LABORATORY
[As per Choice Based Credit System (CBCS) scheme]

(Effective from the academic year 2022 -2023))

SEMESTER - VI
Laboratory Code 21CS43

CIE Marks 50 **SEE Marks 50**

Course objectives: This course (18CSL48) will enable students to:

- Develop and test Program using ARM7TDMI/LPC2148
- Conduct the experiments on an ARM7TDMI/LPC2148 evaluation board using evaluation version of Embedded 'C' & Keil Uvision-4 tool/compiler

Laboratory Experiments: PART -A

1. Write a program to multiply two 16 bit binary numbers.
2. Write a program to find the sum of first 10 integer numbers.
3. Write a program to find factorial of a number.
4. Write a program to add an array of 16 bit numbers and store the 32 bit result in internal RAM
5. Write a program to find the square of a number (1 to 10) using look-up table.
6. Write a program to find the largest/smallest number in an array of 32 numbers.
7. Write a program to arrange a series of 32 bit numbers in ascending/descending order
8. Write a program to count the number of ones and zeros in two consecutive memory locations.

Laboratory Experiments: PART –B

Conduct the following experiments on an ARM7TDMI/LPC2148 evaluation board using evaluation version of Embedded 'C' & Keil Uvision-4 tool/compiler.

9. Display “Hello World” message using Internal UART.
10. Interface and Control a DC Motor.
11. Interface a Stepper motor and rotate it in clockwise and anti-clockwise direction.
12. Determine Digital output for a given Analog input using Internal ADC of ARM controller.

13. Interface a DAC and generate Triangular and Square waveforms.
14. Interface a 4x4 keyboard and display the key code on an LCD.
15. Demonstrate the use of an external interrupt to toggle an LED On/Off.
16. Display the Hex digits 0 to F on a 7-segment LED interface, with an appropriate delay in between

Course outcomes:

- On the completion of this laboratory course, the students will be able to:
- Develop and test program using ARM7TDMI/LPC2148.
- Understand the working and implementation of ALU.

Graduate Attributes (as per NBA)

1. Engineering Knowledge
2. Problem Analysis
2. Design/Development of Solutions
3. Modern Tool Usage

Conduction of Practical Examination:

1. All laboratory experiments (1 to 11 nos) are to be included for practical examination.

2. Students are allowed to pick one experiment from the lot.

3. Strictly follow the instructions as printed on the cover page of answer script.

4. Marks distribution:

For laboratories having only one part – Procedure + Execution + Viva-Voce: 15+70+15 = 100 Marks

For laboratories having PART A and PART B

i. Part A = Procedure + Execution + Viva = 6 + 28 + 6 = 40 Marks

ii. Part B-Procedure + Execution + Viva = 9 + 42 + 9 = 60 Marks

PART-A

Experiment no 1

Write a program to multiply two 16 bit binary numbers.

```
;/* PROGRAM TO MULTIPLY TWO 16BIT NUMBERS  */

;/*     VALUE1:     1900H (6400)          (IN R1)*/

;/*     VALUE2:     0C80H (3200)          (IN R2)*/

;/*     RESULT:     1388000H(20480000)(IN R3)*/

;/* SET A BREAKPOINT AT NOP INSTRUCTION,RUN THE PROGRAM & CHECK THE
RESULT      */

   AREA  MULTIPLY , CODE, READONLY

ENTRY                          ;Mark first instruction to execute

START

      MOV r1,#6400             ; STORE FIRST NUMBER IN R0

      MOV r2,#3200             ; STORE SECOND NUMBER IN R1

      MUL r3,r1,r2             ; MULTIPLICATION

      NOP

      NOP

      NOP

      END                       ;Mark end of file
```

Results:

Experiment no 2

Write a program to find the sum of first 10 integer numbers

```
AREA SUM, CODE, READONLY
ENTRY
START
                MOV R5, #10
                MOV R0, #0
                MOV R1, #1
LOOP    ADD R0, R0, R1
                ADD R1, R1, 1
                SUBS R5, R5, #1
                CMP R5, #0
                BNE LOOP
                LDR R4, =RESULT
                STR R0, [R4]

XSS     B XSS

AREA DATA2, DATA, READWRITE
RESULT DCD 0X0
END
```

Experiment no 3

Write a program to find factorial of a number

```
AREA  FACTORIAL , CODE, READONLY

ENTRY                                  ;Mark first instruction to execute

START

            MOV r0, #7                 ; STORE FACTORIAL NUMBER IN R0

            MOV r1,r0                  ; MOVE THE SAME NUMBER IN R1

FACT  SUBS  r1, r1, #1                 ; SUBTRACTION

            CMP r1, #1                 ; COMPARISON

            BEQ STOP

            MUL r3,r0,r1;              ; MULTIPLICATION

            MOV   r0,r3                ; Result

            BNE FACT                   ; BRANCH TO THE LOOP IF NOT EQUAL

STOP

            NOP

            NOP

            NOP

            END                        ;Mark end of file
```

