENTS
	ORG 0000H	
AGAIN:	SETB P1.7	LED ON
	ACALL **DELAY**	Delay of 0.5sec
	CLR P1.7	LED OFF
	ACALL **DELAY**	Delay of 0.5sec
	SJMP **AGAIN**	
DELAY:	MOV R0,#04H	
UP2:	MOV R1,#250D	
UP1:	MOV R2,#250D	
	MOV R3,#250D	Delay for 1ms / Delay for 250ms / delay for 0.5sec
LABEL1:	DJNZ R2,**LABEL1**	
LABEL2:	DJNZ R3,**LABEL2**	
	DJNZ R1,**UP1**	
	DJNZ R0,**UP2**	
	RET	

RESULT:

DRAW FLOW CHART

EXPERIMENT NO. 11(B)

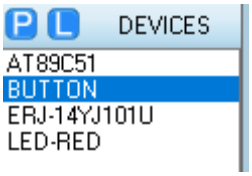
INTERFACE SWITCHES AND LEDS TO 8051

AIM: To make an LED connected to port pin P1.5, light up for specific time on pressing a switch connected to port pin P2.3

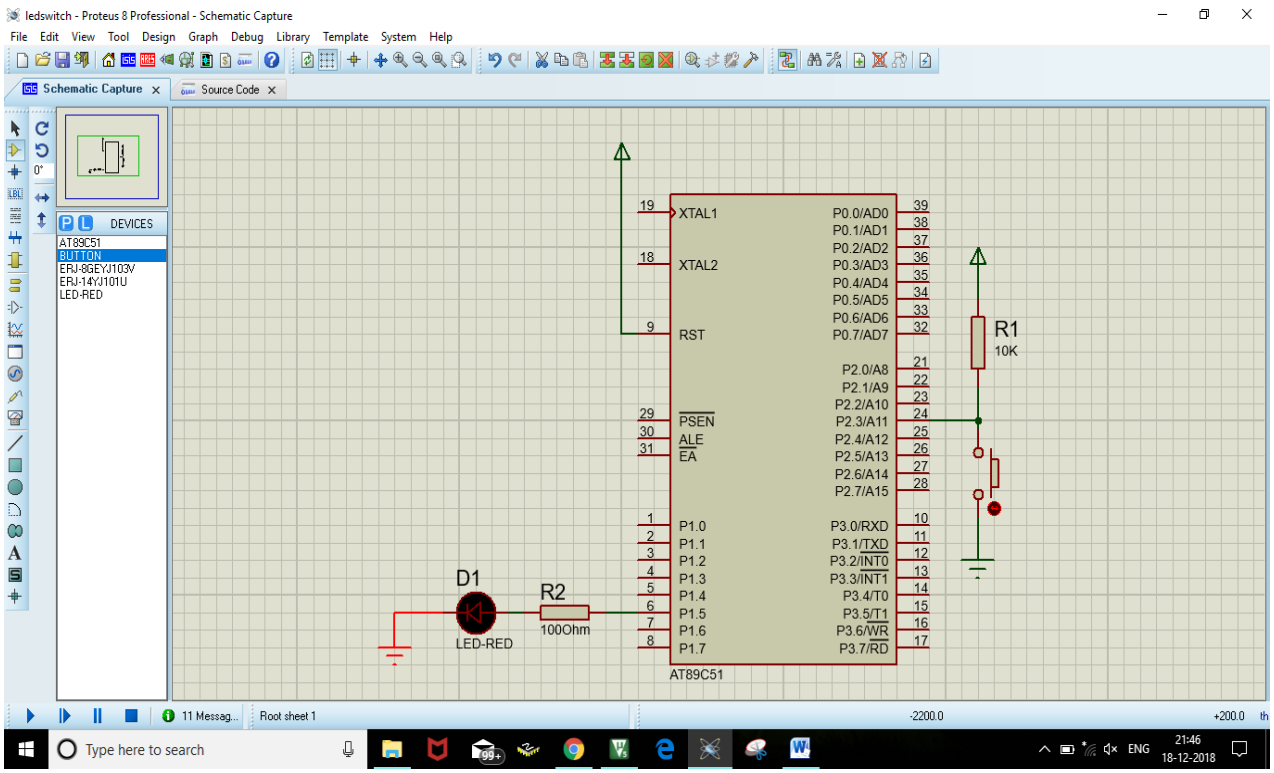
APPARATUS:

1. PC
2. KEIL SOFTWARE
3. PROTEUS SOFTWARE

Components in proteus



PROTEUS CIRCUIT SIMULATION



DRAW INTERFACE DIAGRAM

PROGRAM CODE:

