SYLLABUS

PRACTICAL EXERCISES: 30 PERIODS

Experiments using 8051

1. Programming Arithmetic and Logical Operations in 8051.
2. Generation of Square waveform using 8051.
3. Programming using on – Chip ports in 8051.
4. Programming using Serial Ports in 8051.
5. Design of a Digital Clock using Timers/Counters in 8051.

Experiments using ARM

1. Interfacing ADC and DAC
2. Blinking of LEDs and LCD
3. Interfacing keyboard and Stepper Motor.

Miniprojects for IoT

1. Garbage Segregator and Bin Level Indicator
2. Colour based Product Sorting
3. Image Processing based Fire Detection
4. Vehicle Number Plate Detection
5. Smart Lock System

COURSE OUTCOMES:

CO1: Explain the architecture and features of 8051.

CO2: Develop a model of an embedded system.

CO3: List the concepts of real time operating systems.

CO4: Learn the architecture and protocols of IoT.

CO5: Design an IoT based system for any application.

EXP NO:	**BASIC ARITHMETIC AND LOGICAL OPERATIONS USING 8051**
DATE:	**A. 8 BIT ADDITION**

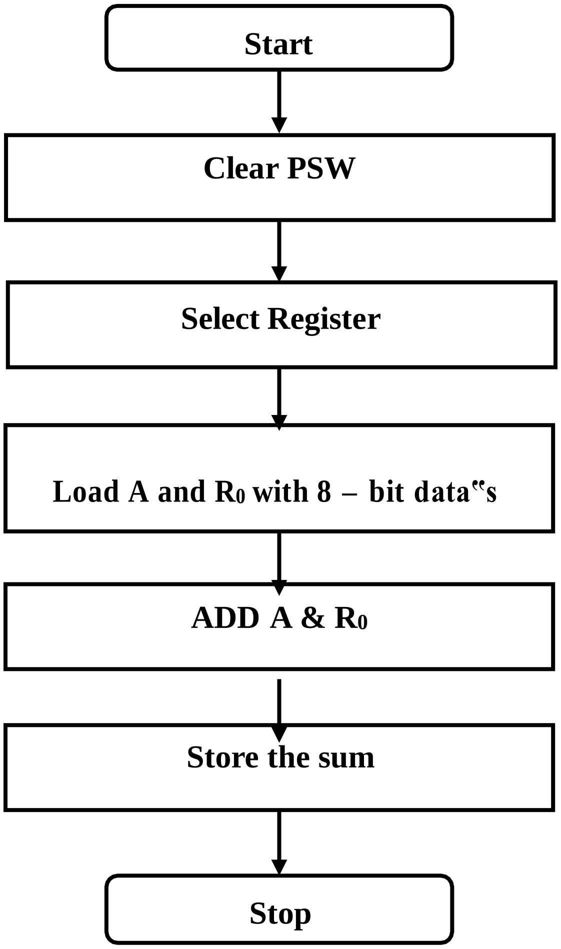
AIM:

To write a program to add two 8-bit numbers using 8051 microcontrollers.

ALGORITHM:

1. Clear Program Status Word.
2. Select Register bank by giving proper values to RS1 & RS0 of PSW.
3. Load accumulator A with any desired 8-bit data.
4. Load the register R 0 with the second 8- bit data.
5. Add these two 8-bit numbers.
6. Store the result.
7. Stop the program.

FLOW CHART:


Start
Clear PSW
Select Register
Load A and R0 with 8 – bit data‟s
ADD A & R0
Store the sum
Stop

PROGRAM:

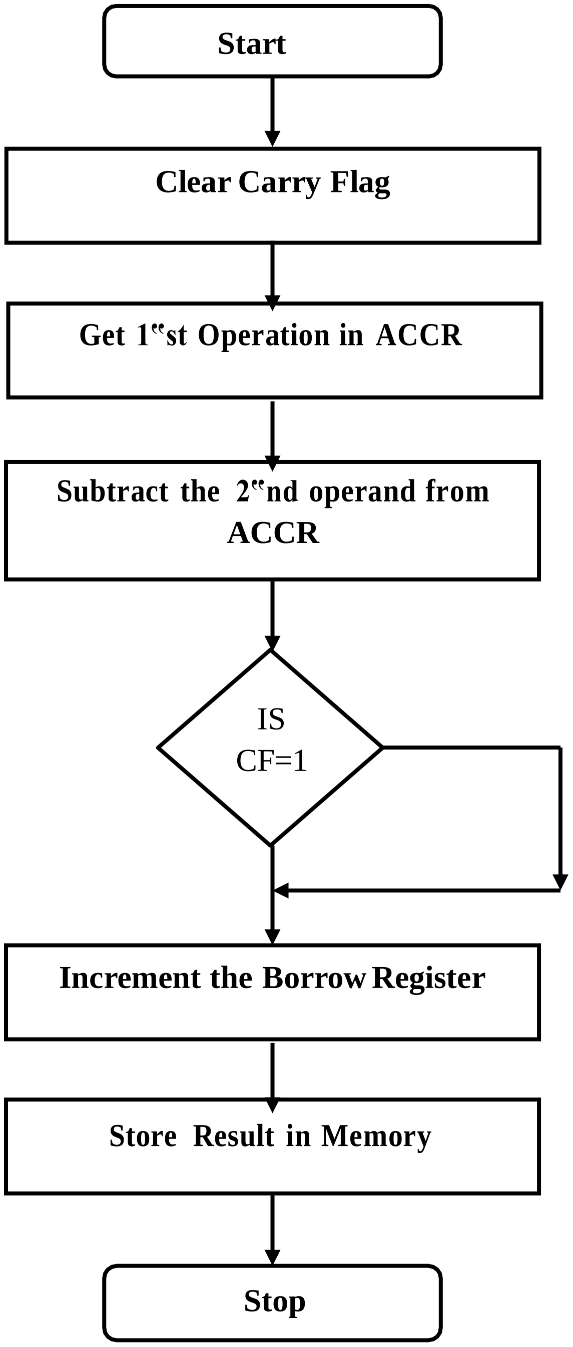
Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START:	8100	CLR	C	C3	Clear CY Flag
	8101	MOV	A,#0A	74 0A	Get the data1 in Accumulator
	8103	ADDC	A,#10	34 10	Add the data1 with data 2
	8105	MOV	DPTR,#8500	90 85 00	Initialize the memory location
	8108	MOVX	@DPTR,A	F0	Store the result in memory location
L1	8109	SJMP	L1	80 FE	Stop the program

Address	**Output**
8500	**1A(LSB)**
8501	**00(MSB)**

RESULT:

Thus the 8051 Assembly Language Program for addition of two 8 bit numbers was executed.

FLOW CHART:


Start
Clear Carry Flag
Get 1ˢᵗst Operation in ACCR
Subtract the 2ⁿᵈnd operand from ACCR
IS CF=1
Increment the Borrow Register
Store Result in Memory
Stop

B. 8 BIT SUBTRACTION

AIM:

To perform subtraction of two 8 bit data and store the result in memory.

ALGORITHM:

1. Clear the carry flag.
2. Initialize the register for borrow.
3. Get the first operand into the accumulator.
4. Subtract the second operand from the accumulator.
5. If a borrow results increment the carry register.
6. Store the result in memory.

PROGRAM:

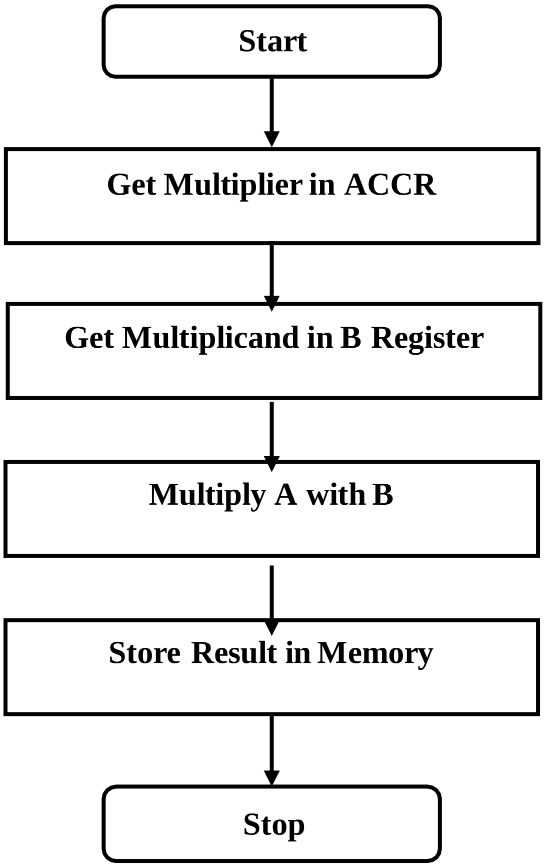
Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START:	8100	CLR	C	C3	Clear CY Flag
	8101	MOV	A,#0A	74 0A	Get the data1 in Accumulator
	8103	SUBB	A,#05	94 05	Subtract data2 from data1
	8105	MOV	DPTR,#4500	90 85 00	Initialize memory location
	8108	MOVX	@DPTR,A	F0	Store the difference in memory location
L1	8109	SJMP	L1	80 FE	Stop the program

Address	**Output**
8500	**05**

RESULT:

Thus the 8051 Assembly Language Program for subtraction of two 8 bit numbers was executed.

FLOW CHART:


Start
Get Multiplier in ACCR
Get Multiplicand in B Register
Multiply A with B
Store Result in Memory
Stop

C. 8 BIT MULTIPLICATION

AIM:

To perform multiplication of two 8 bit data and store the result in memory.

ALGORITHM:

1. Get the multiplier in the accumulator.
2. Get the multiplicand in the B register.
3. Multiply A with B.

Store the product in memory

PROGRAM:

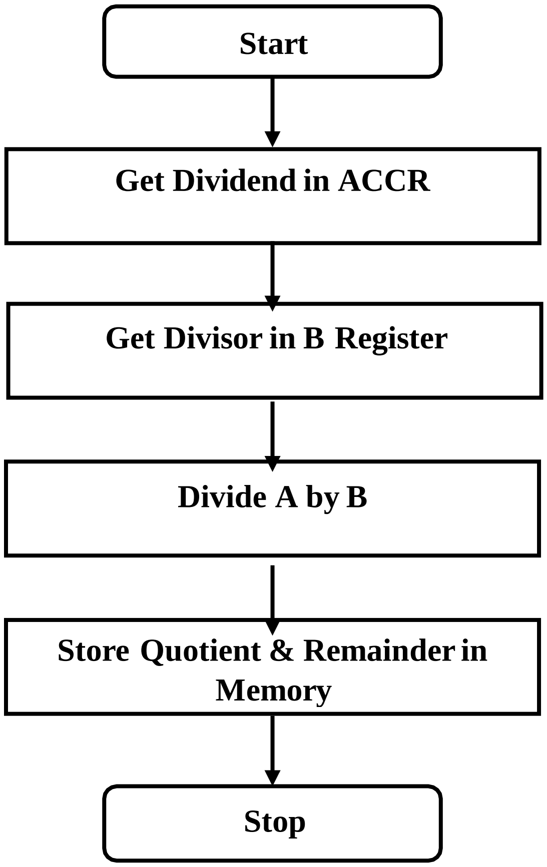
Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START:	8100	MOV	A,#05	74 05	Store data1 in accumulator
	8102	MOV	B,#03	75 F0 03	Store data2 in B register
	8105	MUL	AB	A4	Multiply both
	8106	MOV	DPTR,#4500	90 45 00	Initialize memory location
	8109	MOVX	@DPTR,A	F0	Store lower order result
	810A	INC	DPTR	A3	Go to next memory location
	810B	MOV	A,B	E5 F0	Store higher order result
	810D	MOVX	@DPTR,A	F0	
L1	810E	SJMP	L1	80 FE	Stop the program

Address	**Output**
8500	**0F(LSB)**
8501	**00(MSB)**

RESULT:

Thus the 8051Assembly Language Program for multiplication of two 8 bit numbers was executed.