Experiment no 4

Write a program to add an array of 16 bit numbers and store the 32 bit result in internal RAM

```
  ;/* Program To Add An Array Of 16bit Numbers & Store In Internal Ram*/

;/*Array Of 6 Numbers          0x1111,0x2222,0x3333,0xaaaa,0xbbbb,0xcccc*/

;/* The Sum Is 29997h The Result Can Be Viewed In Location 0x40000000 & Also In R0*/

;/* Set A Breakpoint At Nop Instruction,Run The Program & Check The Result     */

        AREA  ADDITION , CODE, READONLY

ENTRY                                                   ;Mark first instruction to execute

START

        MOV R5,#6                       ; INTIALISE COUNTER TO 6(i.e. N=6)

        MOV R0,#0                       ; INTIALISE SUM TO ZERO

        LDR R1,=VALUE1                  ; LOADS THE ADDRESS OF FIRST VALUE

LOOP

        LDR R2,[R1],#2                  ; WORD ALIGN T0 ARRAY ELEMENT

        LDR R3,MASK                     ; MASK TO GET 16 BIT

        AND R2,R2,R3                    ; MASK MSB

        ADD R0,R0,R2                    ; ADD THE ELEMENTS

        SUBS R5,R5,#1                   ; DECREMENT COUNTER

        CMP R5,#0                       ;

        BNE LOOP                        ;  LOOK BACK TILL ARRAY ENDS

        LDR R4,=RESULT                  ; LOADS THE ADDRESS OF RESULT

        STR R0,[R4]                     ; STORES THE RESULT IN R1

        NOP

        NOP

        NOP
```

```
MASK DCD 0X0000FFFF                    ;  MASK MSB

VALUE1 DCW      0X1111,0X2222,0X3333,0XAAAA,0XBBBB,0XCCCC     ; array of    16 bit numbers(n=6)

        AREA DATA2,DATA,READWRITE                  ; to store result in given   address
        RESULT DCD 0X0

        END                            ; Mark end of file
```

Results

Experiment no 5

Write a program to find the square of a number (1 to 10) using look-up table.

```
;/* Assembly Program To Find Square Of Number   */

;/* Given Number Is 6 (R1) Then Result Is In R3=24h(36)   */

;/* Set A Breakpoint At Nop Instruction,Run The Program & Check The Result     */


        AREA  SQUARE , CODE, READONLY

ENTRY                               ;Mark first instruction to execute

START

        LDR    R0, = TABLE1         ; Load start address of Lookup table

        LDR R1,= 6                  ; Load no whose square is to be find

        MOV R1, R1, LSL#0x2         ; Generate address corresponding to square of given no

        ADD R0, R0, R1              ; Load address of element in Lookup table

        LDR R3, [R0]                ; Get square of given no in R3

        NOP

        NOP

        NOP

        ;Lookup table contains Squares of nos from 0 to 10 (in hex)

TABLE1      DCD 0X00000000;         SQUARE OF 0=0

            DCD 0X00000001;         SQUARE OF 1=1

            DCD 0X00000004;         SQUARE OF 2=4

            DCD 0X00000009;         SQUARE OF 3=9

            DCD 0X00000010;         SQUARE OF 4=16

            DCD 0X00000019;         SQUARE OF 5=25

            DCD 0X00000024;         SQUARE OF 6=36

            DCD 0X00000031;         SQUARE OF 7=49
```

```
          DCD 0X00000040;          SQUARE OF 8=64
          DCD 0X00000051;          SQUARE OF 9=81
          DCD 0X00000064;          SQUARE OF 10=100
     END                        ; Mark end of file
```