LABEL	INSTRUCTIONS	COMMENTS
	ORG 0000H	
	SETB P2.3	Set P2.3 as input pin
	CLR P1.5	LED off
AGAIN:	JB P2.3,**AGAIN**	Check for switch status
	SETB P1.5	LED On
	ACALL **DELAY**	Delay of 1sec
	CLR P1.5	LED Off
	SJMP **AGAIN**	
DELAY:	MOV R0,#04H	
UP2:	MOV R1,#250D	
UP1:	MOV R2,#250D	
	MOV R3,#250D	Delay for 1ms / Delay for 250ms / delay for 1sec
LABEL1:	DJNZ R2,**LABEL1**	
LABEL2:	DJNZ R3,**LABEL2**	
	DJNZ R1,**UP1**	
	DJNZ R0,**UP2**	
	RET	

RESULT:

DRAW FLOW CHART

EXPERIMENT NO.12

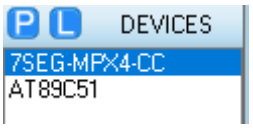
INTERFACE MULTIPLEXED 4-DIGIT 7SEGMENT LED DISPLAY

AIM: Interface multiplexed 4-digit 7SEGMENT LED DISPLAY to 8051 and display 2019

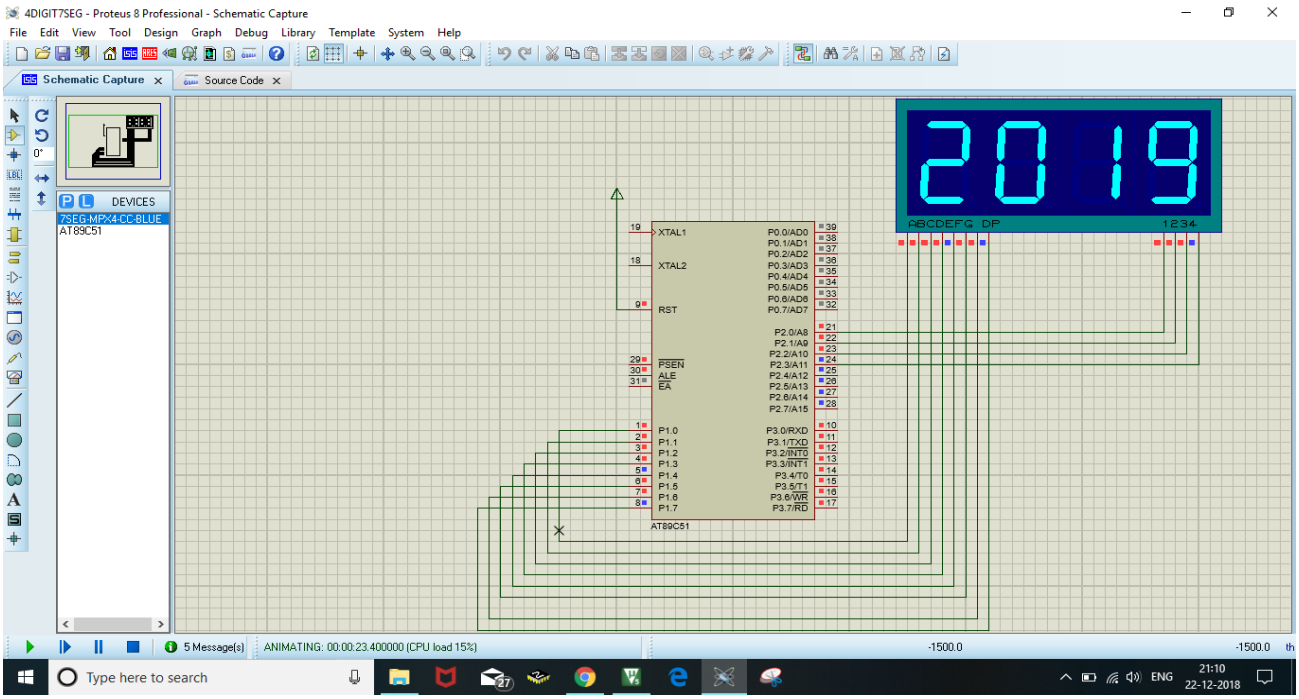
APPARATUS:

1. PC
2. KEIL SOFTWARE
3. PROTEUS SOFTWARE

Components in proteus



PROTEUS CIRCUIT SIMULATION



DRAW INTERFACE DIAGRAM

PROGRAM CODE:

LABEL	INSTRUCTIONS	COMMENTS
	MOV DPTR,#0400H	DPTR is initialized to point look up table
START:	MOV A,#02H	Display
	MOVC A,@A+DPTR	
	MOV P2,#0FEH	Activate first display
	MOV P1,A	Send corresponding hex code of 2 to port1
	ACALL **DELAY**	
	MOV A,#00H	Display
	MOVC A,@A+DPTR	
	MOV P2,#0FDH	Activate second display
	MOV P1,A	Send corresponding hex code of 0 to port1
	ACALL **DELAY**	
	MOV A,#01H	Display
	MOVC A,@A+DPTR	
	MOV P2,#0FBH	Activate third display
	MOV P1,A	Send corresponding hex code of 1 to port1
	ACALL **DELAY**	
	MOV A,#09H	Display
	MOVC A,@A+DPTR	
	MOV P2,#07H	Activate fourth display
	MOV P1,A	Send corresponding hex code of 9 to port1
	ACALL **DELAY**	
	SJMP **START**	
DELAY:	MOV R0,#10D	DEL 10ms time delay
UP1:	MOV R1,#0FFH	
	MOV R2,#0FFH	
LABEL1:	DJNZ R1,**LABEL1**	
LABEL2:	DJNZ R2,**LABEL2**	
	DJNZ R0,**UP1**	
	RET	

	ORG 0400H	
MY DATA:	DB 3FH,06H,5BH,4FH,66H, 6DH,7DH,07H,7FH,6fH	LOOK UP TABLE TO STORE 7 SEGMENT DISPLAY HEX CODE FROM 0 TO 9
	END	

RESULT:

EXPERIMENT NO. 13

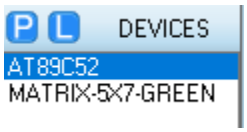
INTERFACE DOT MATRIX LED DISPLAY

AIM: To Interface a Single DOTMATRIX DISPLAY and display the given letter (A)

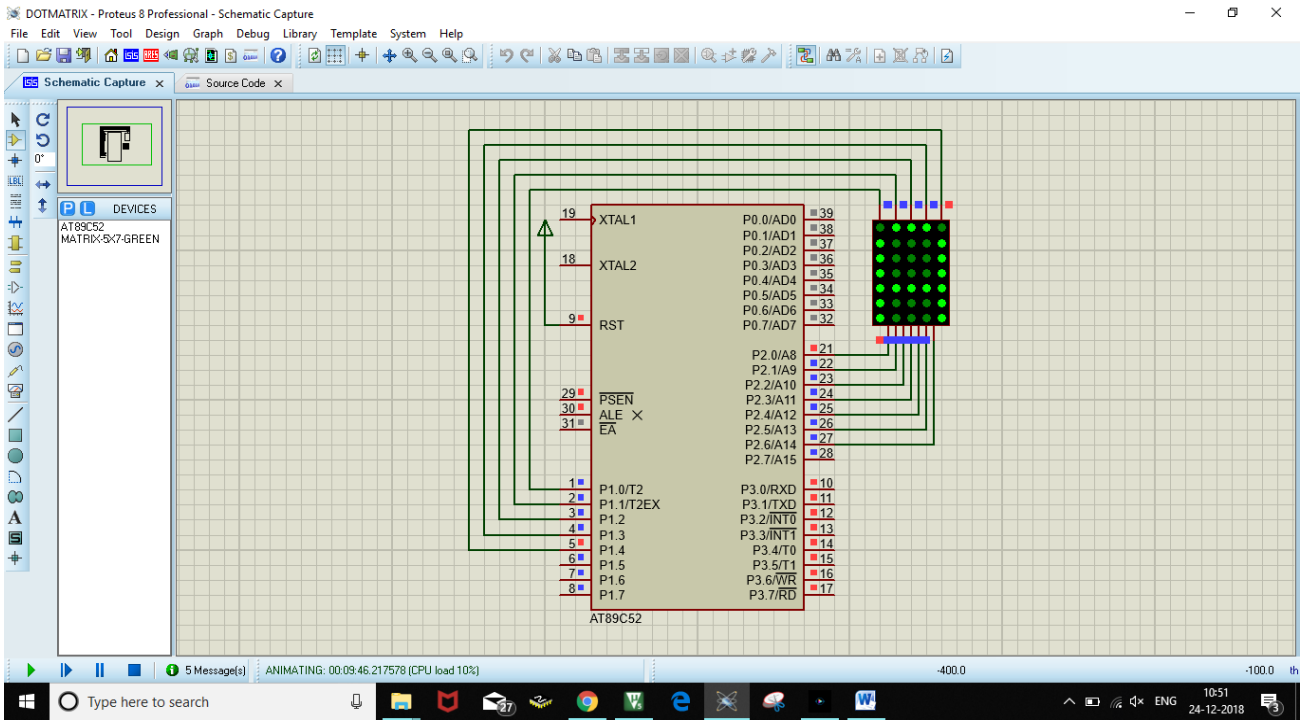
APPARATUS:

1. PC
2. KEIL SOFTWARE
3. PROTEUS SOFTWARE

Components in proteus



PROTEUS CIRCUIT SIMULATION



DRAW INTERFACE DIAGRAM

PROGRAM CODE:

LABEL	INSTRUCTIONS	COMMENTS
	ORG 0000H	
START:	MOV A,#11H	
	MOV P1,A	enable first column
	MOV P2,#01H	
	ACALL **DELAY**	
	RL A	
	MOV P1,A	enable second column
	MOV P2,#0EEH	
	ACALL **DELAY**	
	RL A	
	MOV P1,A	enable third column
	MOV P2,#0EEH	
	ACALL **DELAY**	
	RL A	
	MOV P1,A	enable fourth column
	MOV P2,#0EEH	
	ACALL **DELAY**	
	RL A	
	MOV P1,A	enable fifth column
	MOV P2,#01H	
	ACALL **DELAY**	
	SJMP **START**	
DELAY:	MOV R1,#0FFH	delay subroutine
	MOV R2,#0FFH	
LABEL1:	DJNZ R2,**LABEL1**	
LABEL2:	DJNZ R1,**LABEL2**	
	RET	
	END	

RESULT:

EXPERIMENT NO. 14

16X2 LCD INTERFACE AND DISPLAY

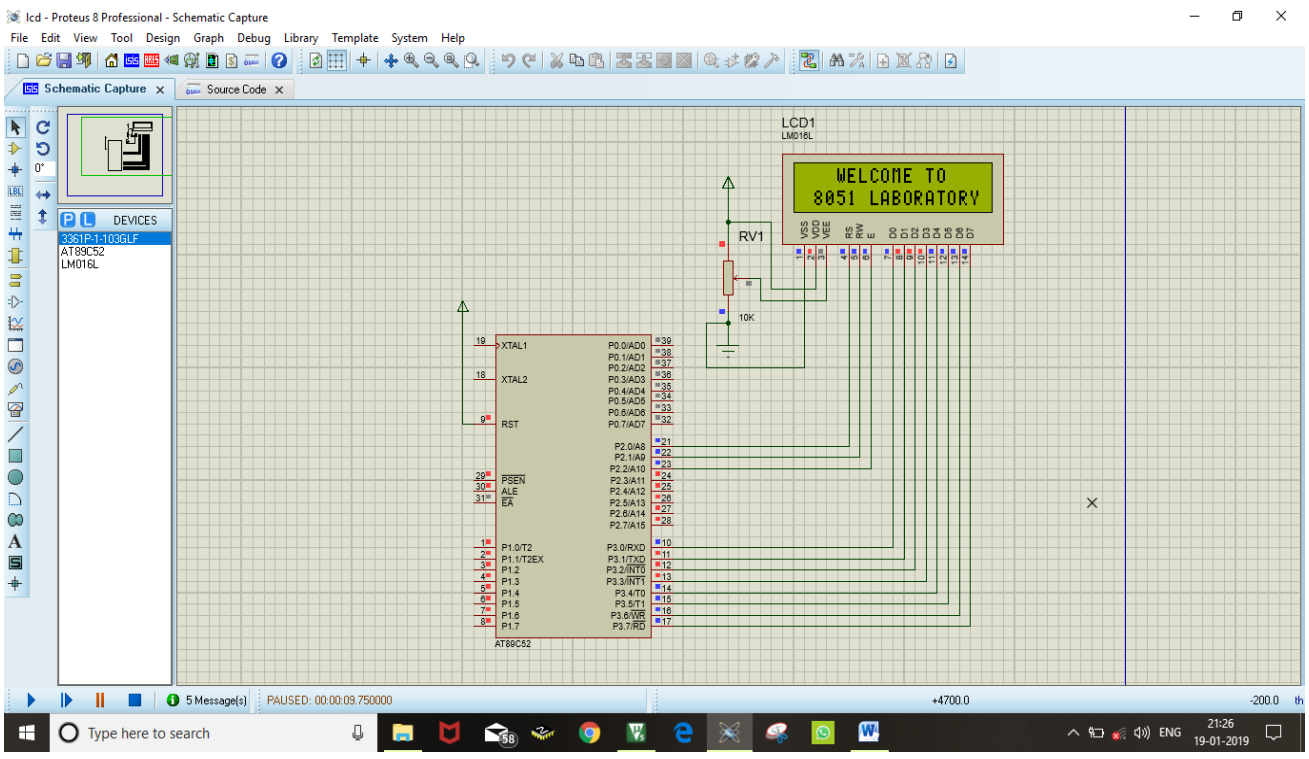
AIM: To Interface 16x2 LCD and display “WELCOME TO 8051 LABORATORY”