FLOW CHART:


Start
Get Dividend in ACCR
Get Divisor in B Register
Divide A by B
Store Quotient & Remainder in Memory
Stop

D. 8 BIT DIVISION

AIM:

To perform division of two 8 bit data and store the result in memory.

ALGORITHM:

1. Get the Dividend in the accumulator.
2. Get the Divisor in the B register.
3. Divide A by B.
4. Store the Quotient and Remainder in memory

PROGRAM:

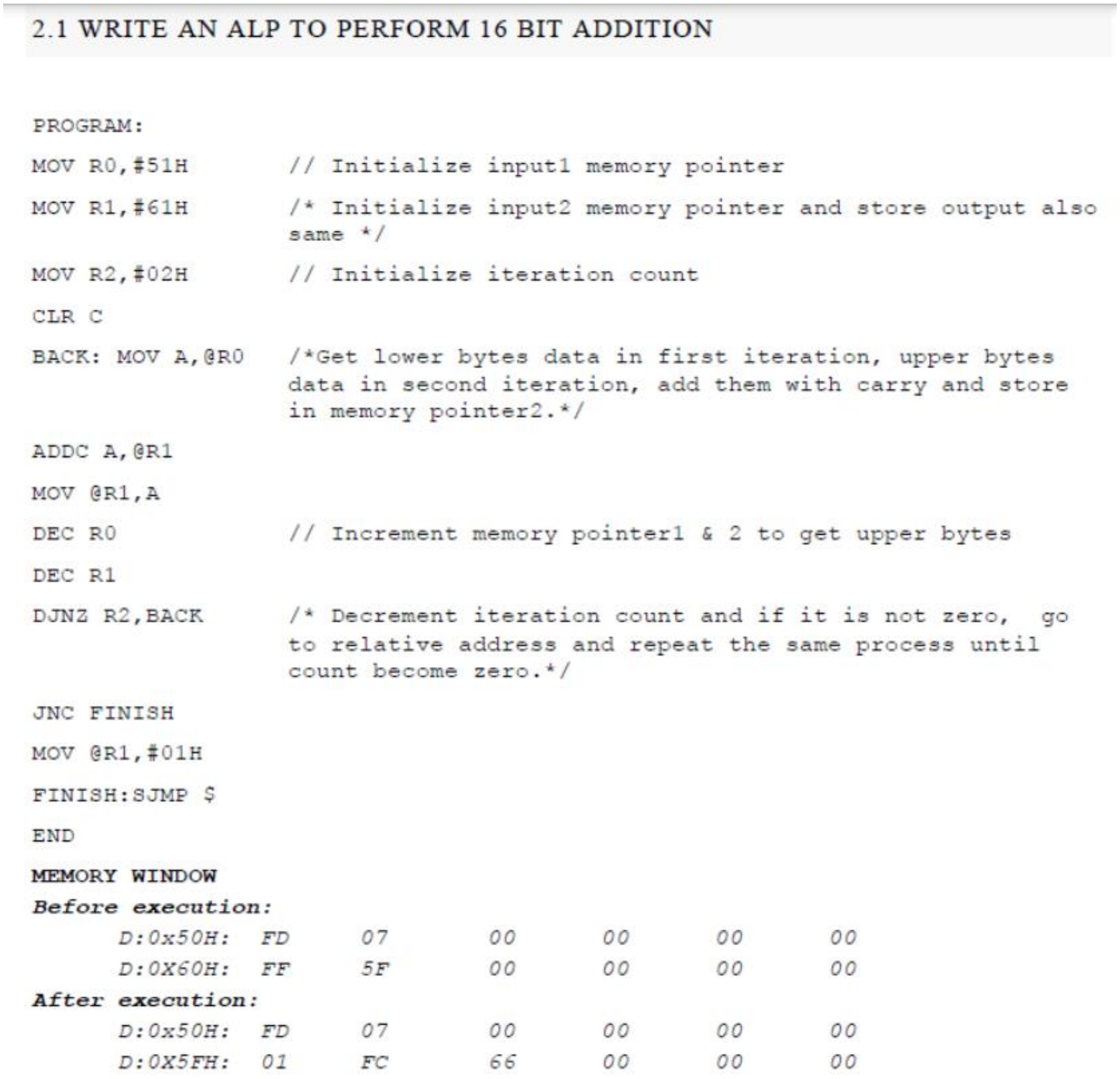
Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START:	8100	MOV	A,#15	74 15	Store data1 in accumulator
	8102	MOV	B,#03	75 F0 03	Store data2 in B register
	8105	DIV	AB	84	Divide
	8106	MOV	DPTR,#4500	90 85 00	Initialize memory location
	8109	MOVX	@DPTR,A	F0	Store remainder
	810A	INC	DPTR	A3	Go to next memory location
	810B	MOV	A,B	E5 F0	Store quotient
	810D	MOVX	@DPTR,A	F0	
L1	810E	SJMP	L1	80 FE	Stop the program

Input	
Memory Location	**Data**
8500 (dividend)	**0F**
8501 (divisor)	**03**

Output	
Memory Location	**Data**
8502 (remainder)	**05**
8503 (quotient)	**00**

RESULT:

Thus the 8051 8051Assembly Language Program for division of two 8 bit numbers was executed.



2.1 WRITE AN ALP TO PERFORM 16 BIT ADDITION

```
PROGRAM:
MOV R0,#51H        // Initialize input1 memory pointer
MOV R1,#61H        /* Initialize input2 memory pointer and store output also
                   same */
MOV R2,#02H        // Initialize iteration count
CLR C
BACK: MOV A,@R0    /*Get lower bytes data in first iteration, upper bytes
                   data in second iteration, add them with carry and store
                   in memory pointer2.*/
ADDC A,@R1
MOV @R1,A
DEC R0             // Increment memory pointer1 & 2 to get upper bytes
DEC R1
DJNZ R2,BACK       /* Decrement iteration count and if it is not zero,  go
                   to relative address and repeat the same process until
                   count become zero.*/
JNC FINISH
MOV @R1,#01H
FINISH:SJMP $
END
```

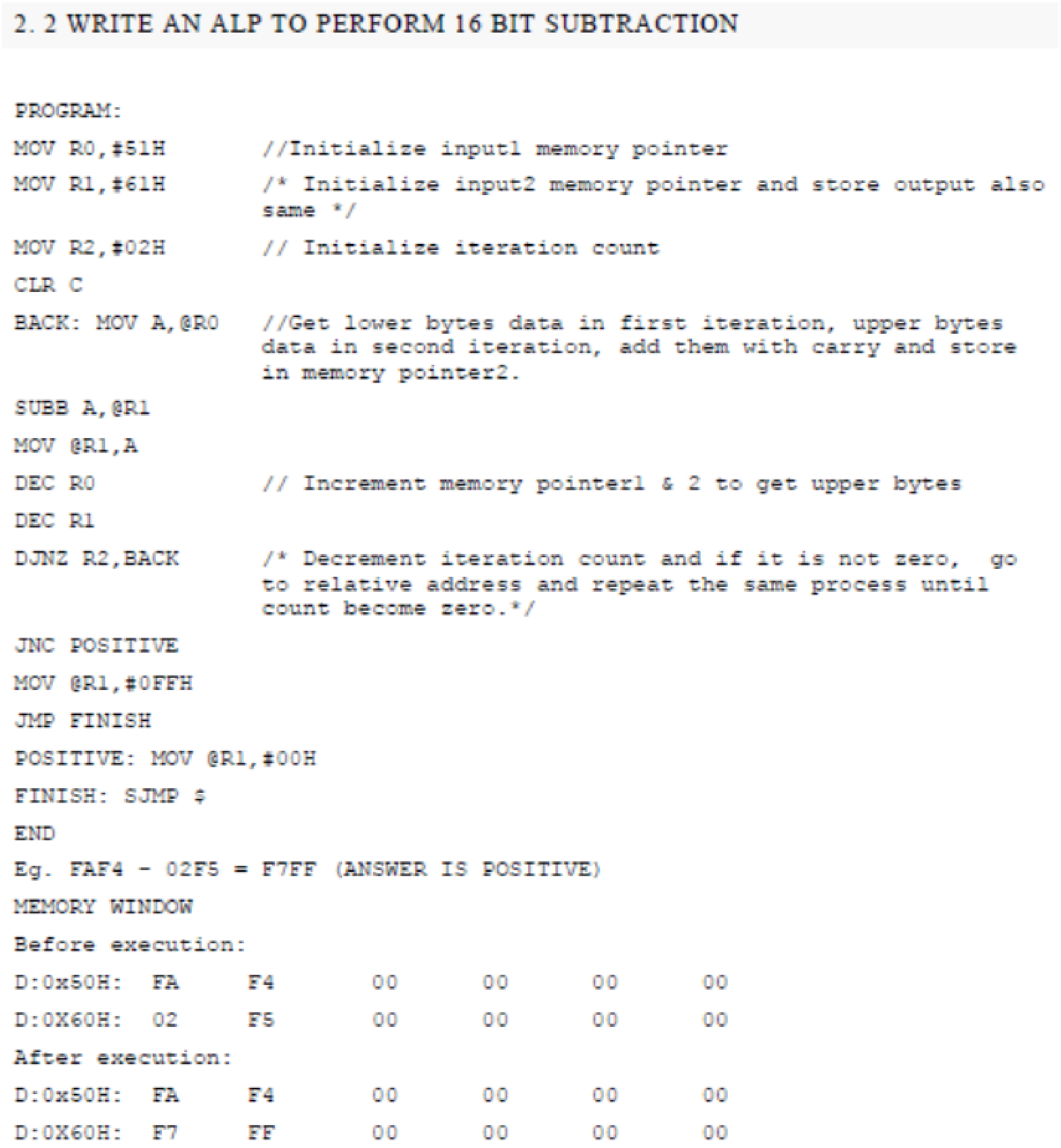
MEMORY WINDOW

Before execution:

```
      D:0x50H:  FD     07      00      00      00      00
      D:0X60H:  FF     5F      00      00      00      00
```