Results

Experiment no 6

Write a program to find the largest/smallest number in an array of 32 numbers .

```
;/* program to find largest number in an array & store in internal ram*/

;/*array of 7 numbers 0x44444444 ,0x22222222,0x11111111,0x33333333,0xaaaaaaaa*/

;/*0x88888888 ,0x99999999*/

;/* result can be viewed in location 0x40000000 & also in r2           */

;/* set a breakpoint at nop instruction, run the program & check the result   */

        AREA   LARGEST , CODE, READONLY

ENTRY                               ;Mark first instruction to execute

START

        MOV R5,#6                   ; intialise counter to 6(i.e. n=7)

        LDR R1,=VALUE1              ; loads the address of first value

        LDR R2,[R1],#4              ; Word Align T0 Array Element

LOOP

        LDR R4,[R1],#4              ; Word Align T0 Array Element

        CMP R2,R4                   ; Compare Numbers

        BHI LOOP1                   ; If The First Number Is > Then Goto Loop1

        MOV R2,R4                   ; If The First Number Is < Then Mov Content R4 TO R2

LOOP1

        SUBS R5,R5,#1               ; Decrement Counter

        CMP R5,#0                   ; Compare Counter To 0

        BNE LOOP                    ; Loop Back Till Array Ends


        LDR R4,=RESULT              ; Loads The Address Of Result

        STR R2,[R4]                 ; Stores The Result In R1

        NOP
        NOP
        NOP
; ARRAY OF 32 BIT NUMBERS(N=7)
VALUE1
                DCD   0X44444444              ;
                DCD    0X22222222             ;
                DCD    0X11111111             ;
                DCD    0X33333333             ;
                DCD    0XAAAAAAAA                 ;
                DCD    0X88888888             ;
                DCD    0X99999999             ;
        AREA DATA2,DATA,READWRITE     ; To Store Result In Given Address

RESULT DCD 0X0
        END                   ; Mark end of file
```

```
;/* Program To Find Smallest Number In An Array & Store In Internal Ram          */
;/*Array Of 7 Numbers 0x44444444 ,0x22222222,0x11111111,0x22222222,0xaaaaaaaa*/

;/*0x88888888 ,0x99999999  */
;/* Result Can Be Viewed In Location 0x40000000 & Also In R2              */
;/* Set A Breakpoint At Nop Instruction,Run The Program & Check The Result     */
        AREA  SMALLEST , CODE, READONLY
ENTRY                           ;Mark first instruction to execute
START
        MOV R5,#6               ; INTIALISE COUNTER TO 6(I.E. N=7)
        LDR R1,=VALUE1          ; Loads The Address Of First Value
        LDR R2,[R1],#4          ; Word Align T0 Array Element
LOOP
        LDR R4,[R1],#4          ; Word Align T0 Array Element
        CMP R2,R4               ; Compare Numbers
        BLS LOOP1               ; If The First Number Is < Then Goto LOOP1
        MOV R2,R4               ; If The First Number Is > Then Mov Content R4 To R2
LOOP1
        SUBS R5,R5,#1                   ; Decrement Counter
        CMP R5,#0                       ; Compare Counter To 0
        BNE LOOP                        ; Loop Back Till Array Ends
        LDR R4,=RESULT                  ; Loads The Address Of Result
        STR R2,[R4]                     ; Stores The Result In R1
        NOP
        NOP
        NOP
```

```
; ARRAY OF 32 BIT NUMBERS(N=7)

VALUE1

                DCD 0X44444444                  ;

                DCD    0X22222222               ;

                DCD    0X11111111               ;

                DCD    0X22222222               ;

                DCD    0XAAAAAAAA                       ;

                DCD    0X88888888               ;

                DCD    0X99999999               ;

        AREA DATA2,DATA,READWRITE               ; TO STORE RESULT IN GIVEN
ADDRESS

RESULT DCD 0X0

        END                             ; Mark end of file
```