APPARATUS:

1. PC
2. KEIL SOFTWARE
3. PROTEUS SOFTWARE

Components in proteus

PROTEUS CIRCUIT SIMULATION



DRAW INTERFACE DIAGRAM

PROGRAM CODE:

LABEL	INSTRUCTIONS	COMMENTS
	ORG 0000H	
START:	ACALL **INITLCD**	Calling LCD initializing subroutine
	MOV A,#' '	
	ACALL **DATAWRT**	Calling LCD data write subroutine
	MOV A,#'**W**'	
	ACALL **DATAWRT**	
	MOV A,#'**E**'	
	ACALL **DATAWRT**	
	MOV A,#'**L**'	
	ACALL **DATAWRT**	
	MOV A,#'**C**'	
	ACALL **DATAWRT**	
	MOV A,#'**O**'	
	ACALL **DATAWRT**	
	MOV A,#'**M**'	
	ACALL **DATAWRT**	
	MOV A,#'**E**'	
	ACALL **DATAWRT**	
	MOV A,#' '	
	ACALL **DATAWRT**	
	MOV A,#'**T**'	
	ACALL **DATAWRT**	
	MOV A,#'**O**'	
	ACALL **DATAWRT**	
	MOV A,#0C0H	Command for to display second line
	ACALL **CMDWRT**	Calling command write subroutine
	MOV A,#' '	
	ACALL **DATAWRT**	
	MOV A,#'**8**'	

	ACALL **DATAWRT**	
	MOV A,#'**0**'	
	ACALL **DATAWRT**	
	MOV A,#'**5** '	
	ACALL **DATAWRT**	
	MOV A,#'**1**'	
	ACALL **DATAWRT**	
	MOV A,#'**L**'	
	ACALL **DATAWRT**	
	MOV A,#'**A**'	
	ACALL **DATAWRT**	
	MOV A,#'**B**'	
	ACALL **DATAWRT**	
	MOV A,#'**O**'	
	ACALL **DATAWRT**	
	MOV A,#'**R**'	
	ACALL **DATAWRT**	
	MOV A,#'**A**'	
	ACALL **DATAWRT**	
	MOV A,#'**T**'	
	ACALL **DATAWRT**	
	MOV A,#'**O**'	
	ACALL **DATAWRT**	
	MOV A,#'**R**'	
	ACALL **DATAWRT**	
	MOV A,#'**Y**'	
	ACALL **DATAWRT**	
	SJMP **START**	
DATAWRT:	MOV P3,A	Send display character to port3
	SETB P2.0	Selecting data register in LCD by send logic 1 to Rs pin

	CLR P2.1	Writing data to LCD by send logic 0 to RW pin
	SETB P2.2	Enable LCD by send logic 1 to 0 transaction to E pin
	ACALL **DELAY1**	
	CLR P2.2	
	ACALL **DELAY2**	
	RET	
CMDWRT:	MOV P3,A	Send command word to port3
	CLR P2.0	Selecting command register in lcd by send logic 1 to Rs pin
	CLR P2.1	Writing data to LCD by send logic 0 to RW pin
	SETB P2.2	Enable LCD by send logic 1 to 0 transaction to E pin
	ACALL **DELAY1**	
	CLR P2.2	
	ACALL **DELAY2**	
	RET	
INITLCD:	MOV A,#38H	2 lines and 5×7 matrix (8-bit mode)
	ACALL **CMDWRT**	Calling command write subroutine
	MOV A,#0EH	Display on, cursor blinking
	ACALL **CMDWRT**	
	MOV A,#01H	Clear display screen
	ACALL **CMDWRT**	
	MOV A,#06H	Increment cursor (shift cursor to right)
	ACALL **CMDWRT**	
	MOV A,#80H	Force cursor to beginning to 1st line
	ACALL **CMDWRT**	
	RET	
DELAY2:	MOV R3,#250D	Long delay
LOOP1:	MOV R4,#255D	
LOOP2:	DJNZ R4,**LOOP2**	

	DJNZ R3,**LOOP1**	
	RET	
DELAY1:	MOV R3,#10H	Short delay
LOOP3:	MOV R4,#255	
LOOP4:	DJNZ R4,**LOOP4**	
	DJNZ R3,**LOOP3**	
	RET	
	END	

