(1) Write ARM ALP to transfer 5 datas each of 32-bit from code memory into data memory and demonstrate using software debugger

```
             area reset,data,readonly
             export __Vectors
__Vectors
             dcd 0; initilization of stack memory
              dcd Reset_Handler

                     area mycode,code,readonly
                       entry
                        export Reset_Handler


Reset_Handler
                                  mov r0,#5
                                  ldr r1,=src
                                  ldr r2,=dst

cntd                                ldr r3,[r1],#4
                                  str r3,[r2],#4
                                  subs r0,r0,#1
                                  cmp r0,#0
                                  bne cntd

stop b stop

src dcd  0x12345678, 0xabcdef01, 0x87654321, 0x1379ace1,0x98765432

     area mydata,data,readwrite
dst space 0

            end
```

(2) Write ARM ALP to verify stack operation

(a) Ascending stack using stmea and ldmea

```
   area reset,data,readonly
         export __Vectors
__Vectors
       dcd 0x10001000; initilization of stack memory
               dcd Reset_Handler
        area mycode,code,readonly
                        entry
                            export Reset_Handler

Reset_Handler
                                      mov r0,#1
                                    mov r1,#2

                                    stmea sp!, {r0}
                                    stmea sp!, {r1}

                                    ldmea sp!, {r2}
                                    ldmea sp!, {r3}

stop b stop

     area mydata,data,readwrite
data1 space 0

            end
```

(b) Descending stack using stmfd and ldmfd

```
  area reset,data,readonly
      export __Vectors

__Vectors

      dcd 0x10001000; initilization of stack memory
             dcd Reset_Handler

       area mycode,code,readonly

                     entry
                        export Reset_Handler


Reset_Handler

                                 mov r0,#1
                               mov r1,#2

                               stmfd sp!,{r0}
                               stmfd sp!,{r1}

                               ldmfd sp!,{r2}
                               ldmfd sp!,{r3}


stop b stop

     area mydata,data,readwrite
data1 space 0

          end
```

(3) Write ARM ALP to find the sum of two 64-bit numbers in registers

```
 area reset,data,readonly
        export __Vectors
__Vectors

   dcd 0
        dcd Reset_Handler


  area mycode,code,readonly

  entry
  export Reset_Handler
Reset_Handler
  ; 1st no. = 0x77777777 99999999
  ; 2nd no. = 0x66666666 80000000


  ldr r0,=0x77777777 ;higher word of 1st no.
  ldr r1,=0x99999999 ;lower word of 1st no.
  ldr r2,=0x66666666 ;higher word of 2nd no.
  ldr r3,=0x80000000 ;lower word of 2nd no.

  adds r4,r1,r3;lower word result
  adc r5,r0,r2 ;higher word result

  ldr r6,=sum
  rev r7,r5
  str r7,[r6],#4 ; higher word result
  rev r8,r4
  str r8,[r6] ; lower word result


stop b stop

   area mydata,data,readwrite
sum space 0

  end
```

(4) Write ARM ALP to find the sum of two array of 4 elements

```
 area reset,data,readonly
              export __Vectors
__Vectors

                dcd 0
              dcd Reset_Handler


  area mycode,code,readonly
n equ 4
  entry
  export Reset_Handler
Reset_Handler


   mov r0,#n ;no. of words

   ldr r1,=array1+12 ;1st array
   ldr r2,=array2+12 ;2nd array
   ldr r3,=array3+16;result

cont ldr r4,[r1],#-4
        ldr r5,[r2],#-4
        adcs r6,r4,r5
        rev r7,r6
        str r7,[r3],#-4

   subs r0,r0,#1
   cmp r0,#0
   bne cont
   bcc stop ;if no carry stop
   mov r8,#1
   rev r9,r8
   str r9,[r3]

stop  b stop

array1 dcd 0x11111111,0x22222222,0x33333333,0x44444444
array2 dcd 0xf5555555,0xf6666666,0xf7777777,0xf8888888

 area mydata,data,readwrite
array3 dcd 0

  end
```

(5) ARM ALP to find the difference of two numbers using
(a) Sub instruction (b) 2s complement

```
 area reset,data,readonly
            export __Vectors
__Vectors
              dcd 0
            dcd Reset_Handler


  area hello,code,readonly

  entry
  export Reset_Handler
Reset_Handler
```

;using sub instruction

```
  ldr r0,= 0x55555555
  ldr r1,= 0x22222222
  subs r2,r0,r1
```

;using mvn instruction based on 2's complement

```
  ldr r3,= 0x66666666
  ldr r4,= 0x22222222

  mvn r5,r4 ;1's complement
  adds r6,r3,r5
  adds r6,r6,#1 ; 2's complement
stop b stop

   end
```

(6) Write ARM ALP to find the sum of 3x+4y+9z, where x=2,y=3 and z=4

```
 area reset,data,readonly
             export __Vectors
__Vectors
               dcd 0
             dcd Reset_Handler


  area mycode,code,readonly
  entry
  export Reset_Handler
Reset_Handler

  mov r1,#2
  mov r2,#3
  mov r3,#4

  add r1,r1,r1,lsl#1                 ;r1=3x
  mov r2,r2,lsl#2                      ;r2=4y
  add r3,r3,r3,lsl#3                    ;r3=9z
  add r1,r1,r2                            ;r1=r1+r2 ie. 3x+4y
  add r1,r1,r3                                ;r1=r1+r3 ie. 3x+4y+9z

stop b stop


  end
```

(7) Write ARM ALP to generate first 10 odd numbers/even numbers

```
 area reset,data,readonly
             export __Vectors
__Vectors
               dcd 0x10001000;initialization of stack pointer
             dcd Reset_Handler ;initilization of PC

  area mycode,code,readonly
  entry
  export Reset_Handler
Reset_Handler

               mov r0,#10
            ldr r1,=data1
            mov r2,#1                              ;mov r2,#0 for even numbers

cont        strb r2,[r1],#1
              add r2,r2,#2
            subs r0,r0,#1
            bne cont

stop b stop

  area mydata,data,readwrite
data1 space 0
   end
```

(8) Write ARM ALP to convert binary to ascii

```
 area reset,data,readonly
             export __Vectors
__Vectors
                dcd 0
             dcd Reset_Handler


  area mycode,code,readonly
  entry
  export Reset_Handler
Reset_Handler
 ; Binary(0x01 to 0x09) = Ascii(0x30 to 0x39)
 ; Binary (0x0a to 0x0f)= Ascii(0x41 to 0x46)

  mov r0,#0xc                    ; binary number say 0xc
  cmp r0,#0x0a
  bne nxt
  beq nxt1
nxt     blt nxt2
nxt1  add r2,r0,#0x37                              ; add 0x37 if morethan 9
         b stop
nxt2  add r2,r0,#0x30                                        ;add 0x30 if lessthan a

stop b stop

 end
```

(9) Write ARM ALP to display sum on port0

```
 area reset,data,readonly
              export __Vectors
__Vectors
                dcd 0
              dcd Reset_Handler


  area mycode,code,readonly

FIO0DIR  equ 0x2009c000
FIO0MASK equ 0x2009c010
FIO0PIN  equ 0x2009c014
FIO0SET  equ 0x2009c018
FIO0CLR  equ 0x2009c01c

  entry
  export Reset_Handler
Reset_Handler

  ldr r0,=0x12345678
  ldr r1,=0x11111111

  adds r2,r0,r1 ; r2 = r0 + r1
  rev r3,r2

  ldr r4,=sum        why do we do rev b4 storing to r4
  str r3,[r4]

  ldr r5,=FIO0DIR
  ldr r6,=0xffffffff ; port0 is configured as o/p port
  str r6,[r5]

  ldr r7,=FIO0PIN
  str r2,[r7] ; send sum to port0  through FIOPIN register


stop b stop
   area mydata,data,readwrite
sum space 0
   end
```

(10) **Write C Programming to blink particular LEDs on PORT1 and port2 using FIOSET and FIOCLR registers using hardware**

```
#include <stdio.h>
#include "lpc17xx.h"

void delay(uint32_t);

int main (void)
{
   LPC_GPIO1->FIODIR | = (1<<28) | (1<<31)
   LPC_GPIO2->FIODIR | = (1<<2) | (1<<6)
  while(1)
  {
       LPC_GPIO1->FIOSET   | = (1<<28) | (1<<31)
      LPC_GPIO2->FIOSET    | = (1<<2) | (1<<6)
            delay(100000);
  LPC_GPIO1->FIOCLR | = (1<<28) | (1<<31)
      LPC_GPIO2->FIOCLR    | = (1<<2) | (1<<6)
            delay(100000);
  }
}

   void delay(uint32_t i)
            {
            uint32_t x;
             for(x=0;x<=i;x++);
             }
```

(11) Write C Programming of P0.0 and P0.1 as input pins and P1.7-P1.0 as output, monitor the status of the switch and based on the switch status using SWITCH statement, Make high some pins using software debugger

```
#include <stdio.h>
#include "lpc17xx.h"
uint32_t value;
int main (void)
{
LPC_GPIO0->FIODIR = 0xffffffff; /* LEDs on PORT0 are output */
LPC_GPIO1->FIODIR &=~(3<<0) ; // p1.1-p1.0 as input

 while(1)
 {

   value = ((LPC_GPIO1->FIOPIN & (3<<0))>>0) ;// read the switch status

           switch(value)
           {
           case (0):
           LPC_GPIO0->FIOSET =(1<<0);
           break;

           case (1):
           LPC_GPIO0->FIOSET =(1<<8);
           break;

           case (2):
           LPC_GPIO0->FIOSET =(1<<16);
           break;

           case (3):
           LPC_GPIO0->FIOSET =(1<<24);
           break;

           }
 }
}
```

(12) **Write C Programming to demonstrate up-counting (0x00 to 0x1f) on P2.2 to P2.6 using hardware**

```
#include <stdio.h>
#include "lpc17xx.h"
 uint32_t x,y,a,b;
void delay(uint32_t);

int main (void)
{

LPC_GPIO2->FIODIR  |= (0x1f<<2);           // LEDs on PORT2are output

  while(1)
  {
          for(a=0x00 ;a<=0x1f;a++)
          {
LPC_GPIO0->FIOPIN = (a<<2);
            delay(10000000);
            }
           }
           }

   void delay(uint32_t i)
          {
          uint32_t x;
           for (x=0;x<=i;x++);
           }
```

(13) Write C Programming to generate sound on P1.25 using buzzer

```
      #include<lpc17xx.h>

void delay(unsigned int x);

int main (void)
{

      LPC_GPIO1->FIODIR |=(1<<25);

  while(1)
  {
                        LPC_GPIO0->FIOSET |=(1<<25);
                        delay(1000000);
LPC_GPIO0->FIOCLR  |=(1<<25);
                        delay(5000000);

                  }
            }


  void delay(unsigned int x)
        {
    unsigned int i;
   for(i=0;i<=x;i++);
```

(14) Write C Programming to rotate stepper motor clockwise/anticlockwise using hardware

```
#include "lpc17xx.h"

 void delay(unsigned int x);
 unsigned char a;
int main (void)
{

LPC_GPIO0->FIODIR  |= (0xf<<27)        /*  Configure P0.27,P0.28,P0.29,P0.30 as Outputs */
        LPC_GPIO1->FIODIR  |= (1<<24);  // supply
        while(1)
  {

                 LPC_GPIO0->FIOPIN =(1<<27);
                delay(50000);
                LPC_GPIO0->FIOPIN =(2<<27);
                delay(50000);
                LPC_GPIO0->FIOPIN =(4<<27);
                delay(50000);
                LPC_GPIO0->FIOPIN =(8<<27);
                delay(50000);
                }
        }


  void delay(unsigned int x)
        {
    unsigned int i;
   for(i=0;i<=x;i++);
 }
```