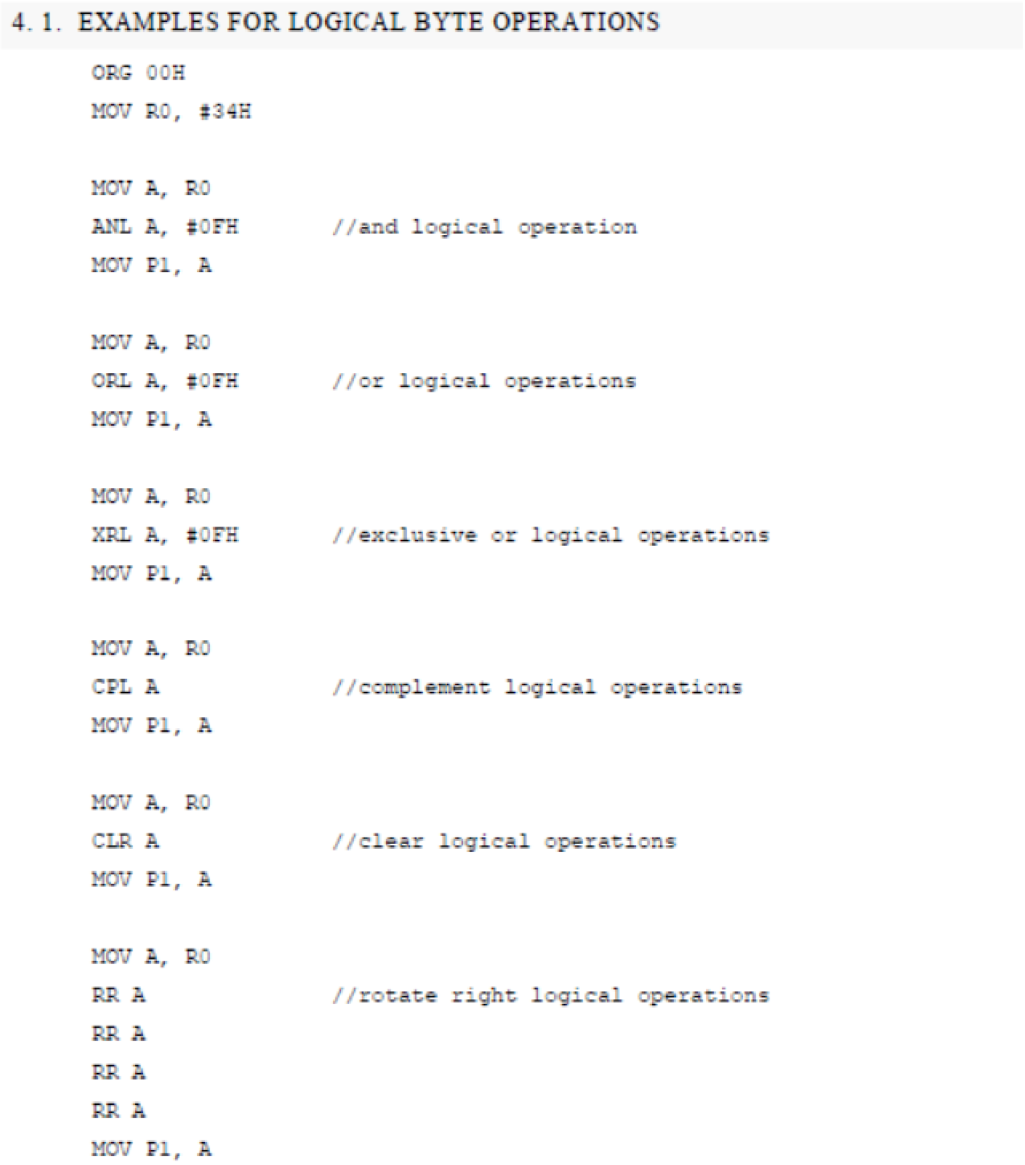
After execution:

```
      D:0x50H:  FD     07      00      00      00      00
      D:0X5FH:  01     FC      66      00      00      00
```



2. 2 WRITE AN ALP TO PERFORM 16 BIT SUBTRACTION

```
PROGRAM:
MOV R0,#51H         //Initialize input1 memory pointer
MOV R1,#61H         /* Initialize input2 memory pointer and store output also
                    same */
MOV R2,#02H         // Initialize iteration count
CLR C
BACK: MOV A,@R0     //Get lower bytes data in first iteration, upper bytes
                    data in second iteration, add them with carry and store
                    in memory pointer2.
SUBB A,@R1
MOV @R1,A
DEC R0              // Increment memory pointer1 & 2 to get upper bytes
DEC R1
DJNZ R2,BACK        /* Decrement iteration count and if it is not zero,  go
                    to relative address and repeat the same process until
                    count become zero.*/
JNC POSITIVE
MOV @R1,#0FFH
JMP FINISH
POSITIVE: MOV @R1,#00H
FINISH: SJMP $
END
Eg. FAF4 - 02F5 = F7FF (ANSWER IS POSITIVE)
MEMORY WINDOW
Before execution:
D:0x50H:  FA     F4       00       00       00       00
D:0X60H:  02     F5       00       00       00       00
After execution:
D:0x50H:  FA     F4       00       00       00       00
D:0X60H:  F7     FF       00       00       00       00
```



4. 1. EXAMPLES FOR LOGICAL BYTE OPERATIONS

```
ORG 00H
MOV R0, #34H

MOV A, R0
ANL A, #0FH       //and logical operation
MOV P1, A

MOV A, R0
ORL A, #0FH       //or logical operations
MOV P1, A

MOV A, R0
XRL A, #0FH       //exclusive or logical operations
MOV P1, A

MOV A, R0
CPL A             //complement logical operations
MOV P1, A

MOV A, R0
CLR A             //clear logical operations
MOV P1, A

MOV A, R0
RR A              //rotate right logical operations
RR A
RR A
RR A
MOV P1, A
```

EXP NO:	Generation of Square waveform using 8051.
DATE:	

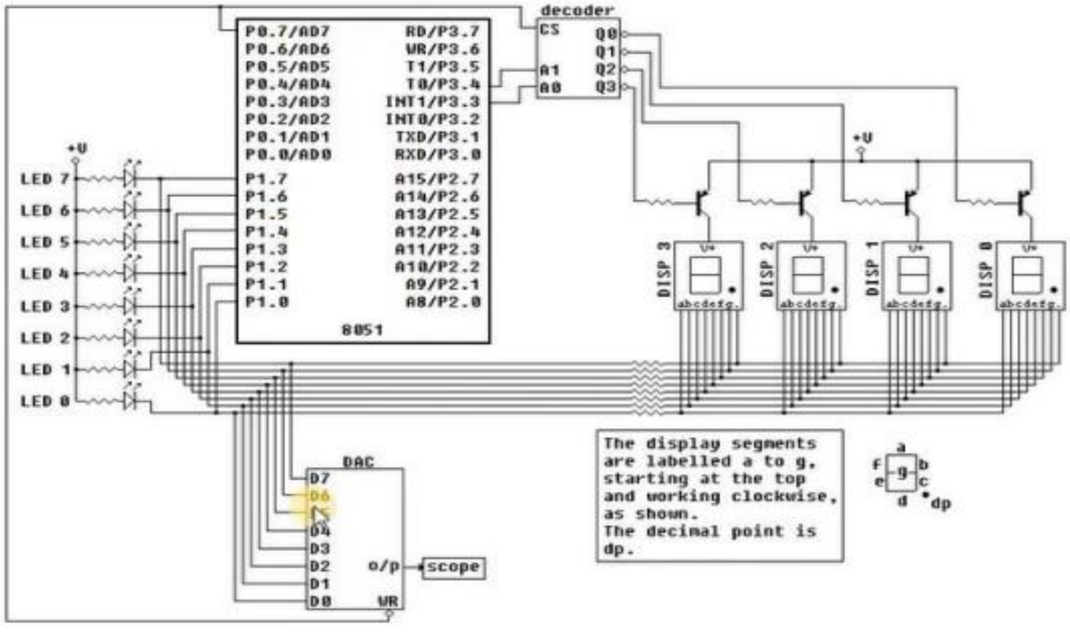
AIM:

To interface the DAC with the 8051 microcontroller and generate the square waveform

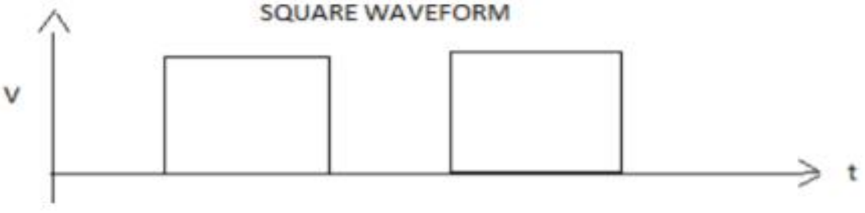
REQUIREMENTS:

- Edsim51DI simulator/8051 trainer kit/DAC interfacing Board/CRO

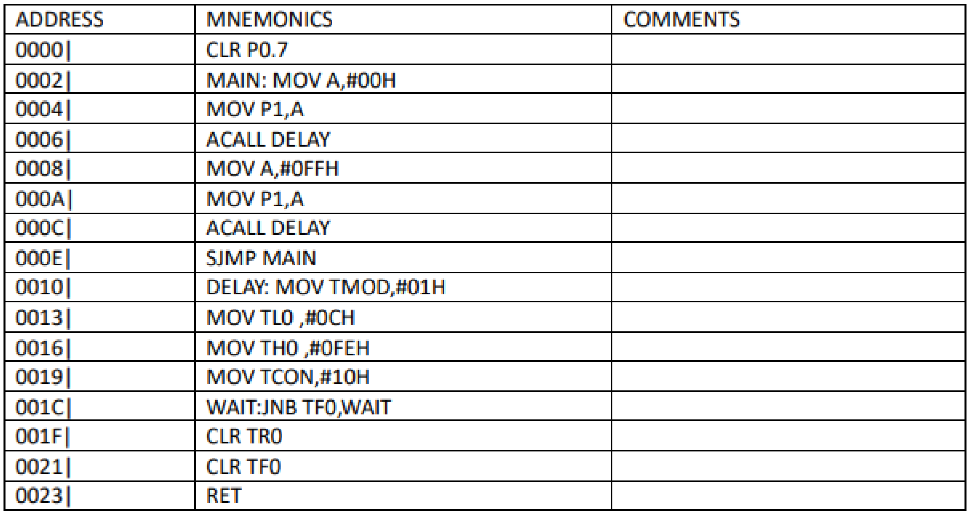
CIRCUIT DIAGRAM:



WAVEFORMS:



PROGRAM:



ADDRESS	MNEMONICS	COMMENTS
0000|	CLR P0.7	
0002|	MAIN: MOV A,#00H	
0004|	MOV P1,A	
0006|	ACALL DELAY	
0008|	MOV A,#0FFH	
000A|	MOV P1,A	
000C|	ACALL DELAY	
000E|	SJMP MAIN	
0010|	DELAY: MOV TMOD,#01H	
0013|	MOV TL0 ,#0CH	
0016|	MOV TH0 ,#0FEH	
0019|	MOV TCON,#10H	
001C|	WAIT:JNB TF0,WAIT	
001F|	CLR TR0	
0021|	CLR TF0	
0023|	RET	

8051 TRAINER KIT PROGRAM

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
FLASH:	8100	CPL	P1.4	B2 94	
	8102	LCALL	8E50	12 8E 50	
	8105	SJMP	8100	80 59	
DELAY:	8E50	PUSH	00	C0 00	
	8E52	PUSH	01	C0 01	
	8E54	PUSH	02	C0 02	
	8E56	MOV	02,#01	75 02 01	
USER_L2:	8E59	MOV	01,#FF	75 01 FF	
USER_L1:	8E5C	MOV	00,#FF	75 02 FF	
	8E5F	DJNZ	00,8E5F	D5 00 FE	
	8E62	DJNZ	01,8E5C	D5 01 F7	
	8E65	DJNZ	02,8E59	D5 02 F1	
	8E68	POP	02	D0 02	
		POP	01	D0 01	
		POP	00	D0 00	
		RET			

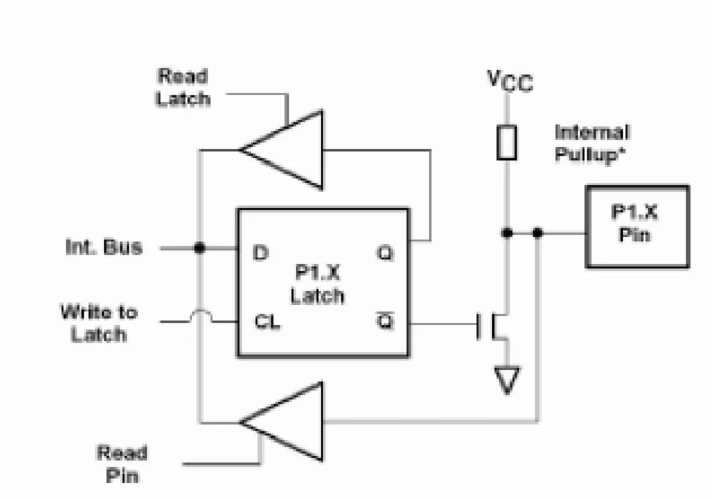
RESULT: Thus, the assembly language program for performing the interfacing of DAC with 8051 has been verified.

EXP NO:	Programming using on – Chip ports in 8051.
DATE:	

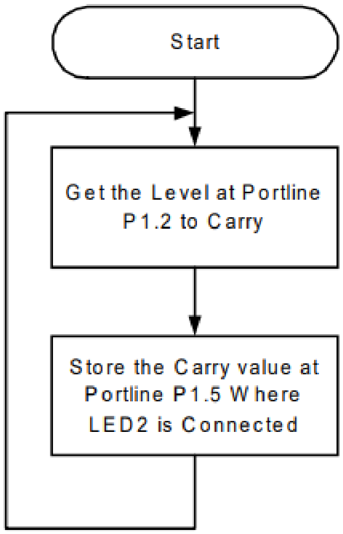
AIM:

To read the status of the switch connected to port line p1.2, p3.2 and display it on led connected to port line p1.5, p1.6

CIRCUIT:



Flow Chart:


Start
Get the Level at Portline P1.2 to Carry
Store the Carry value at Portline P1.5 Where LED2 is Connected

PROGRAM:

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START:	8200	MOV	C, p1.2		
		MOV	P1.5, C		
		MOV	C, p3.3		
		MOV	P1.6, C		
		MOV	C,P3.4		
		MOV	P1.4,C		
		SJMP	8200		

PROGRAM

(a) Keep monitoring switch (at) P1.2 until it becomes high
(b) When P0.1 becomes high, Light LEDs connected at port 2

```
 SETB P1.2 ;                                    make P1.2 as input (switch)
MOV A,#FFH    ;                                 A=11111111
AGAIN: JNB P1.2,AGAIN ;                      get out when P1.2=1
MOV P1.5,A ;                               Light LEDS by sending 1s to P2
SJMP 8200
```

RESULT:

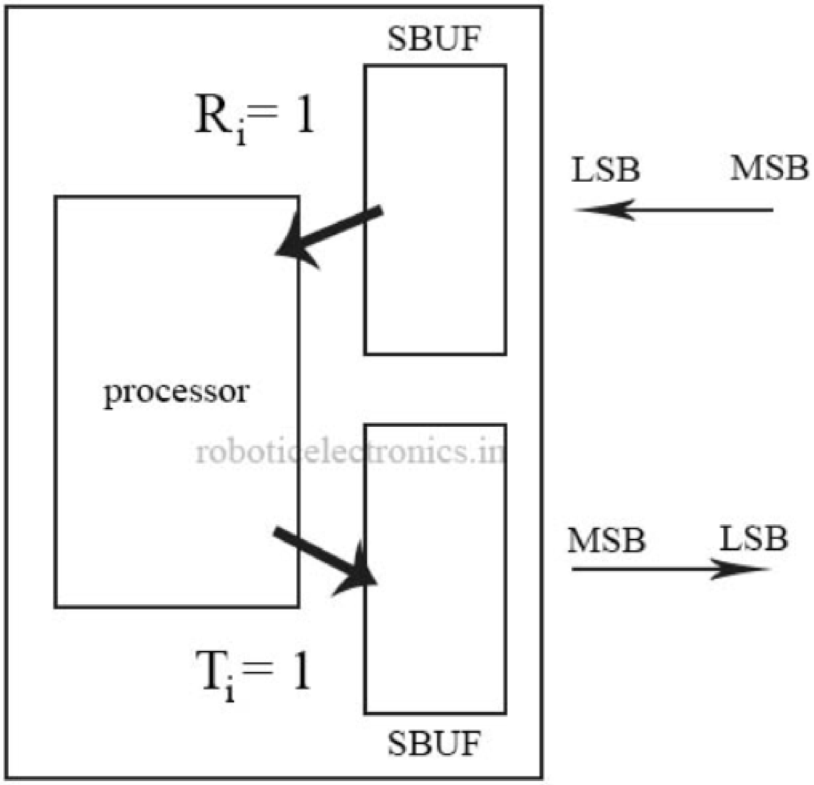
Thus, the assembly language Programming using on – Chip ports in 8051has been verified.

EXP NO:	Programming using Serial Ports in 8051.
DATE:	

AIM:

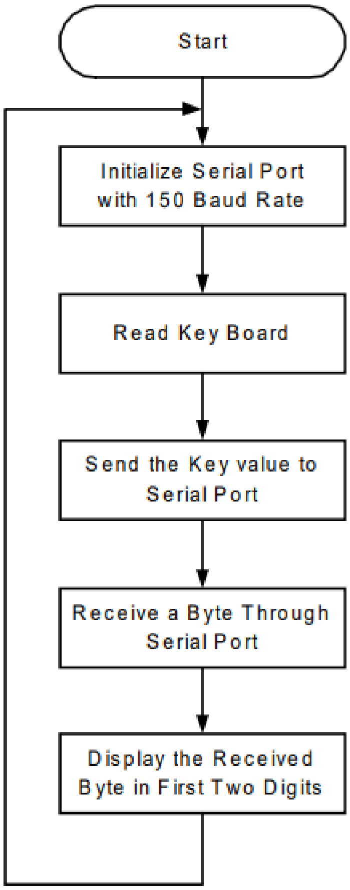
To write a program for the 8051 to transfer character serially at 150 baud rate.

Block diagram:



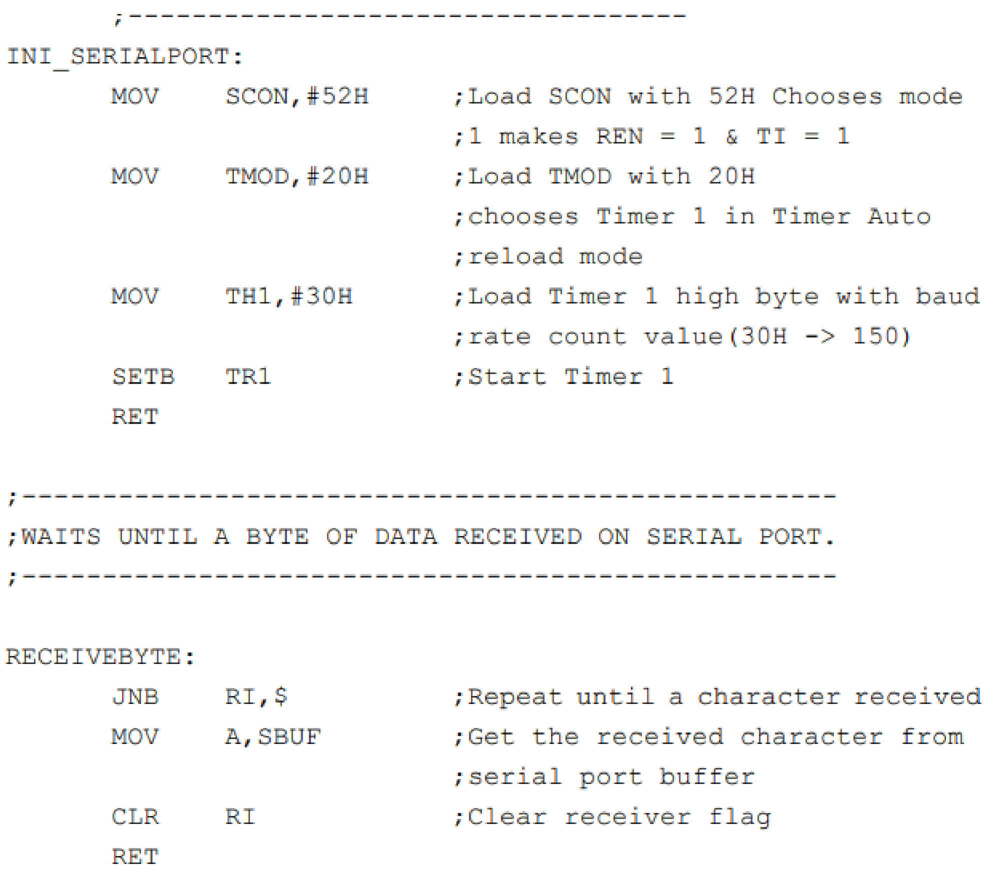
, ---

Flow Chart:


Start
Initialize Serial Port with 150 Baud Rate
Read Key Board
Send the Key value to Serial Port
Receive a Byte Through Serial Port
Display the Received Byte in First Two Digits

PROGRAM:

```
8650:                           LCALL 866E
         STAT SERIAL: MOV DPTR, #0000
                                LCALL 1030
      WAIT FOR KEY: LCALL 1070
                                ANL A,#07
                                JZ 8659
                                LCALL 1080
                                LCALL 8682
                                LCALL 867A
                                LCALL 1050
                               SJMP 8655
```

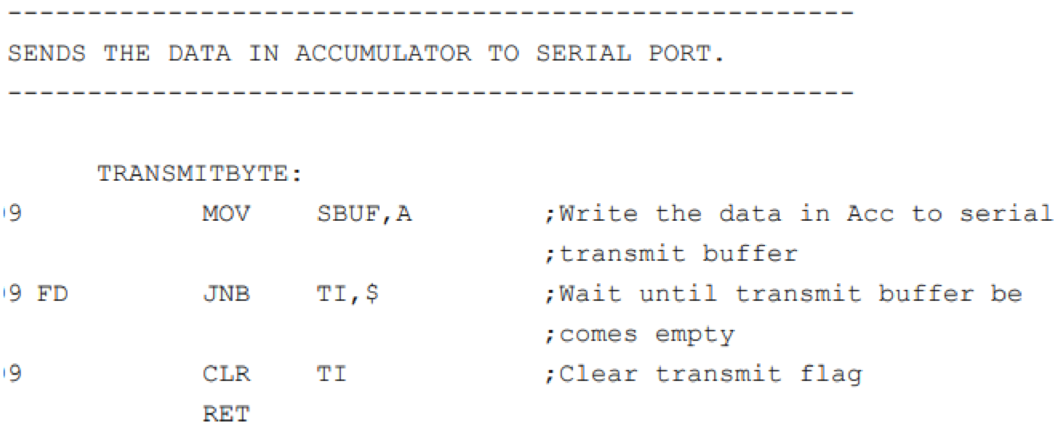


```
        ;-----------------------------------
INI_SERIALPORT:
        MOV     SCON,#52H       ;Load SCON with 52H Chooses mode
                                ;1 makes REN = 1 & TI = 1
        MOV     TMOD,#20H       ;Load TMOD with 20H
                                ;chooses Timer 1 in Timer Auto
                                ;reload mode
        MOV     TH1,#30H        ;Load Timer 1 high byte with baud
                                ;rate count value(30H -> 150)
        SETB    TR1             ;Start Timer 1
        RET

;---------------------------------------------------
;WAITS UNTIL A BYTE OF DATA RECEIVED ON SERIAL PORT.
;---------------------------------------------------

RECEIVEBYTE:
        JNB     RI,$            ;Repeat until a character received
        MOV     A,SBUF          ;Get the received character from
                                ;serial port buffer
        CLR     RI              ;Clear receiver flag
        RET
```

JNB RI,867A



```
----------------------------------------------------
SENDS THE DATA IN ACCUMULATOR TO SERIAL PORT.
----------------------------------------------------

     TRANSMITBYTE:
'9          MOV     SBUF,A          ;Write the data in Acc to serial
                                    ;transmit buffer
'9 FD       JNB     TI,$            ;Wait until transmit buffer be
                                    ;comes empty
'9          CLR     TI              ;Clear transmit flag
            RET
```

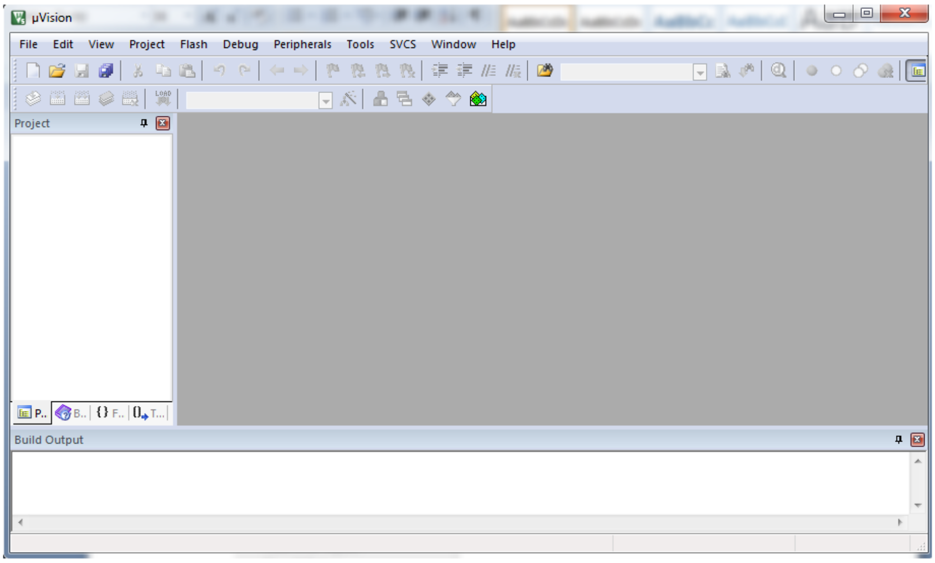
JNB TI,8684

RESULT:

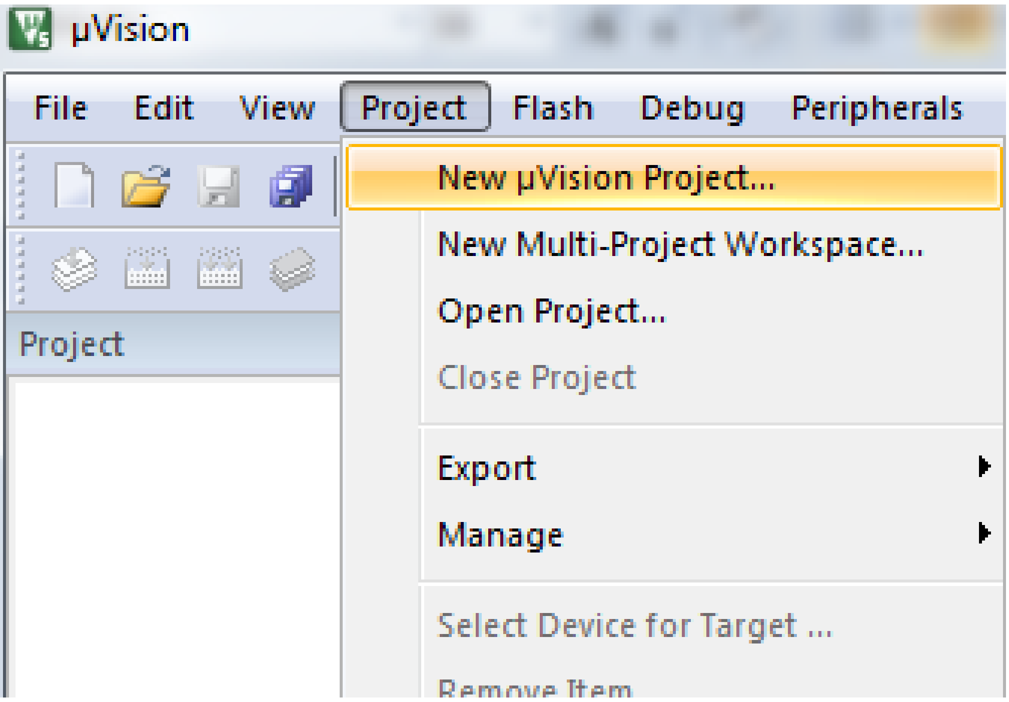
INTRODUCTION TO KEIL µ VISION SOFTWARE

PROCEDURE

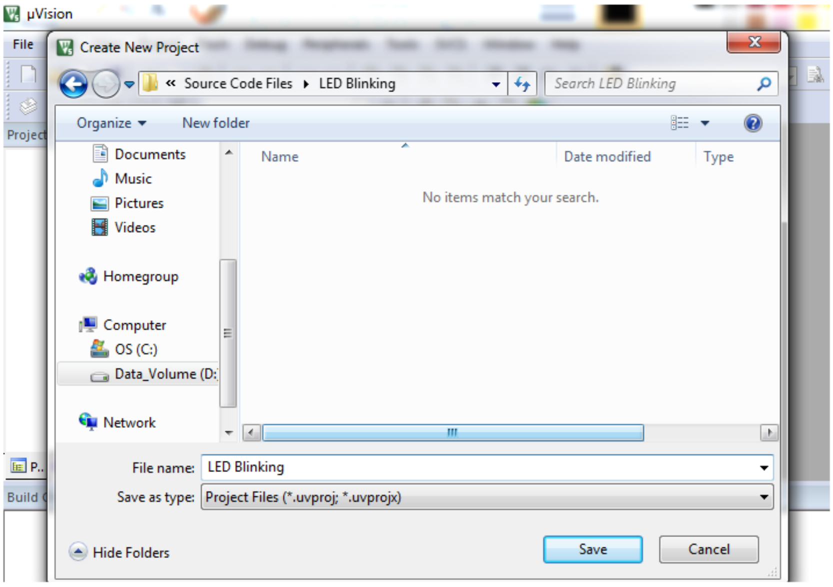
1. Open Keil µVision from the icon created on your desktop.


µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Project
P.. B.. {} F.. 0→ T...
Build Output

2. Go to the **Project** tab. Select New **µVision Project ...**from that menu.


µVision
File Edit View Project Flash Debug Peripherals
New µVision Project...
New Multi-Project Workspace...
Open Project...
Close Project
Export
Manage
Select Device for Target ...
Remove Item
Project

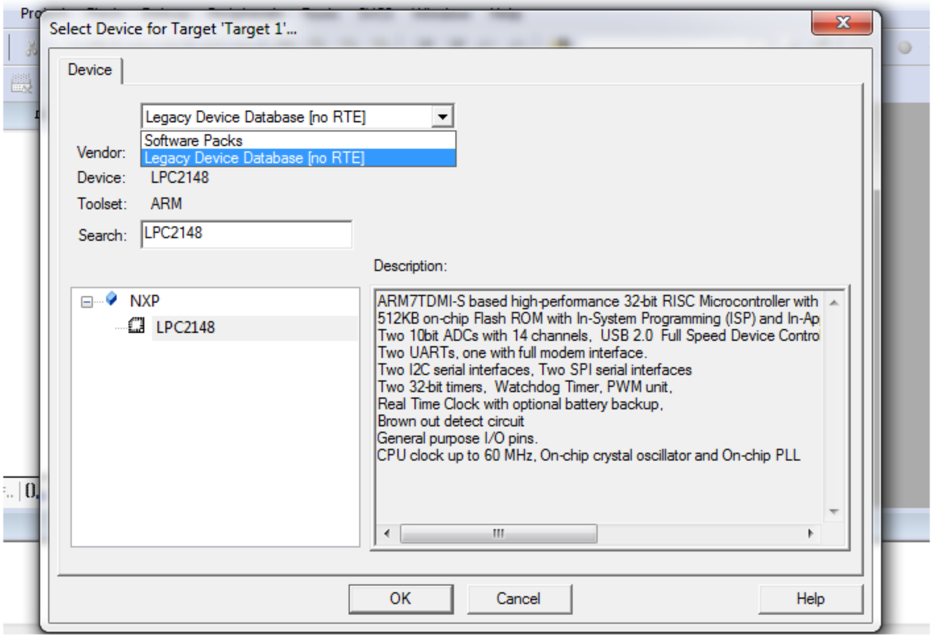
3. **Create New Project** window will pop up. Select the folder where you want to create project and give a suitable name to the project. Then click on **Save**.



File name: LED Blinking

Save as type: Project Files (*.uvproj; *.uvprojx)

4. **Select Device for Target: 'Target1'...**window will pop up next. It has a select window to choose between Software Packs or Legacy Device Database. As LPC2148 is in Legacy Device Database, choose Legacy Device Database.


Select Device for Target 'Target 1'...
Device
Software Packs
Legacy Device Database [no RTE]
NXP
LPC2148
OK
Cancel
Help

Legacy Device Database [no RTE]

Vendor:

Device: LPC2148

Toolset: ARM

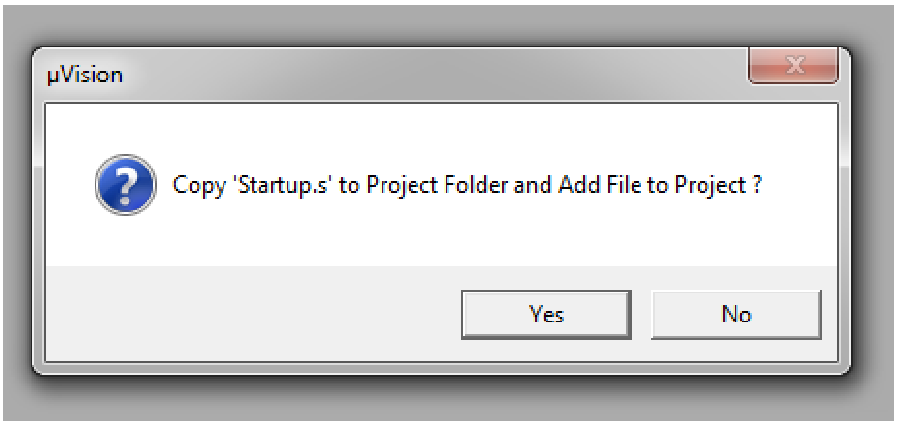
Search: LPC2148

Description:

ARM7TDMI-S based high-performance 32-bit RISC Microcontroller with 512KB on-chip Flash ROM with In-System Programming (ISP) and In-Ap
Two 10bit ADCs with 14 channels, USB 2.0 Full Speed Device Contro
Two UARTs, one with full modem interface.
Two I2C serial interfaces, Two SPI serial interfaces
Two 32-bit timers, Watchdog Timer, PWM unit,
Real Time Clock with optional battery backup,
Brown out detect circuit
General purpose I/O pins.
CPU clock up to 60 MHz, On-chip crystal oscillator and On-chip PLL

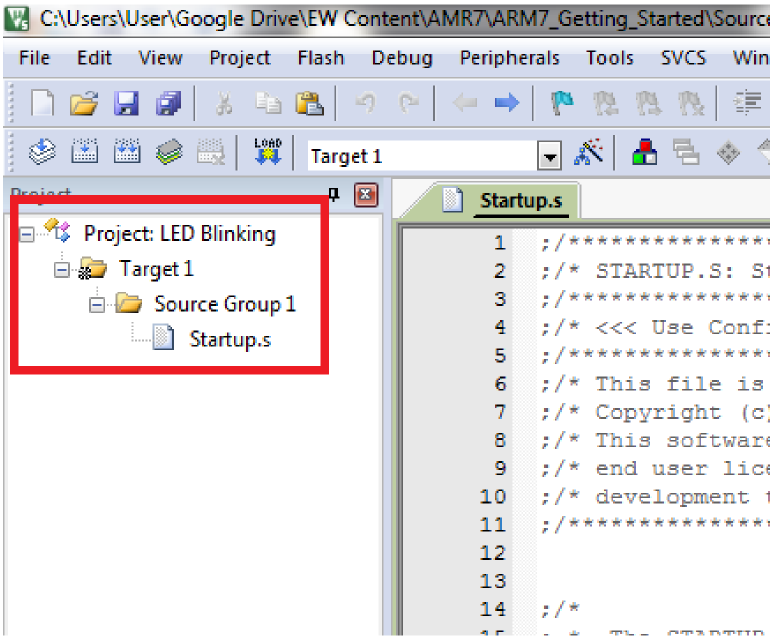
Type in LPC2148 in search and select the device under NXP with the name LPC2148 and click on OK.

5. A window will pop up asking whether to copy Startup.s to project folder and add file to project. Click on **Yes**.



Copy 'Startup.s' to Project Folder and Add File to Project ?

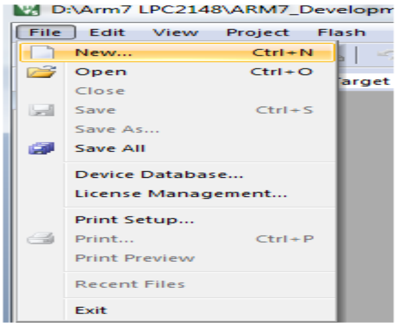
6. The project name and its folders can be seen on the left side in the project window after the previous step is completed as shown below.



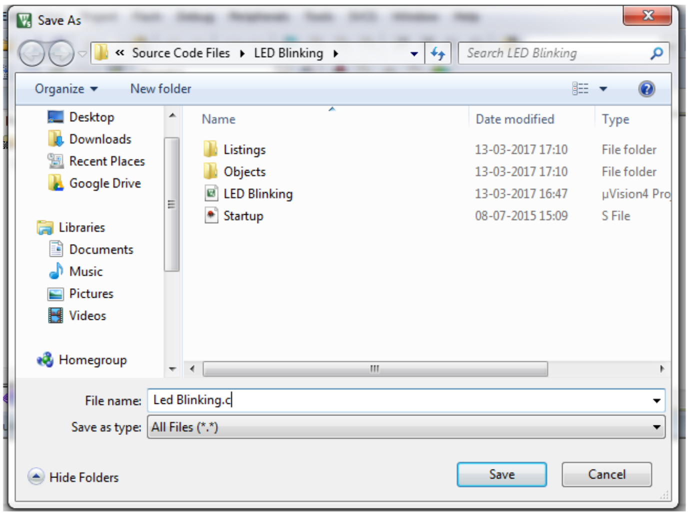
```
;/**************
;/* STARTUP.S: St
;/**************
;/* <<< Use Conf
;/**************
;/* This file is
;/* Copyright (c
;/* This softwar
;/* end user lic
;/* development
;/**************


;/*
```

7. Now go to File tab and add **New** file from the menu.



8. Save the file from the previous step with a specific name. Add .c extension to the file name.

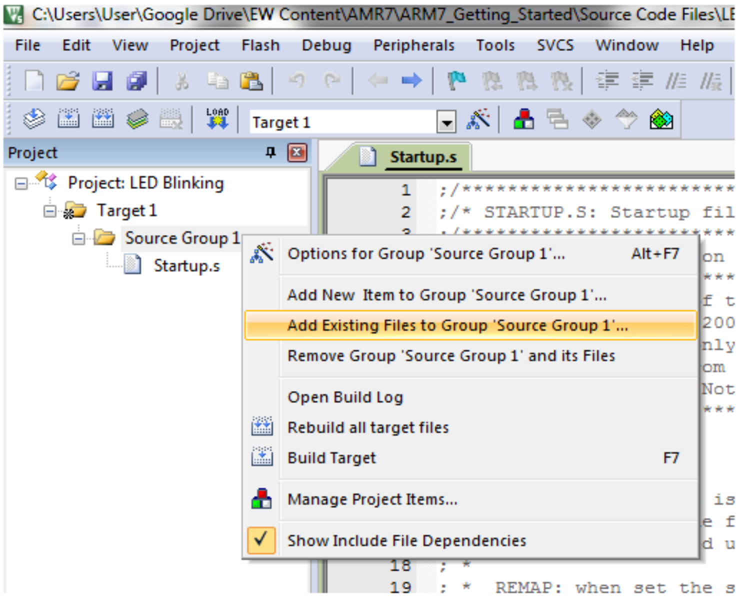


Name	Date modified	Type
Listings	13-03-2017 17:10	File folder
Objects	13-03-2017 17:10	File folder
LED Blinking	13-03-2017 16:47	µVision4 Pro
Startup	08-07-2015 15:09	S File

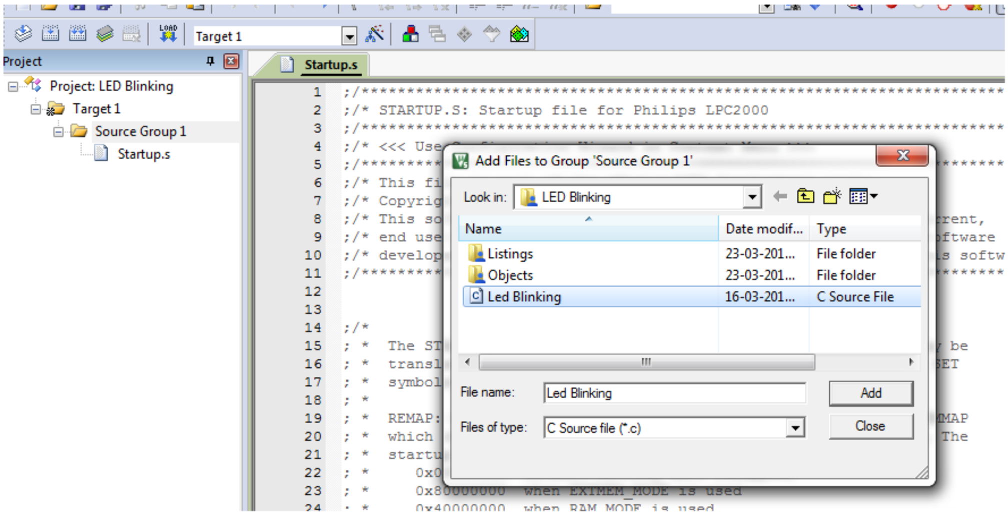
File name: Led Blinking.c

Save as type: All Files (*.*)

9. Add this file to Source Group folder in the project window by right clicking on Source Group1 folder and selecting **Add Existing Files to Group ‘Source Group1’.**


C:\Users\User\Google Drive\EW Content\AMR7\ARM7_Getting_Started\Source Code Files\LE

```
;/*************************
;/* STARTUP.S: Startup fil
;/*************************
; *  REMAP: when set the s
```



```
;/*****************************************************************************
;/* STARTUP.S: Startup file for Philips LPC2000
;/*****************************************************************************
```

Look in: LED Blinking

Name	Date modif...	Type
Listings	23-03-201...	File folder
Objects	23-03-201...	File folder
Led Blinking	16-03-201...	C Source File

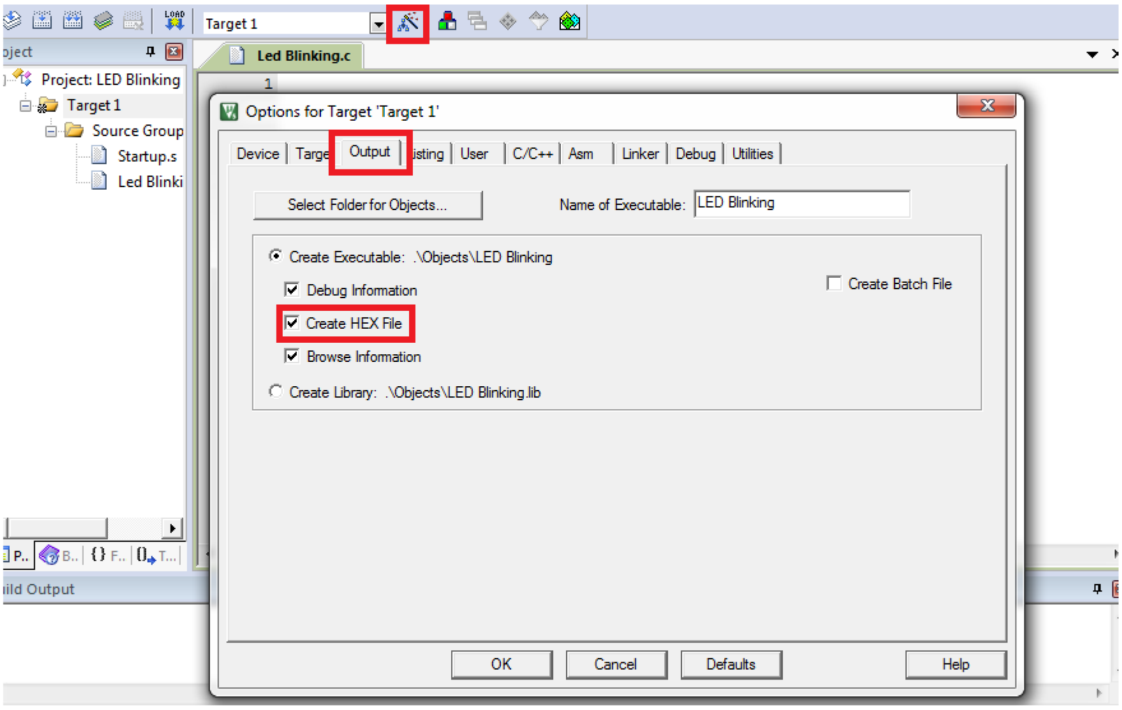
File name: Led Blinking

Files of type: C Source file (*.c)

Select the previously saved file from the window that pops up and add it to the Source Group1

10. Now click on the **Options for Target ‘Target1’...** symbol shown in red box in the image below or press **Alt+F7** or right click on Target1 and click on **Options for Target ‘Target1’....**

Options for target window will open. Go to the **Output** tab in that window. Tick ‘√’ **Create HEX File** option. We need to produce HEX file to burn it into the microcontroller.



Name of Executable: LED Blinking

● Create Executable: .\Objects\LED Blinking

☑ Debug Information

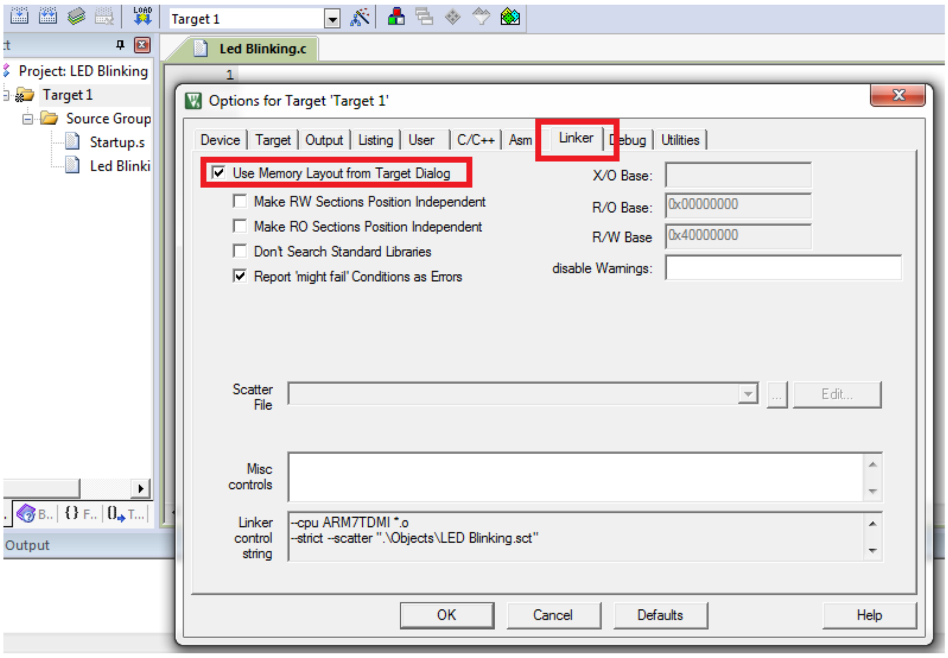
☑ Create HEX File

☑ Browse Information

○ Create Library: .\Objects\LED Blinking.lib

□ Create Batch File

In the options for target window, go to the **Linker** tab. Select the **Use Memory Layout from Target Dialogue** option.