RESULT:

EXPERIMENT NO. 15

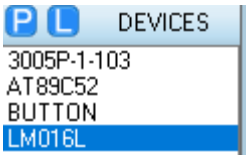
4X4 MATRIX KEYPAD INTERFACE

AIM: Interface a (4x4 matrix) Key Board to 8051 and display the number pressed on a LCD

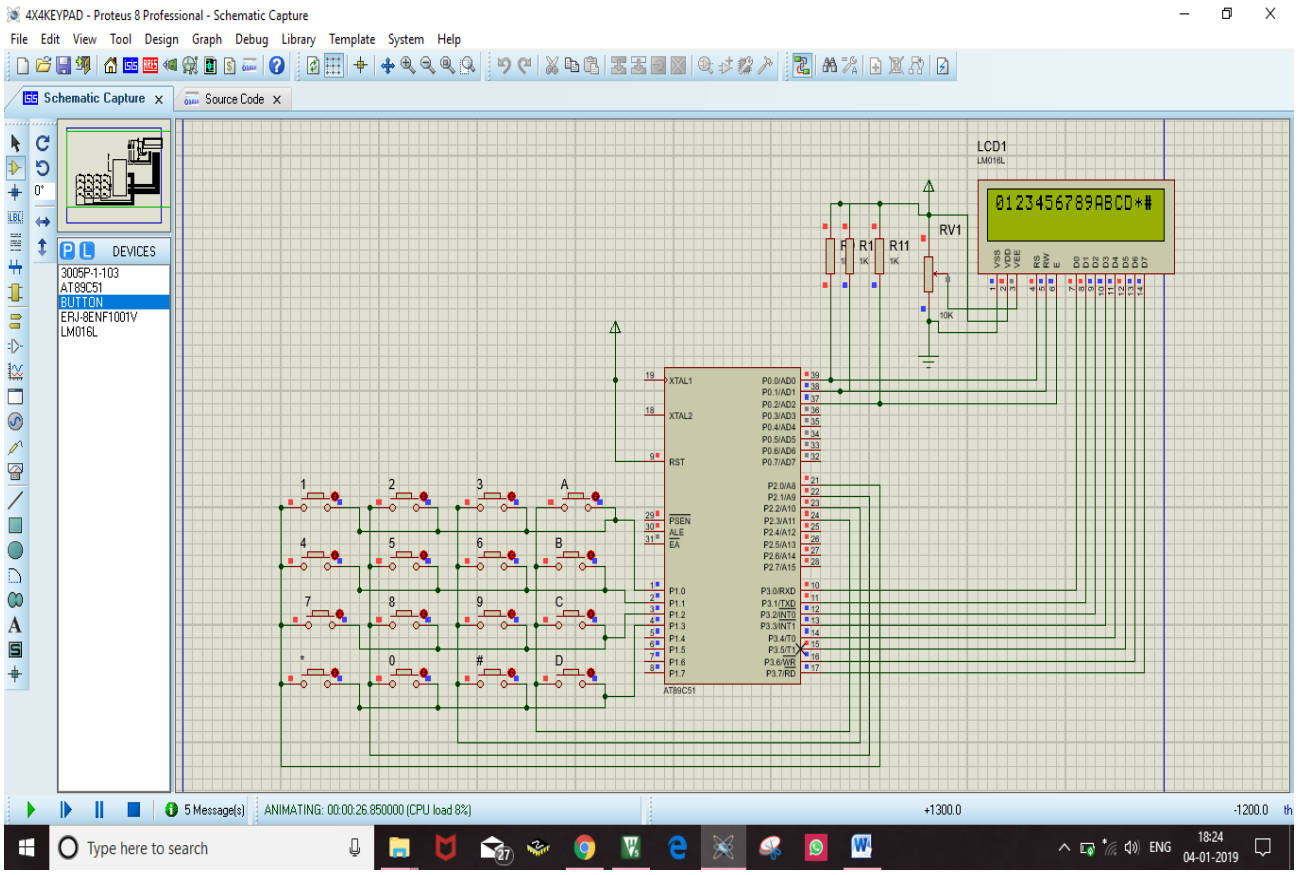
APPARATUS:

1. PC
2. KEIL SOFTWARE
3. PROTEUS SOFTWARE

Components in proteus



PROTEUS CIRCUIT SIMULATION



DRAW INTERFACE DIAGRAM

PROGRAM CODE:

LABEL	INSTRUCTIONS	COMMENTS
	ORG 0000H	
	ACALL **INITLCD**	Lcd Initialization
OPEN:	MOV P2,#0FFH	Make P2 as input port
	MOV P1,#00H	Send logic 0 for all Rows
	MOV A,P2	Read P2 status
	ANL A,#00001111B	Mask un used bits
	CJNE A,#00001111B,**OPEN**	Check for all buttons in open
K2:	ACALL **DELAY**	Wait for some time
	MOV A,P2	Again read P2 status
	ANL A,#00001111B	Mask un used bits
	CJNE A,#00001111B,**OVER**	Check for any button pressed
	SJMP **K2**	
OVER:	ACALL **DELAY**	Wait for some time
	MOV A,P2	Again read P2 status
	ANL A,#00001111B	Mask un used bits
	CJNE A,#00001111B,**OVER1**	Check for any button pressed
	SJMP K2	
OVER1:	MOV P1,#11111110B	Send logic 0 for first row(ROW0)
	MOV A,P2	Read P2 status
	ANL A,#00001111B	Mask unused bits
	CJNE A,#00001111B,**ROW0**	Check for Row0 button pressed
	MOV P1,#11111101B	Send logic 0 for second row(ROW1)
	MOV A,P2	Read P2 status

	ANL A,#00001111B	Mask unused bits
	CJNE A,#00001111B,**ROW1**	Check for Row1 button pressed
	MOV P1,#11111011B	Send logic 0 for third row(ROW2)
	MOV A,P2	Read P2 status
	ANL A,#00001111B	Mask unused bits
	CJNE A,#00001111B,**ROW2**	Check for Row2 button pressed
	MOV P1,#11110111B	Send logic 0 for third row(ROW3)
	MOV A,P2	Read P2 status
	ANL A,#00001111B	Mask unused bits
	CJNE A,#00001111B,**ROW3**	Check for Row3 button pressed
	LJMP OPEN	
ROW0:	MOV DPTR,#KCODE0	Copy address of row0 to DPTR for lookup table
	SJMP **FIND**	Call subroutine to Finding colomn in first row
ROW1:	MOV DPTR,#KCODE1	Copy address of row1 to DPTR for lookup table
	SJMP **FIND**	Finding colomn in seond row
ROW2:	MOV DPTR,#KCODE2	Copy address of row2 to DPTR for lookup table
	SJMP **FIND**	Call subroutine to Finding colomn in third row
ROW3:	MOV DPTR,#KCODE3	Copy address of row3 to DPTR for lookup table
	SJMP **FIND**	Call subroutine to Finding colomn in fourth row
FIND:	RRC A	
	JNC MATCH	Find the colomn in which button pressed
	INC DPTR	
	SJMP **FIND**	

MATCH:	CLR A	
	MOVC A,@A+DPTR	Copy the button to A from look up table
	ACALL **DISPLAY**	Send the pressed button to display
	LJMP OPEN	
DISPLAY:	MOV P3,A	Send pressed button number/character to P3 for display
	SETB P0.0	Select data register by RS=1
	CLR P0.1	Write data into lcd by RW=0
	SETB P0.2	Send logic 1 to 0 transaction to E for enable lcd
	ACALL **DELAY**	
	CLR P0.2	
	ACALL **DELAY**	
	RET	

CMDWRT:	MOV P3,A	Send command word to P3
	CLR P0.0	Select command register by RS=0
	CLR P0.1	Write data into lcd by RW=0
	SETB P0.2	Send logic 1 to 0 transaction to E for enable lcd
	ACALL **DELAY**	
	CLR P0.2	
	ACALL **DELAY**	
	RET	
INITLCD:	MOV A,#38H	2 lines and 5×7 matrix (8-bit mode)
	ACALL **CMDWRT**	Calling command write subroutine
	MOV A,#0EH	Display on, cursor blinking
	ACALL **CMDWRT**	

	MOV A,#01H	Clear display screen
	ACALL **CMDWRT**	
	MOV A,#06H	Increment cursor (shift cursor to right)
	ACALL **CMDWRT**	
	MOV A,#80H	Force cursor to beginning to 1st line
	ACALL **CMDWRT**	
	RET	
DELAY:	MOV R3,#10D	
LOOP3:	MOV R4,#255	
LOOP4:	DJNZ R4,**LOOP4**	
	DJNZ R3,**LOOP3**	
KCODE0:	DB '1','2','3','A'	LOOK UP TABLE
KCODE1:	DB '4','5','6','B'	
KCODE2:	DB '7','8','9','C'	
KCODE3:	DB '*','0','#','D'	
	END	

RESULT:

EXPERIMENT NO. 16

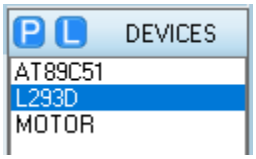
INTERFACE DC MOTOR

AIM: Interface a small DC motor control the direction of rotation of a small DC motor

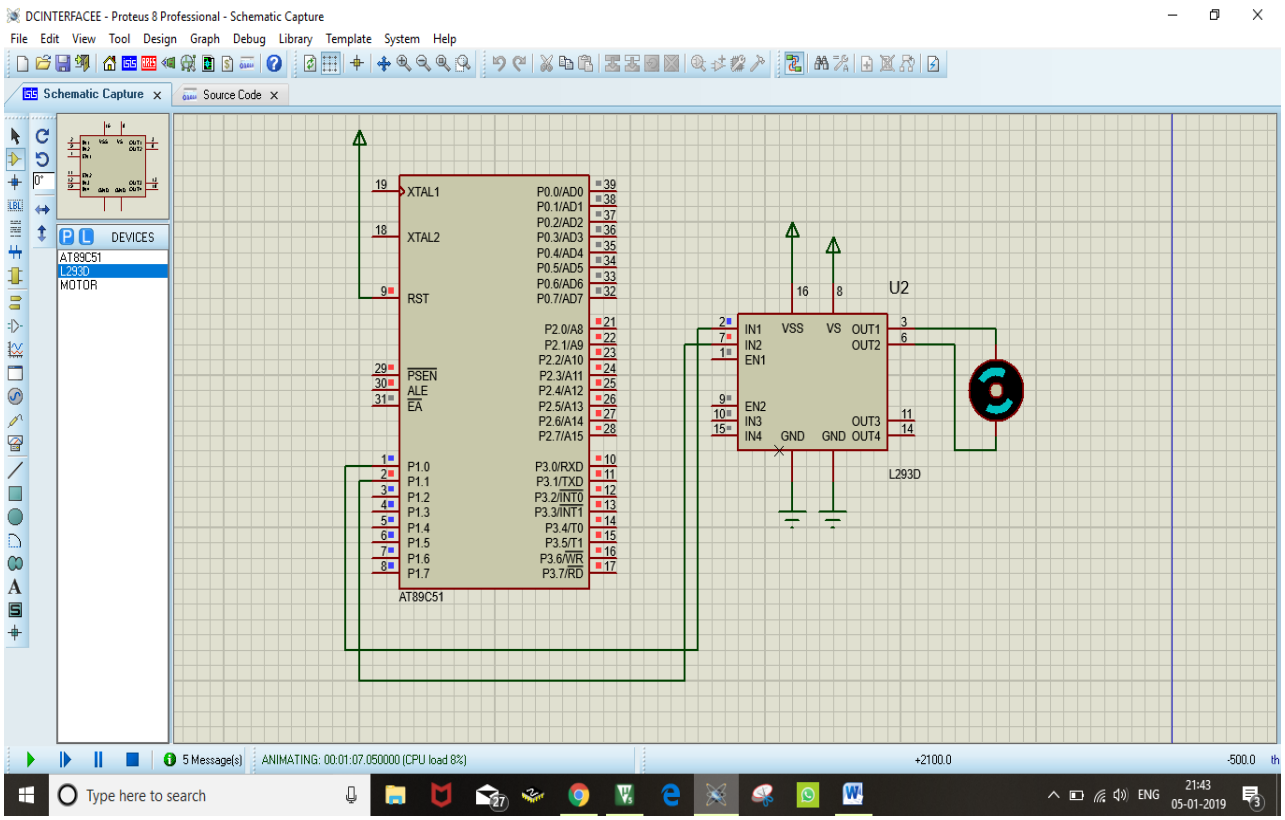
APPARATUS:

1. PC
2. KEIL SOFTWARE
3. PROTEUS SOFTWARE

Components in proteus



PROTEUS CIRCUIT SIMULATION



DRAW INTERFACE DIAGRAM

PROGRAM CODE:

LABEL	INSTRUCTIONS	COMMENTS
	ORG 0000H	
	MOV P1,#00000001B	Run motor in clock wise direction
	ACALL **DELAY**	Wait for some time
	MOV P1,#00000010B	Run motor in anticlockwise direction
	ACALL **DELAY**	
	END	
DELAY:	MOV R0,#0AH	
UP:	MOV R1,#0FFH	
LABEL2	MOV R2,#0FFH	
LABEL1:	DJNZ R2,**LABEL1**	
	DJNZ R1,**LABEL2**	
	DJNZ R0,UP	
	RET	

RESULT: