1.a)

PROGRAM:

Label	Address	Mnemonics		Hex code	comments
		opcode	operand		
START:	8100	CLR	C	C3	clear CY Flag
	8101	MOV	A, #0A	740A	Get the data1 in Accumulator
	8103	ADDC	A, #10	3410	Add the data1 with data 2
	8105	MOV	DPTR, #8500	908500	Initialize the memory location store the result in memory location
L1	8109	SJMP	L1	80FE	stop the program

1.b)

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 data 1 in accumulator
	8103	SUBB	A, #05	9405	Subtract data 2 from data 1
	8105	MOV	DPTR, #4500	90 85 00	Initialize memory location
	8108	MOV X	@DPTR, A	F0	Store the difference in memory location
L1	8109	SJMP	L1	80FE	stop the program

1-C)

PROGRAM:

Label	Address	Mnemonics		Hex code	Comments
		opcode	operand		
START:	8100	MOV	A, #05	74 05	store data in accumulator
	8102	MOV	B, #03	75F003	store data 2 in B register
	8105	MUL	AB	A4	multiply both
	8106	MOV	DPTR, #4500	904500	Initialize memory location
	8109	MOV X	@DPTR, A	F0	store lower order result
	810A	INC	DPTR	A3	Go to next memory location
	810B	MOV X	A, B	E5F0	store higher order result.
	810D	MOV X	@DPTR, A	F0	
L1	810E	SJMP	L1	80FE	stop the program

1.d)

PROGRAM:

Label	Address	Mnemonics		Hex code	Comments
		opcode	operand		
START:	8100	MOV	A, #15	7415	store data1 in accumulator
	8102	MOV	B, #03	75F003	Store data2 in B register
	8105	DIV	AB	84	Divide
	8106	MOV	DPTR, #4500	908500	Initialize memory location
	8109	MOVX	@DPTR, A	F0	Store remainder
	810A	INC	DPTR	A3	Go to next memory location
	810B	MOV	A, B	E5F0	Store quotient
	810D	MOVX	@DPTR, A	F0	
L1	810E	SJMP	L1	80FE	stop the program

PROGRAM:

```
MOV  R0, #51H
MOV  R1, #61H
MOV  R2, #02H
CLR C
BACK : MOV A, @R0
ADDC  A, @R1
MOV @R1,A
DEC R0
DEC R1
DJNZ R2, BACK
```

1.e

```
JNC   FINISH
MOV @R1, #01H
FINISH : SJMP $
END
```

1. ℓ

PROGRAM:

```
MOV R0, #51H
MOV R1, #61H
MOV R2, #02H
CLR C
BACK : MOV A ,@R0
SUBB A, @R1
MOV @R1, A
DEC R0
DEC R1
DJNZ R2, BACK
JNZ POSITIVE
MOV @R1, #0FFH
```

1.f

```
    JMP : FINISH
POSITIVE : MOV @R1, #00H
FINISH : SJMP $
END
```

1. f

2.

PROGRAM :-

Address	Mnemonics
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

2.

8051 TRAINER KIT PROGRAM :-

Label	Address	Mnemonics		Hex code
		opcode	operand	
FLASH:	8100	CPL	P1.4	B294
	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	85 02 F1
	8E68	POP	02	D0 02
		POP	01	D0 01
		POP	00	D0 00
		RET		

RESULT :-

3.

PROGRAM :

Label	Address	Mnemonics	
		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 moving switch out / P1.2 until it becomes high

b) When P0.1 becomes high, light LEDS connected at port 2

```
SETB P1.2 ;
MOV A, #FFH ;
AGAIN : JNB P1.2, AGAIN :
MOV P1.5, A ;
SJMP
```

Make P1.2 as input (switch)
A = 11111111
get out when P1.2 = 1
light LEDS by sending to P2.

PROGRAM :

```
8650 : LCALL 866E

STAT SERIAL : MOV DPTR, #0000
              LCALL 1080

WAIT FOR KEY : LCALL 1070
               ANL A, #07
               JZ 8659
               LCALL 1080
               LCALL 8682
               LCALL 867A
               LCALL 1050
               SJMP 8655

INI - SERIAL PORT:
               MOV SCON, # 52H
               MOV TMOD, # 20H
               MOV TH1, #80H
               SETB TR1
               RET

RECEIVE BYTE :
               JNB RI, $
               MOV A, SBUF
               CLR, RI
               RET

JNBRI, 867A
TRANSMIT BYTE :
               MOV SBUF, A
               JNB TI, $
               CLR TI
               RET
JNB TI, 8684
```

RESULT :

Thus, the assembly language Programming using Serial Ports in 8051 has been verified.

4

03/05/24

PROGRAM :

```
#Include <LPC 214X.H>
void Delay (int)
int main (void)
{
  IODIRO = 0X05800000;
  IODIRI = 0X00FF0000;
  while (1)
  {
    IOCLRI = 0X00030000;
    IOCLRO = 0X00580000;
    IOSETO = 0X00580000;
    Delay(10);
    IOSETI = 0X00030000;
    IOCLRO = 0X00580000;
    IOSETO = 0X00580000;
    Delay(10);
  }
}
void Delay (int n)
{
   int p, q;
   for (p=0; p<n; p++)
   {
     for (q=0; q<0x99900; q++);
   }
}
```

5

PROGRAM 2:

```
#include <nxp/iolpc2148.H>
void delay()
{
for (int i = 0x00; i<= 0xff; i++);
for (int j = 0x00; j<= 0xFf; j++);
}
void main()
{
    PINSEL2 =0x00000000;
    IO1DIR = 0xFF000000;
     while (1)
     {
      IO1SET= 0xFF000000;
      delay();
      IO1CLR = 0xFF000000;
       delay();
      }
}
```

5

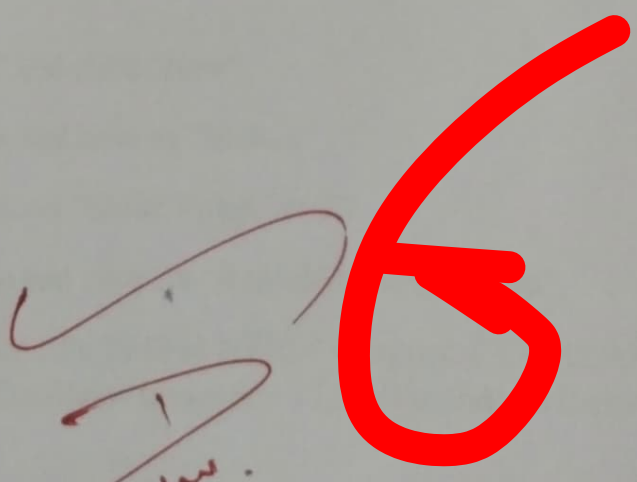
RESULT :

Thus, the program for LED & flashing LED has been executed and verified successfully.

PROGRAM :

```
# include "LPC214X.h"
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, charData);

int main(void)
{
    InitializeLCD();
    DisplayLCD(0, "NXP2148 ARM");
    DisplayLCD(1, "Evaluation System");
    while(1);
}
void _gcc main()
{
}
```



PROGRAM :

```
#include "LPC214X.H"
irq void T0_Srv(void);
void Initialize TIMER0(void);
void Initialize KeyBoard(void);
void ScanKeyboard(void);
unsigned char Read Column(void);
void SetRow(unsigned char data);
void Display LCD(char LineNumber, char* Message);
Char KeyboardFlag;
unsigned char keyboard code = 0;
char Counter = 0, ReleaseFlag = 0, StatusFlag = 0,
                    IdentificationFlag = 0;

void WriteCommandLCD(unsigned char Command
                                    Byte);
void WriteDataLCD(unsigned char Data Byte);
void LCDDelay(void);
void LCDDelay1600(void);
Void SendByte(unsigned char value);
void InitializeLCD(void);
void Data Address Direction(void);
void DisplayLCD(char Line Number, char* Message);

void DisplayLCD2 Digit(char LineNumber, char
                    Position, charData);
```

7.

```
int main (void)
{
  KeyboardFlag = 0;
  InitializedLCD();
  IODIR0 |= 0X00078000;
  Initialize Timer R0();
  DisplayLCD(0, "Matrix Keyboard");
  DisplayLCD(1, "KeyPressed: ");
  While(1)
  {
      if (KeyboardFlag == 1)
      {
        DisplayLCD2Digit (1,12, Keyboard code);
        KeyboardFlag = 0;
      }
  }
}
void ScanKeyboard (void)
{
  unsigned char a,b,c;
  unsigned char f;
  if (Release Flag == 1)
  {
    counter ++;
    If (counter == 10)
    {
      Release Flag = 0;
    }
```

7

```
        return
}
if (StatusFlag == 1)
{
    if (IdentificationFlag == 1)
    {
      SetRow(0x00);
      if (Read Column() = 0x0f)
      {
        Counter = 0;
        Release Flag = 1;
        Identification Flag = 0;
        Status Flag = 0;
      }
      return;
    }
    else
    {
    counter++;
     if (counter == 10)
     {
       Keyboard code = 0;
       SetRow(0x00);
       if (Read Column() != 0x0f)
       {
          for (a = 0; a < 4; a++)
          {
             f = ~(0x01 << a);
             SetRow(f);
             b = Read Column();
```

7

```
                for (c=0; c<4; c++)
                {
                  f = 0x01<<c;
                  if((b & f)==0)
                  {
                    IdentificationFlag=1;
                    StatusFlag=1;
                    KeyboardFlag=1;
                    Keyboard code & = 0x0f;
                    return;
                  }
                  Keyboard code ++;
                }
              }
            }
            IdentificationFlag=0;
            StatusFlag=0;
            return
          }
          else
          {
            return;
          }
        }
}
SetRow(0x00);
if(ReadColumn()!=0x0f)
{
    counter=0;
```

```
        StatusFlag=1;
        IdentificationFlag=0;
      }
}
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;

}
unsigned char ReadColumn(void)

{   unsigned char  a=0;

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

7

```
        return (a);
}
void Initialize TIMER0 (void)
{
        VPBDIV = 0x00000002;
        T0PR   = 0x0000012B;
        T0TCR = 0x00000002;
        T0MCR = 0x00000003;
        T0MR0 = 0x00000064;
        T0TCR  = 0x00000001;
        VICVectAddr4 = (unsigned) T0_ISR;
        VICVectCntl4 = 0x00000024;
        VICIntEnable|= 0x00000010;
}
__irq void T0_ISR (void)
{
    scankeyboard ();
keyboard status
        T0IR |= 0x00000001;
        VICVectAddr = 0x00000000;
{
```

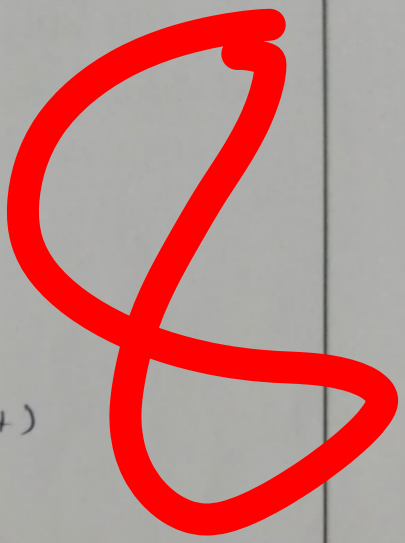
7

RESULT :

Thus, the program for INTERFACING OF MATRIX KEYBOARD has been executed and verified successfully.

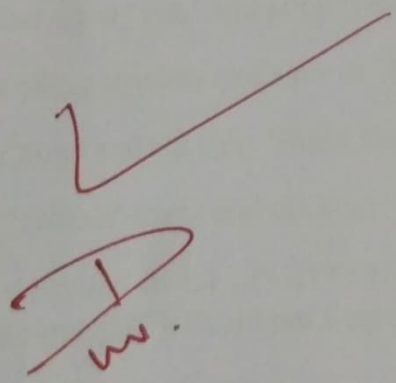
PROGRAM:

```
# include "LPC214X.H"
void InitializeIO(void);
int main(void)
{
    unsigned char a;
    unsigned int b;
    long c, d;
    InitializeIO();
    while(1)
    {
        for(d=0; d<200; d++)
        {
          b=0x2222;
          for(a=0; a<4; a++)
          {
        IOPIN1=(b&0x7f00)<<8;
                IOCLR0=0x00400000;
                IOSET0=0x00400000;
                for(c=0; c<0xa000; c++);
                b=b<<1;
          }
        }
        for(d=0; d<200; d++)
        {
          b=0x8888;
```



```
            for (a=0; a<4; a++)
            {
        IOPIN1 = (b & 0x ff00) << 8;
            IOCLR0 = 0x00400000;
            IOSET0 = 0x00400000;
            for (c=0; c<0xa000; c++);
            b = b >> 1;
            }
        }
    }
}
void InitializeIo (void)
{
    IODIR0 = 0x00580000;
    IODIR1 = 0x00ff0000;
    IOCLR0 = 0x00580000;
    IOSET0 = 0x00180000;
}
void _gccmain()
{


}
```



PROGRAM :

```
#include "LPC214X.h"
int ReadADC (char charNumber);
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, charData);

int main(void)
{
int a;
    unsigned char channel=2;
    PINSEL1 = 0x04000000;
    InitializeLCD();
    DisplayLCD(0, "ADC DEMO");
    DisplayLCD(1, "channel1:");
    while(1)
    {
      a= ReadADC(channel);
      DisplayLCD2Digit(1,10,(a>>B));
      DisplayLCD2Digit(1,12,(a&0x1f));
      LCDDelay1600();
```

9