Results

	11111								

22222	**11111**	**222222**	**Aaaaaa**	**888888**	**999999**				

				11	**11**	**11**	**11**	

Experiment no 7

Write a program to arrange a series of 32 bit numbers in ascending/descending order.

```
;/* Program To Sort In Ascending Order                                    */
;/*Array Of 4 Numbers 0x44444444 ,0x11111111,0x33333333,0x22222222        */
;/* Set A Breakpoint At Start1 Lable & Run The Program                    */
;/*Check The Unsorted Numbers At Location 0x40000000 Next                 */
;/* Set A Breakpoint At Nop Instruction,Run The Program & Check The Result / *
;/* Result Can Be Viewed At Location 0x40000000                           */
        AREA  ASCENDING , CODE, READONLY
ENTRY                           ;Mark first instruction to execute
START
            MOV R8,#4                   ; INTIALISE COUNTER TO 4(i.e. N=4)
            LDR R2,=CVALUE              ; ADDRESS OF CODE REGION
            LDR R3,=DVALUE              ; ADDRESS OF DATA REGION
LOOP0
            LDR R1,[R2],#4              ;  Loading Values From Code Region
            STR R1,[R3],#4              ;  STORING VALUES TO DATA REGION
            SUBS R8,R8,#1               ; DECREMENT COUNTER
            CMP R8,#0                   ; COMPARE COUNTER TO 0
            BNE LOOP0                   ; LOOP BACK TILL ARRAY ENDS
START1      MOV R5,#3                   ; INTIALISE COUNTER TO 3(i.e. N=4)
            MOV R7,#0                   ; Flag To Denote Exchange Has Occured
            LDR R1,=DVALUE              ; Loads The Address Of First Value
LOOP        LDR R2,[R1],#4              ; WORD ALIGN T0 ARRAY ELEMENT
```

```
            LDR R3,[R1]                    ; LOAD SECOND NUMBER
            CMP R2,R3                      ; COMPARE NUMBERS
            BLT LOOP2               ; If The First Number Is < Then Goto Loop2
            STR R2,[R1],#-4         ; INTERCHANGE NUMBER R2 & R3
            STR R3,[R1]             ; INTERCHANGE NUMBER R2 & R3
            MOV R7,#1               ; Flag Denoting Exchange Has Taken Place
            ADD R1,#4               ; RESTORE THE PTR
LOOP2       SUBS R5,R5,#1           ; DECREMENT COUNTER
            CMP R5,#0               ; COMPARE COUNTER TO 0
            BNE LOOP                ; LOOP BACK TILL ARRAY ENDS
            CMP R7,#0               ; COMPARING FLAG
            BNE  START1             ; If Flag Is Not Zero Then Go To Start1 Loop
            NOP
            NOP
            NOP
; ARRAY OF 32 BIT NUMBERS(N=4) IN CODE REGION
CVALUE
            DCD 0X44444444                 ;
            DCD   0X11111111               ;
            DCD   0X33333333               ;
            DCD   0X22222222               ;
      AREA DATA1,DATA,READWRITE            ; Array Of 32 Bit Numbers In Data Region
DVALUE
            DCD 0X00000000                 ;
            END                     ; Mark end of file
```

```
;/* Program To Sort In Descending Order                                    */
;/*Array Of 4 Numbers 0x44444444 ,0x11111111,0x33333333,0x22222222          */
;/* Set A Breakpoint At Start1 Lable & Run The Program                     */
;/*Check The Unsorted Numbers At Location 0x40000000 Next                  */
;/* Set A Breakpoint At Nop Instruction,Run The Program & Check The Result / *
;/* Result Can Be Viewed At Location 0x40000000                            */

        AREA  ASCENDING , CODE, READONLY
ENTRY                           ;Mark first instruction to execute
START
        MOV R8,#4               ; INTIALISE COUNTER TO 4(I.E. N=4)
        LDR R2,=CVALUE          ; Address Of Code Region
        LDR R3,=DVALUE          ; Address Of Data Region
LOOP0
        LDR R1,[R2],#4          ;  Loading Values From Code Region
        STR R1,[R3],#4          ;  Storing Values To Data Region
        SUBS R8,R8,#1           ; Decrement Counter
        CMP R8,#0               ; Compare Counter To 0
        BNE LOOP0               ; Loop Back Till Array Ends
START1  MOV R5,#3               ; Intialise Counter To 3(I.E. N=4)
        MOV R7,#0               ; Flag To Denote Exchange Has Occured
        LDR R1,=DVALUE           ; Loads The Address Of First Value
LOOP    LDR R2,[R1],#4          ; Word Align T0 Array Element
        LDR R3,[R1]             ; Load Second Number
        CMP R2,R3               ; Compare Numbers
        BGT LOOP2               ; If The First Number Is > Then Goto Loop2
        STR R2,[R1],#-4         ; Interchange Number R2 & R3
```

```
                STR R3,[R1]                 ; Interchange Number R2 & R3

                MOV R7,#1                   ; Flag Denoting Exchange Has Taken Place

                ADD R1,#4                   ; Restore The Ptr

LOOP2           SUBS R5,R5,#1               ; Decrement Counter

                CMP R5,#0                   ; Compare Counter To 0

                BNE LOOP                    ; Loop Back Till Array Ends

                CMP R7,#0                   ; Comparing Flag

                BNE  START1                 ; If Flag Is Not Zero Then Go To Start1 Loop

                NOP

                NOP

                NOP

; ARRAY OF 32 BIT NUMBERS(N=4) IN CODE REGION

CVALUE

                DCD 0X44444444                      ;

                DCD   0X11111111                    ;

                DCD   0X33333333                    ;

                DCD   0X22222222                    ;

        AREA DATA1,DATA,READWRITE           ; Array Of 32 Bit Numbers In Data Region

DVALUE

                DCD 0X00000000                      ;

                END                         ; Mark end of file
```

Experiment no 8

Write a program to count the number of ones and zeros in two consecutive memory locations.

```
AREA  ONEZERO , CODE, READONLY

ENTRY                               ;Mark first instruction to execute

START

            MOV R2,#0               ; COUNTER FOR ONES

            MOV R3,#0               ; COUNTER FOR ZEROS

            MOV R7,#2               ; COUNTER TO GET TWO WORDS

            LDR R6,=VALUE           ; LOADS THE ADDRESS OF VALUE

LOOP        MOV R1,#32              ; 32 BITS COUNTER

            LDR R0,[R6],#4          ; GET THE 32 BIT VALUE

LOOP0       MOVS R0,R0,ROR #1       ; RIGHT SHIFT TO CHECK CARRY BIT (1's/0's)

            BHI ONES          ; If Carry Bit Is 1 Goto Ones Branch Otherwise Next

ZEROS

            ADD R3,R3,#1      ;If Carry Bit Is 0 Then Increment The Counter By 1(R3)

            B LOOP1           ; BRANCH TO LOOP1

ONES

            ADD R2,R2,#1      ; If Carry Bit Is 1 Then Increment The Counter By 1(R2)

LOOP1

            SUBS R1,R1,#1     ; COUNTER VALUE DECREMENTED BY 1

      BNE LOOP0               ; IF NOT EQUAL GOTO TO LOOP0 CHECKS 32BIT

      SUBS R7,R7,#1           ; COUNTER VALUE DECREMENTED BY 1

      CMP R7,#0                     ; COMPARE COUNTER R7 TO 0

      BNE LOOP                      ; IF NOT EQUAL GOTO TO LOOP

      XSS   B XSS
```

```
VALUE DCD 0X3,0X2;   TWO VALUES IN AN ARRAY

END                              ; Mark end of file
```

Part -B

Program 9:

Display “Hello World” message using Internal UART.

```
include <LPC21xx.H>            /* LPC21xx definitions */

#include "Serial.h"

void delay_ms(int count)

{

   int j=0,i=0;

   for(j=0;j<count;j++)

   {

      for(i=0;i<35;i++);

   }

}

int main (void)

{

   uart0_init();                                        // Initialize UART0

   delay_ms(100000);

   while (1)

   {

      uart0_puts ("\n\rHello World\n\r");

     delay_ms(1000000);

   }

}
```

Program 10:

Interface and Control a DC Motor.

```
//**********************************************************************
// FileName              : DC motor Programming using Port-1
// Microcontroller       : LPC2148
// Compiler              : Keil v-4
// Target Hardware       : ARM7 Development Board
// Description           :DC motor rotating clockwise and anticlockwise direction
// Pin Connection        P1.30 and P1.31 pins of Port-0 connected to L293D IC(DC motor driver)
//**********************************************************************


#include<lpc214x.h>                           // Header file for LPC2148
//*************** Function Declaration **************//
void delay(void);
void dc_clock(void);
void dc_A_clock(void);
//*************** END ofFunction Declaration **************//
//*************** MAIN Program **************//
int main()
{
        PINSEL2 = 0x00000000;          // P1.0 to P1. 31 configured as GPIO
        IODIR1 = 0xFFFF0000;    // P1.16 to P1. 31 configured as ouput port


        while(1)
        {
                dc_A_clock();                          // Function calling
                delay();delay();delay();
                dc_clock();
                delay();delay();delay();               // Function calling
        }
}
//*************** END of MAIN Program **************//
//*************** Delay Program **************//
void delay(void)                                              // Delay Sub program
{
        unsigned int i,j;
        for(i=0;i<1000;i++)
        for(j=0;j<1000;j++);
}
//*************** END of Delay Program **************//
//*************** DC motor Program **************//
void dc_A_clock(void)           // DC-motor anticlockwise rotation Sub program
{
        IOSET1 = 0x80000000;
        delay();
        IOCLR1 = 0x80000000;
        delay();
}
```

```
void dc_clock(void)                    // DC-motor anticlockwise rotation Sub program
{
        IOSET1 = 0x40000000;
        delay();
        IOCLR1 = 0x40000000;
        delay();
}
//*************** END of DC motor Program **************//
```

Program 11:

Interface a Stepper motor and rotate it in clockwise and anti-clockwise direction.

```
//***************************************************************************

// FileName             : Stepper Motor Programming using Port-0

// Microcontroller      : LPC2148

// Compiler             : Keil v-4

// Target Hardware      : ARM7 Development Board

// Description          :Stepper Motor Roatating Clockwise direction after certain delay roatating
Anticlockwise direction

// Pin Connection   :   P0.28 to P0.31 connected to Stepper motor driver(ULN2003/2803)

//***************************************************************************

#include<lpc214x.h>

// ***************** Function Decleration ***********//

void delay(void);

void stepper_clock(void);

void stepper_A_clock(void);

// ***************** END of Function Decleration ***********//
// ***************** MAIN Program ***********//

int main()

{

        PINSEL0 = 0x00000000;               // P0.0 to P0.15 configured as GPIO
        PINSEL1 = 0x00000000;               // P0.15 to P0.31 configured as GPIO

                IODIR0 = 0xFFFFFFFF;        // P0.0 to P0.31 configured as ouput port

        while(1)

        {
stepper_clock();        // stepper motor clock Fuction calling (P0.28 to P0.31 port pins connected
                        to stepper motor)

delay();delay();delay();                    // Delay fuction call
stepper_A_clock();              // stepper motor anti clock Fuction calling (P0.28 to P0.31 port
                                pins connected to stepper motor)

delay();delay();delay();    // Fuction call (P0.28 to P0.31 port pins connected to stepper motor)

        }
}

// ***************** END of MAIN program ***********//
// ***************** Function Definations ***********//
void delay(void)                                        // delay sub program

{

        unsigned int i,j;

        for(i=0;i<1000;i++)

        for(j=0;j<1000;j++);
}

void stepper_A_clock(void)                      // stepper motor anticlockwise rotation sub program

{

        IOSET0 = 0x90000000;                    // Assigning the value to IOSET0 reg
        delay();
        IOCLR0 = 0x90000000;                    // Assigning the value to IOCLR0 reg
        delay();
        IOSET0 = 0xC0000000;
        delay();
        IOCLR0 = 0xC0000000;
        delay();
        IOSET0 = 0x60000000;
        delay();
        IOCLR0 = 0x60000000;
        delay();
        IOSET0 = 0x30000000;
        delay();
        IOCLR0 = 0x30000000;
        delay();
```

```
}
void stepper_clock(void)                          // stepper motor clockwise rotation sub program

{

        IOSET0 = 0x30000000;
        delay();
        IOCLR0 = 0x30000000;
        delay();
        IOSET0 = 0x60000000;
        delay();
        IOCLR0 = 0x60000000;
        delay();
        IOSET0 = 0xC0000000;
        delay();
        IOCLR0 = 0xC0000000;
        delay();
        IOSET0 = 0x90000000;
        delay();
        IOCLR0 = 0x90000000;
        delay();
}


// ***************** END of Function Definations ***********//
```

Program 4: Determine Digital output for a given Analog input using Internal ADC of ARM controller.

```
#include<lpc214x.h>

#define rs 0x00400000

#define rw 0x20000000

#define en 0x10000000

unsigned int result;

float voltage;

char volt[18];

void delay(unsigned int x)

{

        unsigned int i,j;

        for(i=0;i<x;i++)

        for(j=0;j<1275;j++)

}

void cmd( char c)

{

  IOCLR0=0x00003fc0;

  IOSET0=c<<6;

  IOCLR0=rw;

  IOCLR0=rs;

  IOSET0=en;

 delay(100);

 IOCLR0=en;

}
```

```
void data( char c)
{
   IOCLR0=0x00003fc0;
   IOSET0=c<<6;
   IOCLR0=rw;
   IOSET0=rs;
   IOSET0=en;
   delay(100);
   IOCLR0=en;
}
void lcd_str(char  *s)
{
    while(*s)
   {
      data(*s);
      s++;
     delay(20);
  }
}
void adc_init()
{
            AD0CR=0x00210308;
            PINSEL1=0x10000000;
}
```

```
void display(unsigned int n)
{
    if(n==0)
      data(n+0x30);
   if(n)
   {
       display(n/10);
       data((n%10)+0x30);
    }
}
void init()
{
            cmd(0x38);
            cmd(0x0e);
            cmd(0x80); // starting address of the first line
           cmd(0x01);
}
void main()
{
           IODIR0|=0x30403fc0;
           init();
           adc_init();
           while(1)
           {
```

```
          cmd(0x01);
           while(AD0DR3 & (0x80000000)==0);
         result=(AD0DR3 & (0x3ff <<6)); // to store data in result bits(6-15)
         result=result >> 6; //to push the results to data bits
         lcd_str("ADC:");
       cmd(0x84);
       display(result);
       voltage = ((result/1023.0)*3.3); //voltage will have float values
       sprintf(volt,"voltage:%.2f V",voltage);
       cmd(0xc0);
       lcd_str(volt);
      delay(1000);
    }
}
```

Program 5:

Interface a DAC and generate Triangular and Square waveforms.

```
/* Triangle wave */

#include "LPC214X.h"

unsigned int  value;

int main()

{

      PINSEL1|=0x00080000;

      while(1)

     {

        value = 0;

        while ( value != 1023 )

        {

            DACR = ( (1<<16) | (value<<6) );

            value++;

        }

       while ( value != 0 )

      {

         DACR = ( (1<<16) | (value<<6) );

         value--;

      }

   }

}
```

```
/* square wave /*
```

```
#include "LPC214X.h"

unsigned int  result=0x00000040,val;

int main()

{

    PINSEL1|=0x00080000;

    while(1)

   {

       while(1)

      {

              val =0xFFFFFFFF;
              DACR=val;

            {

               break;

           }

      }

     while(1)

     {
         val =0x00000000;

        DACR=val;

         {

           break;

         }

    }

  }

}
```

