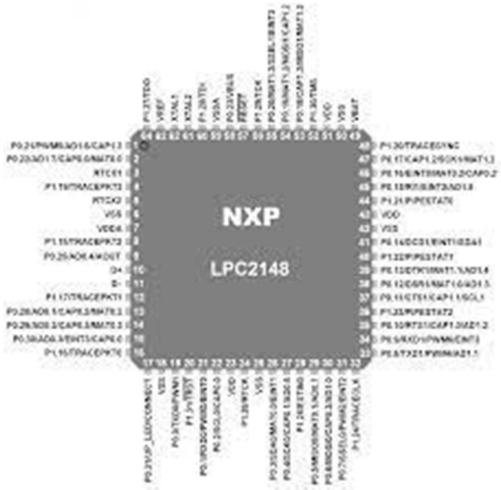
SRI SIDDHARTHA INSTITUTE OF TECHNOLOGY

(A CONSTITUENT COLEGE OF SSAHE)

LAB MANUAL

SUBJECT: MICROPROCESSORS AND EMBEDED SYSTEMS

SUBJECT CODE: 18CS408



DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING,SSIT

Arm Labaroratory

Part A : Develop and exercise the following using ARM Assembly Language

1. To perform Arithmetic operations of two integer numbers.

Addition 32-bit :

```
    AREA ADDTIN,CODE
        ENTRY
        LDR  R0,=VALUE1
        LDR  R1,[R0]
        LDR  R0,=VALUE2
        LDR  R2,[R0]
        ADDS  R3,R2,R1
VALUE1 DCD &BBBBBBBB
VALUE2 DCD &CCCCCCCC
        END
```

Subtraction 32-bit :

```
AREA SUBTIN,CODE
        ENTRY
        LDR  R0,=VALUE1
        LDR  R1,[R0]
        LDR  R0,=VALUE2
        LDR  R2,[R0]
        SUBS  R3,R2,R1
VALUE1 DCD &BBBBBBBB
VALUE2 DCD &CCCCCCCC
        END
```

Subtraction 16-bit :

```
AREA SUBTIN,CODE
        ENTRY
        LDR  R0,=VALUE1
        LDRH  R1,[R0]
        LDR  R0,=VALUE2
        LDRH  R2,[R0]
        SUBS  R3,R2,R1
VALUE1 DCW  &BBBB
VALUE2 DCW  &CCCC
        END
```

Subtraction 8-bit :

```
AREA SUBTIN,CODE
        ENTRY
        LDR  R0,=VALUE1
        LDRB  R1,[R0]
        LDR  R0,=VALUE2
        LDRB  R2,[R0]
        SUBS  R3,R2,R1
VALUE1 DCB  &BB
VALUE2 DCB  &CC
        END
```

Multiplication 32-bit :

```
    AREA MULTIN,CODE
        ENTRY
        LDR  R0,=VALUE1
        LDR  R1,[R0]
        LDR  R0,=VALUE2
        LDR  R2,[R0]
        UMULL  R4,R3,R2,R1
VALUE1 DCD &BBBBBBBB
VALUE2 DCD &CCCCCCCC
        END
```

Multiplication 16-bit :

```
    AREA MULTIN,CODE
        ENTRY
        LDR  R0,=VALUE1
        LDRH  R1,[R0]
        LDR  R0,=VALUE2
        LDRH  R2,[R0]
        UMUL  R3,R2,R1
VALUE1 DCW &BBBB
VALUE2 DCW &CCCC
        END
```

Multiplication 8-bit :

```
    AREA MULTIN,CODE
        ENTRY
        LDR  R0,=VALUE1
        LDRB  R1,[R0]
        LDR  R0,=VALUE2
        LDRB  R2,[R0]
        UMUL  R3,R2,R1
VALUE1 DCB &BB
VALUE2 DCB &CC
        END
```

Addition 64-bit :

```
    AREA ADDTIN,CODE
        ENTRY
        LDR  R0,=VALUE1
        LDR  R1,[R0]
        LDR  R2,[R0,#4]
        LDR  R0,=VALUE2
        LDR  R3,[R0]
        LDR  R4,[R0,#4]
        ADDS  R5,R3,R1
        ADC   R6,R4,R2
        LDR  R0,=RESULT
        STR  R4,[R0]
        STR  R7,[R0,#4]
VALUE1 DCD &12345678,&11111111
VALUE2 DCD &AAAAAAAA,&BBBBBBBB
RESULT DCD 0
        END
```

Subtraction 64-bit :

```
    AREA SUBTIN,CODE
        ENTRY
        LDR R0,=VALUE1
        LDR R1,[R0]
        LDR R2,[R0,#4]
        LDR R0,=VALUE2
        LDR R3,[R0]
        LDR R4,[R0,#4]
        SUBS R5,R3,R1
        SBC R6,R4,R2
        LDR R0,=RESULT
        STR R4,[R0]
        STR R7,[R0,#4]
VALUE1 DCD &12345678,&11111111
VALUE2 DCD &AAAAAAAA,&BBBBBBBB
RESULT DCD 0
         END
```

Multiplication 64-bit :

```
    AREA MULTIN,CODE
        ENTRY
        LDR R0,=VALUE1
        LDR R1,[R0]
        LDR R2,[R0,#4]
        LDR R0,=VALUE2
        LDR R3,[R0]
        LDR R4,[R0,#4]
        UMULL R6,R5,R3,R1
        UMLAL R8,R7,R4,R2
        LDR R0,=RESULT
```

```
        STR R4,[R0]
        STR R9,[R0,#4]
VALUE1 DCD &12345678,&11111111
VALUE2 DCD &AAAAAAAA,&BBBBBBBB
RESULT DCD 0
         END
```

2. To perform Logical,shift,Rotate and Compare instructions of two integer numbers.

```
AREA Program,CODE, READONLY
        ENTRY
        LDR  R0,=VALUE1
        LDR  R1,[R0]
        LDR  R0,=VALUE2
        LDR  R2,[R0]
        AND  R3,R2,R1
        OR R4,R2,R1
        EOR R5,R2,R1
LOOP  LDR R6,R5,R5,LSL,#2
        LDR R7,R5,R5,LSR,#2
        LDR R8,R5,R5,ROR,#2
        CMP R6,R7
        BNE LOOP
VALUE1 DCD &BBBBBBBB
VALUE2 DCD &CCCCCCCC
        END
```

2. To perform the following operation:

I. To move a block of data from one memory segment to another.

II. To exchange block of data between two memory segments.

```
AREA   Block, CODE, READONLY
num   EQU    20    ; set number of words to be copied
    ENTRY          ; mark the first instruction called
start
    LDR    r0, =src     ; r0 = pointer to source block
    LDR    r1, =dst     ; r1 = pointer to destination block
    MOV    r2, #num  ; r2 = number of words to copy
    MOV    sp, #0x400   ; Set up stack pointer (sp)
blockcopy
    MOVS    r3,r2, LSR #3       ; Number of eight word multiples
    BEQ    copywords    ; Fewer than eight words to move?
    PUSH   {r4-r11}       ; Save some working registers
octcopy
    LDM    r0!, {r4-r11}    ; Load 8 words from the source
    STM    r1!, {r4-r11}    ; and put them at the destination
    SUBS   r3, r3, #1       ; Decrement the counter
    BNE    octcopy         ; ... copy more
    POP    {r4-r11}      ; Don't require these now - restore
                         ; originals
copywords
    ANDS   r2, r2, #7       ; Number of odd words to copy
    BEQ    stop              ; No words left to copy?
wordcopy
    LDR    r3, [r0], #4   ; Load a word from the source and
    STR    r3, [r1], #4    ; store it to the destination
    SUBS   r2, r2, #1     ; Decrement the counter
    BNE    wordcopy      ; ... copy more
stop
  AREA   BlockData, DATA, READWRITE
src   DCD    1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,1,2,3,4
```

```
1.    AREA PROGRAM,CODE
        ENTRY
        LDR R2,=0X0A
        LDR R0,=SOURCE1
        LDR R1,=DEST
  UP    LDR R3,[R0],#4
        STR R3,[R1],#4
        SUBS R2,#1
        BNE UP
 STOP B STOP
   AREA SOURCE1,DATA,READONLY
        DCD
0X20,0X10,0X30,0X40,0X50,0X60,0X70,0X80,0X90,0X0A
   AREA DEST,DATA,READWRITE
   SPACE 0X28
        ALIGN
     END
```

dst DCD 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 END	

```
2.  AREA PROGRAM,CODE,READONLY
        ENTRY
        LDR R2,=0X05
        LDR R0,=SOURCE1
        LDR R1,=SOURCE2
UP      LDR R3,[R0]
        LDR R4,[R0]
        STR R4,[R0],#4
        STR R3,[R1],#4
        SUBS R2,#1
        BNE UP
STOP B STOP
  AREA SOURCE1,DATA,READWRITE
        SPACE 0X18
        DCD 0X00,0X00,0X00,0X00,0X00
        AREA SOURCE2,DATA,READWRITE
        SPACE 0X28
        END
```

3. To find largest and smallest of n 16-bit numbers.

```
1.              AREA PROGRAM,CODE
2.              ENTRY
3.              LDR R3,=0X04
4.              LDR R0,=SOURCE
5.              LDR R1,[R0],#4
        TOP     LDR R2,[R0],#4
6.              CMP R1,R2
7.              MOVCS R1,R2
8.              SUBS R3,R3,#1
9.              BNE TOP
10.             LDR R4,=RESULT
11.             STR R1,[R4],#4
12.             LDR R3,=0X04
13.             LDR R0,=SOURCE
14.             LDR R1,[RO],#4
         UP     LDR R2,[R0],#4
15.             CMP R2,R1
16.             MOVCS R1,R2
17.             SUBS R3,R3,#1
18.             BNE UP
19.             STR R1,[R4]
```

```
        STOP B STOP
20.             AREA SOURCE,DATA,READONLY
21.             DCD 0X12,0X34,0X05,0X25,0X10
22.             AREA RESULT,DATA,READWRITE
23.             SPACE 0X0A
24.             END
```

4. To sort a set of 8-bit numbers.

```
    AREA PROGRAM,CODE,READONLY
        ENTRY
        LDR R3,=0X0A
        SUB R3,#1
ABOVE MOV R4,R3
        LDR R0,=SOURCE
UP      LDRB R1,[R0],#01
   LDRB R2,[R0]
        CMP R1,R2
        BCC DOWN
        STRB R1,[R0],#-1
        STRB R2,[R0],#1
DOWN SUBS R4,#01
        BNE UP
        SUBS R3,#01
        BNE ABOVE
STOP B STOP
   AREA SOURCE,DATA,READWRITE
        SPACE 0X0A
        END
```

5.To search for a given element in a set of 32-bit numbers

```
        AREA PROGRAM,CODE,READONLY
        ENTRY
        MOV R4,#0X00
        MOV R5,#0XFF
 BACK LDR R3,=0X05
        LDR R6,=RES
        LDR R0,=SOURCE
        MOV R1,#040
        LDR R2,[R0],#4
        BEQ EXIT
        SUBS R3,R3,#1
        BNE BACK
        STR R4,[R6]
  STOP B STOP
  EXIT STR R5,[R6]
        B STOP
        AREA SOURCE,DATA,READONLY
        DCD 0X10,0X20,0X30,0X40,0X50
        AREA KEY,DATA,READONLY
        DCD 0X05
        AREA RES,DATA,READWRITE
        SPACE 0X04
        END
```

Part B: Develop and execute the following using c :

6. To interface LED and realize Ring and Johnson counter.

Ring counter:

```
#include<LPC21XX.h>
unsigned int delay,i,k;
int main(void)
{
IO0DIR = 0x00FF0000;
while(1)
{
k= 0x00010000;
for(i=0;i<8;i++)
{
IO0SET=0x00FF0000;
for(delay=0;delay<50000;delay++);
IO0CLR = k;
for(delay=0;delay<50000;delay++);
k=k<<1;
}
}
```

Johnson counter:

```
#include<LPC21XX.h>
unsigned int delay,i,k;
int main(void)
{
IO0DIR = 0x00FF0000;
while(1)
{
k= 0x00010000;
for(i=0;i<8;i++)
{
IO0SET=0x00FF0000;
```

```
for(delay=0;delay<50000;delay++);
IO0CLR = k;
for(delay=0;delay<50000;delay++);
k=(k<<1)|k;
}
}
```

7.To realize BCD up/down counter using 7-segment display.

```
#include <LPC21xx.h>
signed int delay, count=0, Switchcount=0;
unsigned int Disp[10]={0x003F0000, 0x00060000, 0x005B0000, 0x004F0000, 0x00660000,0x006D0000,0x007D0000,
0x00070000, 0x007F0000, 0x006F0000 };
#define ALLDISP  0x10000000
int main (void)
{
        PINSEL0 = 0x00000000;
        IO0DIR  = 0xF0FF0000;
        IO0SET |= ALLDISP;
        while(1)
            {
               for(Switchcount=0;Switchcount<=9;Switchcount++)
               {
                 IO0CLR  = 0x00FF0000;
                        for(delay=0;delay<100;delay++)
                                IO0SET = Disp[Switchcount];
                        for(delay=0;delay<1000000;delay++)
                        {}
                        }
               for(Switchcount=9;Switchcount>=0;Switchcount--)
               {

                 IO0CLR  = 0x00FF0000;
                        for(delay=0;delay<100;delay++)
                                IO0SET = Disp[Switchcount];
                        for(delay=0;delay<1000000;delay++)
```

```
                    {}
                }
            }
}
```

8.To genreate the waveforms using DAC.
1.Sinewave 2.Squarewave 3.Trianglewave

Sinewave:

```
int count=0,sinevalue,value;
unsigned char sine_tab[49]=
{0x80,0x90,0xA1,0xB1,0xC0,0xCD,0xDA,0xE5,0xEE,0xF6,0xFB,0xFE,0xFF,0xFE,0xFB,0xF6,0xEE,0xE5,0xDA,0xC
D,0xC0,0xB1,0xA1,0x90,0x80,0x70,0x5F,0x4F,0x40,0x33,0x26,0x1B,0x12,0x0A,0x05,0x02,0x00,0x02,0x05,0x0A,0x
12,0x1B,0x26,0x26,0x33,0x4F,0x5F,0x70,0x80};
int main(void)
{
PINSEL0=0x00000000;
IO0DIR = 0x00FF0000;
Ccount=0;
While(1)
{
for(count=0;count<48;count++)
{
  sinevalue=sine_tab[count];
  value= 0x00FF0000&(sinevalue<<16);
  IO0PIN=value;
}
}
}
```

Square wave:

```
#include<LPC21XX.h>
void delay(void);
int main()
{
   PINSEL0 = 0x00000000;
```

```
  PINSEL1 = 0x00000000;
  IO0DIR = 0x00FF0000;
  While(1)
 {
  IO0PIN = 0x00000000;
  delay();
  IO0PIN = 0x00FF0000;
  delay();
 }
}
Void delay(void)
{
 unsigned int i=0;
 for(i=0;i<=95000;i++);
}
```

Triangle wave :

```
#include<LPC21XX.h>
int main()
{
 unsigned long int temp = 0x00000000;
unsigned int i=0;
IO0DIR=0x00FF0000;
while(1)
{
 for(i=0;i!=0xFF;i++)
  {
   temp= i;
   temp = temp<<16;
   IO0PIN = temp;
  }
for(i=0xFF;i!=0;i--)
  {
   temp= i;
   temp = temp<<16;
   IO0PIN = temp;
  }
}
}
```

9. To interface and rotate stepper motor clockwise and anticlockwise direction.

```
#include<LPC21XX.h>
void clockwise(void);
void anticlockwise(void);
```

```
unsigned long int v1,v2;
unsigned int i=0;j=0,k=0;
int main(void)
{
  PINSEL0 = 0x00FFFFFF;
  IO0DIR = 0x0000F000;
  while(1)
  {
    for(j=0;j<50;j++);
     clockwise();
    for(k=0;k<6500;k++);

    for(j=0;j<50;j++);
     anticlockwise();
    for(k=0;k<6500;k++);
  }
}
void clockwise(void);
{
 v1= 0x00000800;
 for(i=0;i<=3;i++)
  {
    v1=v1<<1;
    v2 = v1;
    v2 = v2 & 0x0000F000;
    IO0PIN = v2;
    for(k=0;k<3000;k++);
   }
  }
void anticlockwise(void);
{
 v1= 0x00010000;
 for(i=0;i<=3;i++)
  {
    v1=v1>>1;
    v2 = v1;
    v2 = v2 & 0x0000F000;
    IO0PIN = v2;
    for(k=0;k<3000;k++);
   }
  }
```

Part C: Develop and execute the following using the Genuine Aurdino UNO board

12. Build a motion detector using a PIR sensor and display appropriate messages.

```
void setup()
{
 pinMode(2,OUTPUT)
 pinMode(3,INPUT)
}
void loop()
{
 if(digitalRead(2)==HIGH)
  {
    digitalWrite(3,HIGH);
    delay(100);
    digitalWrite(3,LOW);
    delay(100);
  }
}
```

13. Traffic light simulation using Breadboard, LED's,Resistors.

```
void setup()
{
 pinMode(8,OUTPUT);
 pinMode(9,OUTPUT);
 pinMode(10,OUTPUT);
}
void loop()
{
 digitalWrite(8,HIGH);
 digitalWrite(9,LOW);
 digitalWrite(10,LOW);
 delay(100);
 digitalWrite(9,HIGH);
 digitalWrite(8,LOW);
 digitalWrite(10,LOW);
 dealy(100);
 digitalWrite(10,HIGH);
 digitalWrite(8,LOW);
 digitalWrite(9,LOW);
 delay(100);
}
```

14. Program to test the UART Function

```
#include <lpc214x.h>

void uart_interrupt(void)__irq;

unsigned char temp;
unsigned char rx_flag=0,tx_flag=0;

int main(void)
{
        PINSEL0=0X0000005;                                  /
IODIR1 = 0X00ff0000;

        U0LCR  = 0X00000083;
        U0DLM = 0X00;
   U0DLL = 0x13;
   U0LCR  = 0X00000003;
        U0IER = 0X03;

   VICVectAddr0 = (unsigned long)uart_interrupt;
   VICVectCntl0 = 0x20|6;
        VICIntEnable = 0x00000040;

   rx_flag = 0x00;
   tx_flag = 0x00;

   while(1)
   {
        while(rx_flag == 0x00);
                rx_flag = 0x00;
      while(tx_flag == 0x00);
                tx_flag = 0x00;
   }
}

// Do this forever
void uart_interrupt(void)__irq
{
        temp = U0IIR;
        temp = temp & 0x06;

   if(temp  == 0x02)
        {
        tx_flag = 0xff;
        VICVectAddr=0;
        }

   else if(temp  == 0x04)
   {
        U0THR = U0RBR;
        rx_flag = 0xff;
```

```
        VICVectAddr=0;
        }
}
```

15. Rain indicator using rain sensor and water resource

```
void setup()
{
      pinMode(12,INPUT);
      pinMode(13,OUTPUT);
}
Void loop()
{
   if(digitalRead(12)==LOW)
       {
           digitalWrite(13,HIGH);
           delay(100);
         digitalWrite(13,LOW);
         delay(500);
}
}
```