☑ Use Memory Layout from Target Dialog
☐ Make RW Sections Position Independent
☐ Make RO Sections Position Independent
☐ Dont Search Standard Libraries
☑ Report 'might fail' Conditions as Errors

X/O Base:
R/O Base: 0x00000000
R/W Base 0x40000000
disable Warnings:

Scatter File

Misc controls

Linker control string
```
--cpu ARM7TDMI *.o
--strict --scatter ".\Objects\LED Blinking.sct"
```

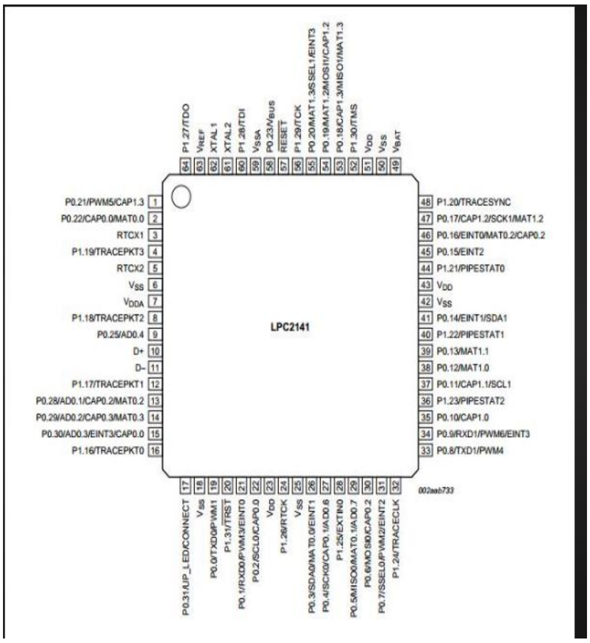
11. Now write the code in c file

12. Once the code is written, **Build** the code by clicking on the button shown in red in the image below. You can also build the project from the **Build Target** option in the Project tab or by pressing **F7** on the keyboard.

You can see **creating hex file ...** in the Build Output window as shown in the image.

13. Once the project is built, a **hex file** is created in the **Objects** folder inside the folder of your project. Use **Flash Magic** software to burn this hex file in your microcontroller.

ARM LPC 2148 PIN DETAIL



EXP NO: **DATE**	**LED & FLASHING OF LED'S**

AIM:

To write and execute the program for LED & Flashing Led's with ARM7 (LPC2148) processor.

HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	ARM Processor board	1
2	USB/FRC Connector	few
3	LED Module	1
4	Power Supply adaptor (5V, DC)	1
5	Keil & flash magic Software	1

PROCEDURE

1. Create a New project, Go to "Project" and close the current project "Close Project".
2. Next Go to the Project New µvision Project Create New Project Select Device for Target.
3. Select the data base NXP LPC2148.
4. Add Startup file and Next go to "File" and click "New".
5. Write a program on the editor window and save as "Main.c".
6. Add this source file to Group and click on "Build Target" or F7.
7. Create a Hex file from "Project" menu and click on "Rebuild all target Files".
8. Open Flash magic and select the device LPC2148 in ARM 7 category, Choose the hardware connected COM port, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.
9. Next browse the path of hex file and select the file.
10. After selecting ISP mode on the Hardware Kit and click on start then device will start to program

PROGRAM:

```
#include <LPC214X.H>
void Delay(int);
int main (void)
{
        IODIR0 = 0x00580000;
        IODIR1 = 0x00FF0000;

        while(1)
        {

                        IOCLR1 = 0x00030000;                    // Clear LED line
                        IOCLR0 = 0x00580000;                    // Clear A0,A1 and A2
                        IOSET0 = 0x00580000;                    // Set A0,A1 and A2
                        Delay(10);
                        IOSET1 = 0x00030000;                    // Switch on one the line
                        IOCLR0 = 0x00580000;                    // Clear A0,A1 and A2
                        IOSET0 = 0x00580000;                    // Set A0,A1 and A2
                        Delay(10);                                      // Delay
        }
}
void Delay(int n)
{
        int p,q;
        for(p=0;p<n;p++)
        {
                for(q=0;q<0x99900;q++);
        }
}
```

PEOGRAM:2

```
#include<nxp/iolpc2148.H>
void delay()
{
 for(int i=0x00;i<=0xff;i++)
 for(int j=0x00;j<=0xFf;j++);
}
// LED INTERFACE LINES
// LED0   -   LED7 :  P1.24 - P1.31
void main()
{
     PINSEL2 = 0X00000000;       // P1.24 TO P1.31 as GPIO
     IO1DIR  = 0XFF000000;       // p1.24 TO P1.31 Configured as O
utput port.
   while(1)
   {
    IO1SET=0XFF000000;           // P1.24 TO P1.31 goes to high sta
te
    delay();
    IO1CLR=0XFF000000;           // P1.24 TO P1.31 goes to low stat
e
    delay();
   }
```

RESULT:

EXP NO:	INTERFACING OF LCD
DATE	

AIM:

To write and execute the program for LCD with ARM7 (LPC2148) processor.

HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	ARM Processor board	1
2	USB/FRC Connector	few
3	LED Module	1
4	Power Supply adaptor (5V, DC)	1
5	Keil & flash magic Software	1

PROCEDURE

1. Create a New project, Go to "Project" and close the current project "Close Project".
2. Next Go to the Project New µvision Project Create New Project Select Device for Target.
3. Select the data base NXP LPC2148.
4. Add Startup file and Next go to "File" and click "New".
5. Write a program on the editor window and save as "Main.c".
6. Add this source file to Group and click on "Build Target" or F7.
7. Create a Hex file from "Project" menu and click on "Rebuild all target Files".
8. Open Flash magic and select the device LPC2148 in ARM 7 category, COM port will be COM 3, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.
9. Next browse the path of hex file and select the file.
10. After selecting ISP mode on the Hardware Kit and click on start then device will start to program

PROGRAM:

```
/************************************************/
/* LCD Routines for 2 line X 16 Characters Display*/
/************************************************/
#include "LPC214x.h"                    /* LPC214x definitions  */
void WriteCommandLCD(unsigned char CommandByte);
void WriteDataLCD(unsigned char DataByte);
void LCDDelay(void);
void LCDDelay1600(void);
void SendByte(unsigned char Value);
void InitializeLCD(void);

void DataAddressDirection(void);
void DisplayLCD(char LineNumber,char *Message);
void DisplayLCD2Digit(char LineNumber,char CharPosition,char Data);
int main(void)
{
        InitializeLCD();                                                //Initialize graphics LCD
        DisplayLCD(0,"  NXP2148 ARM   ");
        DisplayLCD(1,"Evaluatin System");
        while(1);
}
void __gccmain()
{

}
```

RESULT:

EXP NO:	**INTERFACING OF MATRIX KEYBOARD**
DATE	

AIM:

To write and execute the program for Matrix Keyboard with ARM7 (LPC2148) processor.

HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	ARM Processor board	1
2	USB/FRC Connector	few
3	LCD Module	1
4	Power Supply adaptor (5V, DC)	1
5	Keil & flash magic Software	1
6	Matrix Keypad Module	1

PROCEDURE

1. Create a New project, Go to “Project” and close the current project “Close Project”.
2. Next Go to the Project New μvision Project Create New Project Select Device for Target.
3. Select the data base NXP LPC2148.
4. Add Startup file and Next go to “File” and click “New”.
5. Write a program on the editor window and save as “Main.c”.
6. Add this source file to Group and click on “Build Target” or F7.
7. Create a Hex file from “Project” menu and click on “Rebuild all target Files”.
8. Open Flash magic and select the device LPC2148 in ARM 7 category, COM port will be COM 3, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.
9. Next browse the path of hex file and select the file.
10. After selecting ISP mode on the Hardware Kit and click on start then device will start to program

PROGRAM

```
/**********************************************/
/* Matrix Keyboard with LCD Display*/
/*********************************************/
#include "LPC214x.H"                    // LPC214x definitions

irq void T0_Srv(void);
void InitializeTIMER0(void);
void InitializeKeyboard(void);
void ScanKeyboard(void);
unsigned char ReadColumn(void);
void SetRow(unsigned char dat);
void DisplayLCD(char LineNumber,char *Message);

char KeyboardFlag;
unsigned char KeyboardCode=0;
char Counter=0,ReleaseFlag=0,StatusFlag=0,IdentificationFlag=0;

void WriteCommandLCD(unsigned char CommandByte);
void WriteDataLCD(unsigned char DataByte);
void LCDDelay(void);
void LCDDelay1600(void);
void SendByte(unsigned char Value);
void InitializeLCD(void);

void DataAddressDirection(void);
void DisplayLCD(char LineNumber,char *Message);
void DisplayLCD2Digit(char LineNumber,char CharPosition,char Data);
```

```
int main(void)
{
        KeyboardFlag =0;                            //Clear keyboard flag
        InitializeLCD();
        IODIR0 |= 0x00078000;               //Initialize Port lines for keyboard
        InitializeTIMER0();                         //Initialize TIMER0, Keyboard scan timer
        DisplayLCD(0,"Matrix Keyboard ");
        DisplayLCD(1,"Key Pressed:    ");
        while(1)
        {
                if(KeyboardFlag ==1)        //wait for key press
                {
                        DisplayLCD2Digit(1,12,KeyboardCode);
                        KeyboardFlag=0;                     //reset keyboard flag
                }
        }
}

//Scan keyboard
void ScanKeyboard(void)
{
        unsigned char  a,b,c;
        unsigned char  f;
        if(ReleaseFlag == 1)
        {       //debounce time for release
                Counter++;
                if(Counter ==10)
                {
                        ReleaseFlag = 0;
                }
        return;
    }
    if(StatusFlag ==1)
    {
        if(IdentificationFlag == 1)
        {       //Wait for key Release
            SetRow(0x00);
            if(ReadColumn() == 0x0f)            //Check the keyboard status
            {       //No key pressed
                Counter = 0;
                ReleaseFlag = 1;
                IdentificationFlag = 0;
                StatusFlag= 0;
            }
            return;
        }
        else
        {
            Counter++;
            if(Counter == 10)
            {       //Check key press
                KeyboardCode = 0;
                SetRow(0x00);                           //Set all rows to 0
                if(ReadColumn() != 0x0f)    //Read column levels
                {       //any one key pressed
                    for(a=0;a<4;a++)
                    {       //Row Setting
                        f = ~(0x01 << a);       //set one row to 0
                        SetRow(f);
                        b = ReadColumn();     //read column levels
                    for(c=0;c<4;c++)
                    {   //Column checking
                        f = 0x01 << c;
                        if(( b & f) == 0)
                        {
//Pressed Key identified
                            IdentificationFlag = 1;
                            StatusFlag = 1;
                            KeyboardFlag=1;
                            KeyboardCode &= 0x0f;
                            return;
                        }

                        KeyboardCode++;
                    }
                }
            }
            IdentificationFlag  = 0;
            StatusFlag= 0;
            return;
        }
        else
        {
            return;
        }
    }
}
SetRow(0x00);
if(ReadColumn() != 0x0f)                                    //Check for any key press
{       // key press detected
    Counter=0;
				StatusFlag=1;
				IdentificationFlag=0;
			}
}
/* Sets the given data to Row */
void SetRow(unsigned char dat)
{
	if(dat & 0x01)
		IOSET0 = 0x00008000;
	else
		IOCLR0 = 0x00008000;
	if(dat & 0x02)
		IOSET0 = 0x00010000;
	else
		IOCLR0 = 0x00010000;
	if(dat & 0x04)
		IOSET0 = 0x00020000;
	else
		IOCLR0 = 0x00020000;
	if(dat & 0x08)
		IOSET0 = 0x00040000;
	else
		IOCLR0 = 0x00040000;
}
/* Reads the Column status and returns the same */
unsigned char ReadColumn(void)
{
	unsigned char a=0;

	a = (IOPIN0>>11) & 0x0f;
```

```
        return(a);
}
//Initialise timer0 used in keyboard scanning
void InitializeTIMER0(void)
{       //for 1msec delay
        VPBDIV              = 0x00000002;       //Configure the  VPB divider
                                                                        //CCLK/2 =
PCLK = 30MHz
        T0PR            = 0x0000012B;      //Load prescaler = 300, 30MHz/300=100KHz
        T0TCR               = 0x00000002;               //Reset counter and prescaler
        T0MCR               = 0x00000003; //On match reset the counter and generate an
interrupt
        T0MR0               = 0x00000064;//Set the cycle time, 100KHZ/100 = 1KHZ =
1ms
        T0TCR               = 0x00000001;   //enable timer
        VICVectAddr4 = (unsigned)T0_Srv;//Set the timer ISR vector address
        VICVectCntl4 = 0x00000024;                      //Set channel
        VICIntEnable |= 0x00000010;                     //Enable the interrupt
}
/* Timer0 interrupt service routine */
__irq void T0_Srv(void)
{
        ScanKeyboard();                                         //Check and update
keyboard status
        T0IR            |= 0x00000001;                  //Clear match 0 interrupt
        VICVectAddr  = 0x00000000;                      //Dummy write to signal end of
interrupt
}
```

RESULT:

EXP NO:	INTERFACING OF STEPPER MOTOR
DATE	

AIM:

To write and execute the program for Stepper Motor with ARM7 (LPC2148) processor.

HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	ARM Processor board	1
2	USB/FRC Connector	few
3	Stepper Motor Module	1
4	Power Supply adaptor (5V, DC)	1
5	Keil & flash magic Software	1

PROCEDURE

1. Create a New project, Go to "Project" and close the current project "Close Project".
2. Next Go to the Project New µvision Project Create New Project Select Device for Target.
3. Select the data base NXP LPC2148.
4. Add Startup file and Next go to "File" and click "New".
5. Write a program on the editor window and save as "Main.c".
6. Add this source file to Group and click on "Build Target" or F7.
7. Create a Hex file from "Project" menu and click on "Rebuild all target Files".
8. Open Flash magic and select the device LPC2148 in ARM 7 category, COM port will be COM 3, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.
9. Next browse the path of hex file and select the file.
10. After selecting ISP mode on the Hardware Kit and click on start then device will start to program

PROGRAM:

```
/****************************************/
/* Program to drive a stepper motor*/
/***************************************/
#include "LPC214X.H"
void InitializeIO(void);
int main (void)
{
        unsigned char a;
        unsigned int b;
        long c,d;
        InitializeIO();                                         //Initialize the I/O lines
        while(1)
        {
                for(d=0;d<200;d++)
                { // Rotate clock wise
                        b=0x2222;
                        for(a=0;a<4;a++)
                        {
                IOPIN1 = (b & 0xff00) << 8 ;
                                IOCLR0 = 0x00400000;                    // Clear A3
                                IOSET0 = 0x00400000;                    // Set A3
                                for(c=0;c<0xa000;c++);              // Delay
                                b = b << 1;                         // Shift left side one
                        }
                }
                for(d=0;d<200;d++)
                {       // Rotate counter clock wise
                        b=0x8888;
```

```
            for(a=0;a<4;a++)
            {
        IOPIN1 = (b & 0xff00) << 8 ;
                IOCLR0 = 0x00400000;            / Clear A3
                IOSET0 = 0x00400000;            // Set A3
                for(c=0;c<0xa000;c++);          // Delay
                b = b >> 1;                     // Shift right side one
            }
        }
    }
}
void InitializeIO(void)
{
    IODIR0 = 0x00580000;
    IODIR1 = 0x00ff0000;
    IOCLR0 = 0x00580000;
    IOSET0 = 0x00180000;
}
void __gccmain()
{

}
```

RESULT:

EXP NO:	**INTERFACING ADC & DAC**
DATE:	

AIM:

To Write and Execute a program for reading an on-chip ADC, convert it into decimal and to display it and to generate a buzzer using DAC interfacing.

PROCEDURE

1. Create a New project, Go to “Project” and close the current project “Close Project”.
2. Next Go to the Project New µvision Project Create New Project Select Device for Target.
3. Select the data base NXP LPC2148.
4. Add Startup file and Next go to “File” and click “New”.
5. Write a program on the editor window and save as “Main.c”.
6. Add this source file to Group and click on “Build Target” or F7.
7. Create a Hex file from “Project” menu and click on “Rebuild all target Files”.
8. Open Flash magic and select the device LPC2148 in ARM 7 category, COM port will be COM 3, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.
9. Next browse the path of hex file and select the file.
10. After selecting ISP mode on the Hardware Kit and click on start then device will start to program

PROGRAM:

```
/***************************************************/
/* Read ADC channel 2 and  display it on LCD Display*/
/***************************************************/
#include "LPC214x.h"                    /* LPC21xx definitions  */
int ReadADC(char ChannelNumber);
void WriteCommandLCD(unsigned char CommandByte);
void WriteDataLCD(unsigned char DataByte);
void LCDDelay(void);
void LCDDelay1600(void);
void SendByte(unsigned char Value);
void InitializeLCD(void);
void DataAddressDirection(void);
void DisplayLCD(char LineNumber,char *Message);
void DisplayLCD2Digit(char LineNumber,char CharPosition,char Data);
int main(void)
{
   int a;
         unsigned char  Channel = 2;
         PINSEL1 = 0x04000000;                    // Select ADC to pin P0.29
         InitializeLCD();                         // Initialize LCD
         DisplayLCD(0,"    ADC DEMO    ");         // Display message
         DisplayLCD(1,"Channel 2:      ");         // Display message
         while(1)
         {
                  a=ReadADC(Channel);                      // Read ADC channel 2
                  DisplayLCD2Digit(1,10, (a >> 8));   // Display it on 2nd line of LCD
                  DisplayLCD2Digit(1,12, (a & 0xff));
                  LCDDelay1600();
         }
```

```
}
//Read ADC data from given channel number
int ReadADC(char ChannelNumber)
{
        int val,ch;
        ch = 1<<ChannelNumber;
        AD0CR   = 0x00210400 | ch;                // Setup A/D: 10-bit AIN @ 3MHz
        AD0CR  |= 0x01000000;                     // Start A/D Conversion
        do
        {
           val = AD0DR2;                        // Read A/D Data Register
        }
        while ((val & 0x80000000) == 0);          // Wait for the conversion to complete
        val = ((val >> 6) & 0x03FF);              // Extract the A/D result
        AD0CR  &= ~0x01000000;                   // Stop A/D Conversion
        return(val);                                       // Return the Data Read
}
void InitializeLCD()
{
        DataAddressDirection();
        IOSET0 = 0x00580000;                      // Set A0, A1, A2
        WriteCommandLCD(0x38);                    //Command to select 8 bit interface
        LCDDelay1600();
        WriteCommandLCD(0x38);                    //Command to select 8 bit interface
        LCDDelay();                                        //Small delay
        WriteCommandLCD(0x38);                    //Command to select 8 bit interface
        LCDDelay();
        WriteCommandLCD(0x0c);                    //Command to on cursor,blink cursor
        LCDDelay();
        WriteCommandLCD(0x06);                    //Command for setting entry mode
        LCDDelay();
        WriteCommandLCD(0x01);              //Clear LCD
        LCDDelay1600();
}
/* Writes a command byte to LCD */
void WriteCommandLCD(unsigned char CommandByte)
{
        IOCLR1 = 0x03000000;                // Clear RS and RW
        SendByte(CommandByte);
        LCDDelay();                         //Small delay
}
/* Send a byte of data to LCD */
void SendByte(unsigned char Value)
{
        IOPIN1 &= 0xff00ffff;
        IOPIN1 |= Value << 16;          /* Write data to data bus */
        IOSET0 = 0x00100000;            /* Generate chip enable signal for LCD */
        IOCLR0 = 0x00480000;
        LCDDelay();
        IOSET0 = 0x00580000;            /* Set A0, A1 & A2 to disable LCD */
        LCDDelay();
}
/* Writes a Data byte to LCD */
void WriteDataLCD(unsigned char DataByte)
{
        IOCLR1 = 0x01000000;                /* clear RW */
        IOSET1 = 0x02000000;                /* Set RS */
        SendByte(DataByte);
        LCDDelay();                                 //Small delay
}
```

```
/* Small delay */
void LCDDelay(void)
{
        int     a;
        for(a=0;a<0x1000;a++);
}
/* Big delay */
void LCDDelay1600(void)
{
        long    a;
        for(a=0;a<0x050000;a++);
}
/* Makes cursor visible */
void CursorON(void)
{
        WriteCommandLCD(0x0f);                  //Command to switch on cursor
}
/* Makes cursor invisible */
void CursorOFF(void)
{
        WriteCommandLCD(0x0c);                  //Command to switch off cursor
}
void DisplayLCD2Digit(char LineNumber,char CharPosition,char Data)
{
        unsigned char a;
        if(LineNumber ==0)
        {       //First Line
                a = 0x80;                               //command for first line select

        }
        else
```

```
    {       //Second line
            a = 0xc0;                               //command for second line selection
    }
    a+=(CharPosition);                              //Calculate the character position
    WriteCommandLCD(a);                             //Send command to select the given digit
    if( (Data & 0xf0) < 0xa0)                       //Check for less than 0xa0
    {
            a = ((Data & 0xf0) >> 4) + '0';         //Get the ASCII character
    }
    else
    {
            a = ((Data & 0xf0) >> 4) + 'A'- 0x0a; //Get the ASCII character
    }
    WriteDataLCD(a);                        //Display the first character
    if( (Data & 0x0f) < 0x0a)               //Check for less tham 0x0a
    {
            a = (Data & 0x0f)+'0';          //Get the ASCII character
    }
    else
    {
            a = (Data & 0x0f)+'A' - 0x0a;//Get the ASCII character
    }
    WriteDataLCD(a);
}
/* Displays a message on LCD */
void DisplayLCD(char LineNumber,char *Message)
{
    //int   a;
    if(LineNumber ==0)
    {       //First Line
```

```
                WriteCommandLCD(0x80);          //Select the first line
        }
        else
        {       //Second line
                WriteCommandLCD(0xc0);          //Select the second line
        }
        while(*Message)
        {
                WriteDataLCD(*Message);         //Display a character
                Message++;                      //Increment pointer
        }
}
void DataAddressDirection(void)
{
        IODIR0 |= 0x00580000;                   // Set A0, A1, A2 output lines
        IODIR1 |= 0x03ff0000;
}

void __gccmain()
{


}
```

RESULT:

DAC:

PROGRAM:

```
/**************************************************/
/* Program to DAC */
/**************************************************/
#include "LPC214X.H"
void InitializeDAC(void);
int main (void)
{
        long c;
        InitializeDAC();                                        //Initialize DAC
        while(1)
        {
                DACR = 0x00;                                    // Set DAC = 0
                for(c=0;c<0xf0000;c++);                         // Delay
                DACR = 0x0000ffc0;                              // Set DAC = (0x3ff << 6)
                for(c=0;c<0xf0000;c++);                         // Delay
        }
}
void InitializeDAC(void)
{
        PINSEL1 = 0x00080000;                                   // Set P0.25 for DAC output
}
void __gccmain()
{

}
```

RESULT