Program 6:

Interface a 4x4 keyboard and display the key code on an LCD.

```
#include <LPC214x.H>                    /* LPC214x definitions */

#include "lcd.h"

// Matrix Keypad Scanning Routine

// COL1 COL2 COL3 COL4

// 0    1    2    3    ROW 1

// 4    5    6    7    ROW 2

// 8    9    A    B    ROW 3

// C    D    E    F    ROW 4

#define SEG7_CTRL_DIR        IO0DIR

#define SEG7_CTRL_SET       IO0SET

#define SEG7_CTRL_CLR       IO0CLR

#define COL1           (1 << 16)

#define COL2           (1 << 17)

#define COL3           (1 << 18)

#define COL4           (1 << 19)

#define ROW1                    (1 << 20)

#define ROW2                    (1 << 21)

#define ROW3                    (1 << 22)

#define ROW4                    (1 << 23)

#define COLMASK                             (COL1 | COL2 | COL3 | COL4)

#define ROWMASK                             (ROW1 | ROW2 | ROW3 | ROW4)

#define KEY_CTRL_DIR        IO1DIR

#define KEY_CTRL_SET       IO1SET

#define KEY_CTRL_CLR       IO1CLR
```

```
#define KEY_CTRL_PIN       IO1PIN

/////////////// COLUMN WRITE ////////////////////

void col_write( unsigned char data )

{

  unsigned int temp=0;

   temp=(data << 16) & COLMASK;

  KEY_CTRL_CLR |= COLMASK;

  KEY_CTRL_SET |= temp;

}

//////////////////////////////// MAIN  //////////////////////////////////////

int main (void)

{

unsigned char key, i;

unsigned char rval[] = {0x7,0xB,0xD,0xE,0x0};

unsigned char keyPadMatrix[] =

{

    '4','8','B','F',

    '3','7','A','E',

    '2','6','0','D',

    '1','5','9','C'

};

  init_lcd();

  KEY_CTRL_DIR |= COLMASK;    //Set COLs as Outputs

  KEY_CTRL_DIR &= ~(ROWMASK); //  Set ROW lines as Inputs

  lcd_putstring16(0,"Press HEX Keys..");

  lcd_putstring16(1,"Key Pressed =   ");
```

```
 while (1)
{
   key = 0;
   for( i = 0; i < 4; i++ )
   {
      // turn on COL output one by one
                  col_write(rval[i]);
        // read rows - break when key press detected
      if (!(KEY_CTRL_PIN & ROW1))
         break;
      key++;
      if (!(KEY_CTRL_PIN & ROW2))
         break;
      key++;
      if (!(KEY_CTRL_PIN & ROW3))
         break;
      key++;
                  if (!(KEY_CTRL_PIN & ROW4))
         break;
      key++;
   }
         if (key == 0x10)
                  lcd_putstring16(1,"Key Pressed =   ");
         else
                  {        lcd_gotoxy(1,14);
                           lcd_putchar(keyPadMatrix[key]);
```

```
                    }
 }
}
```

Program 7:

Demonstrate the use of an external interrupt to toggle an LED On/Off.

```
#include <LPC214x.H>

int i;

__irq void Ext_ISR(void) // Interrupt Service Routine-ISR

//The _irq keyword tells the compiler that the function is an interrupt routine

{

        IO1DIR |= 0x00010000;

        IO1CLR |=  0x00010000;

        for(i=0; i<3000000;i++);

        IO1SET |=  0x00010000;

        EXTINT |= 0x4;                  //clear interrupt

        VICVectAddr = 0;       // End of interrupt execution

}

void init_ext_interrupt()  // Initialize Interrupt

{

  EXTMODE = 0x4;              //Edge sensitive mode on EINT2

  EXTPOLAR &= ~(0x4);         //Falling Edge Sensitive

  PINSEL0 = 0x80000000; //Select Pin function P0.15 as EINT2

  /* initialize the interrupt vector */

  VICIntSelect &= ~ (1<<16);          // EINT2 selected as IRQ 16

  VICVectAddr5 = (unsigned int)Ext_ISR; // address of the ISR

  VICVectCntl5 = (1<<5) | 16;

//  Basically Vector Address Register store the address of the function i.e. ISR and used to assign
or enable vector IRQ slot..Pointer Interrupt Function (ISR)

  VICIntEnable = (1<<16);             // EINT2 interrupt enabled
  EXTINT &= (0x4);
}
int main (void)
{
   init_ext_interrupt();   // initialize the external interrupt
    while(1);
}
```

Program 8:

Display the Hex digits 0 to F on a 7-segment LED interface, with an appropriate delay in between

```
#include <LPC214x.H>

void delay_led(unsigned long int);

int main(void)

{

   IO0DIR = 0x000007FC;

   while(1)

  {

         IO0CLR = 0x00000FFF;

        IO0SET = 0x00000604;

       delay_led(150000);

      IO0CLR = 0x00000FFF;

      IO0SET = 0x000007E4;

      delay_led(150000);

      IO0CLR = 0x00000FFF;

      IO0SET = 0x00000648;

      delay_led(150000);

      IO0CLR = 0x00000FFF;

      IO0SET = 0x00000618;

      delay_led(150000);

      IO0CLR = 0x00000FFF;

      IO0SET = 0x00000730;

      delay_led(150000);

      IO0CLR = 0x00000FFF;
```

```
 IO0SET = 0x00000690;
 delay_led(150000);
IO0CLR = 0x00000FFF;
IO0SET = 0x00000680;
delay_led(150000);
IO0CLR = 0x00000FFF;
IO0SET = 0x0000063C;
delay_led(150000);
IO0CLR = 0x00000FFF;
IO0SET = 0x00000600;
delay_led(150000);
IO0CLR = 0x00000FFF;
IO0SET = 0x00000630;
delay_led(150000);
IO0CLR = 0x00000FFF;
IO0SET = 0x00000620;
delay_led(150000);
IO0CLR = 0x00000FFF;
IO0SET = 0x00000780;
delay_led(150000);
IO0CLR = 0x00000FFF;
IO0SET = 0x000006C4;
delay_led(150000);
IO0CLR = 0x00000FFF;
IO0SET = 0x00000708;
delay_led(150000);
```

```
    IO0CLR = 0x00000FFF;
    IO0SET = 0x000006C0;
    delay_led(150000);
    IO0CLR = 0x00000FFF;
    IO0SET = 0x000006E0;
    delay_led(150000);
    IO0CLR = 0x00000FFF;
  }
}
void delay_led(unsigned long int count1)
{
   while(count1 > 0) {count1--;}
}
```

Programs Beyond Syllabus

//Demonstrate the use of an external interrupt to operate buzzer

```
#include<LPC214x.h>
void main()
{   IO1DIR=0x0F000000;
  while(1)
  {
  IO1CLR=0x01000000;
  while(IO0PIN & 0x00008000);
  while(!(IO0PIN & 0x00008000));
  IO1SET=0x01000000;
  while(IO0PIN & 0x00008000);
  while(!(IO0PIN & 0x00008000));
  }
}
```

// Interface and Control Relay

```
#include<LPC214x.h>
void main()
{

  IO1DIR=0x0F000000;
  while(1)
  {
  IO1CLR=0x02000000;
  while(IO0PIN & 0x00008000);
  while(!(IO0PIN & 0x00008000));
  IO1SET=0x02000000;
  while(IO0PIN & 0x00008000);
  while(!(IO0PIN & 0x00008000));
  }
}
```

// BLINKING an LED

```
#include <LPC214x.H>
void delay_led(unsigned long int);
int main(void)
{
IO1DIR = 0x00FF0000;
while(1)
{
  IO1CLR = 0x00FF0000;
  delay_led(150000);
  IO1SET = 0x00FF0000;
  delay_led(150000);
}
}
void delay_led(unsigned long int count1)
{
while(count1 > 0) {count1--;}
}
```

// Interface a DAC and generate Sine wave

```
#include "LPC214x.h"
unsigned int  result=0x00000040;
```

```
static int  a[64]={127,139,152,164,176,187,198,208,217,225,233,239,244,249,252,253,254, /*
DAC SAMPLING VALUES*/

253,252,249,244,239,233,225,217,208,198,187,176,164,152,139,127,115,102,90,78,67,56,46,

37,29,21,15,10,5,2,1,0,1,2,5,10,15,21,29,37,46,56,67,78,90,102,115};

int main()

{

  int i;

  PINSEL1|=0x00080000;

  while(1)

  {

    for(i=0;i<64;i++)

    {

      result=(a[i] << 6) & 0x0001FFC0;

      DACR=result;

    }

  }


}
```

VIVA QUESTIONS

1. What is Microcontroller?
2. List out the differences between Microcontroller and Microprocessor.
3. How are Microcontrollers more suitable than Microprocessor for Real Time Applications?
4. What are the General Features of Microcontroller?
5. ARM stands for _________
6. How many registers are there in ARM7?
7. Explain the main features of the ARM Instruction Set.
8. Explain six operating modes of ARM.
9. Explain data processing Instructions of ARM.
10. Explain the following assembler directives

    i) AREA ii) CODE iii) DATA iv) READONLY v)READWRITE
11. Explain the Features of LPC2148
12. What is an embedded system?
13. What are the components of embedded system?
14. Why we use embedded systems?