```
}
int ReadADC (char ChannelNumber)
{int
        int val, ch;
        ch = 1 << channel Number;
        ADOCR = 0x00210400 | ch;
        ADOCR1 = 0x01000000;
        do
        {
            val = ADODR2;
        }
        while ((val & 0x80000000) == 0)
        val = ((val >> 6) & 0x03FF);
        ADOCR &= ~0x01000000;
        return (val);
}
void Initialize LCD()
{
        Data Address Direction ();
        IOSET0 = 0x00580000;
        Write Command LCD (0x38);
        LCD Delay 1600();
        Write Command LCD (0x38);
        LCD Delay ();
        Write command LCD (0x38);
        LCD Delay();
        Write command LCD (0x0C);
        LCD Delay();
        Write Command LCD (0x06);
        LCD Delay();
```

```
        WriteCommandLCD(0x01);
        LCDDelay1600();
}
void WriteCommandLCD(unsigned char CommonByte);
{
        IOCLR1 = 0x03000000;
        SendByte(commandByte);
        LCDDelay();
}
void SendByte(unsigned charvalue)
{
        IOPIN1 & = 0xff00ffff;
        IOPIN1|= value << 16;
        IOSET0 = 0x00100000;
        IOCLR0 = 0x00480000;
        LCDDelay();
        IOSET0 = 0x00580000;
        LCD Delay();
        IOSET0 = 0x00580000;
        LCD Delay();
}
void WriteDataLCD(unsigned char DataByte)
{
        IOCLR1 = 0x01000000;
        IOSET1 = 0x02000000;
        SendByte(DataByte);
        LCDDelay();
}
```

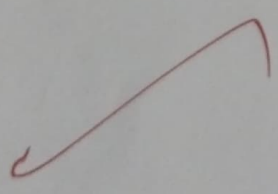
9

```
void LCDDelay (void)
{
    int a;
    for (a=0; a<0x1000; a++);
}
void LCDDelay 1600 (void)
{
    long a;
    for (a=0; a<0x050000; a++);
}
void cursor ON (void)
{
    Write Command LCD (0x01);
}
void cursor OFF (void)
{
    Write Command LCD (0x0C);
}
void Display LCD Digit (char LineNumber, char
                        charPosition, charData);
{
    unsigned char a;
    if (LineNumber == 0)
    {
        a = 0x80;
    }
    else
```

9

```
{
  a = 0x80;
}
a += (char Position);
WriteCommandLCD(a);
If (Data & 0xf0) < 0xa0)
{
    a=((Data & 0xf0)>>4)+'0';
}
else
{
    a=((Data & 0xf0)>>4)+'A'-0x0a;
}
WriteDataLCD(a);
if ((Data & 0x0f) < 0x0a)
{
    a = (Data & 0x0f)+'0';
}
else
{
    a= (Data & 0x0f) + 'A'-0x0a;
}
WriteDataLCD(a);

}
void DisplayLCD (char LineNumber, char *Message)
{
    If (LineNumber ==0)
    {
```



```
        WriteCommandLCD (0x80);
    }
    else
    {
        WriteCommandLCD (0xC0);
    }
    while (*Message)
    {
        WriteDataLCD (*Message);
        Message++;
    }
}
void DataAddressDirection (void)
{
    IODIR0 |= 0x00580000;
    IODIR1 |= 0x03ff0000;
}
void __gccmain()
{
}
```

9

DAC :

PROGRAM :

```
# include "LPC 214X.H"
void Initialize DAC (void);
int main (void)
{
    long cc ;
    Initialize DAC ();
    while (1)
    {
        DAC R =0X00 ;
        for (c=0; c<0xf0000 ; c++);
        DAC R =0X0000ff C0;
        for(c=0 ; c<0xf0000 ; c++);
    }
}
void Initialize DAC (void)
{
        PINSEL 1 = 0X00080000;
}
void_gcc main ()
{
}
```

9

31/05/24

Program:

```
        ORG 0H
        MOV TL0, #0CH;
        MOV TH0, #0H;
        MOV TL1, #4FH;
        MOV TH1, #3CH;
        MOV TR0, #3CH;
        MOV TR1, #1
MAIN
        ACALL Display
        SJMP MAIN;
DISPLAY
        MOV A, TL0;
        MOV R2; #10;
        DIVAB
        MOV R0, A;
        MOV A, B;
        MOV R1, A;
MOV 12, #0FFH;
MOV A, R0;
ADD A, #30H;
MOV P2, A;
CALL DELAY;
MOV P2 #0FFH;
Display minute
MOV A, R1;
```

10

```
ADD A, #30H;
MOV P2, A;
CALL DELAY;
MOV P2, #0FFH;
Display Seconds
MOV A, TL1
MOV B, TH1
MOV R2 #10;
DIV AB;
MOV R0, A;
MOV A, B;
MOV R1, R0;
ADD A, #30H;
MOV P2 - A;
A CALL DELAY
MOV P2, A;
CALL DELAY;
RET'
DELAY:
        MOV R5, # 250
OUTER - LOOP
        MOV R4, # 250
INNER - LOOP
        DJNZ R4, INNER - LOOP;
RET
END
```

10