

EXPERIMENT – 13

AIM: Write a program in C language to operate the

1. LCD
2. LEDs
3. 7 Segment display
4. Stepper Motor

APPARATUS REQUIRED: Keil

CODE: LCD

```
//*****INCLUDES*****
#include <LPC214x.h>
//////////////////USER SETTINGS//////////////////
#define DATA_PORT_SET    IOSET1
#define DATA_PORT_CLR    IOCLR1
#define DATA_DIR         IODIR1
#define D7                 23
#define D6                 22
#define D5                 21
#define D4                 20
#define D3                 19
#define D2                 18
#define D1                 17
#define D0                 16
//Set data port pins
#define DATA_PORT          (unsigned
long)((1<<D7)|(1<<D6)|(1<<D5)|(1<<D4))|((1<<D3)|(1<<D2)|(1<<D1)|(1<<D0))

#define CTRL_PORT_SET      IOSET1
#define CTRL_PORT_CLR      IOCLR1
#define CTRL_DIR           IODIR1
#define CTRL_RS            24
#define CTRL_EN            25
//////////////////End of USER SETTINGS//////////////////
//////////////////
#include "delay.h"
#include "lcd.h"
//*****End of INCLUDES*****

int main()
{
    int i;
    init_lcd();

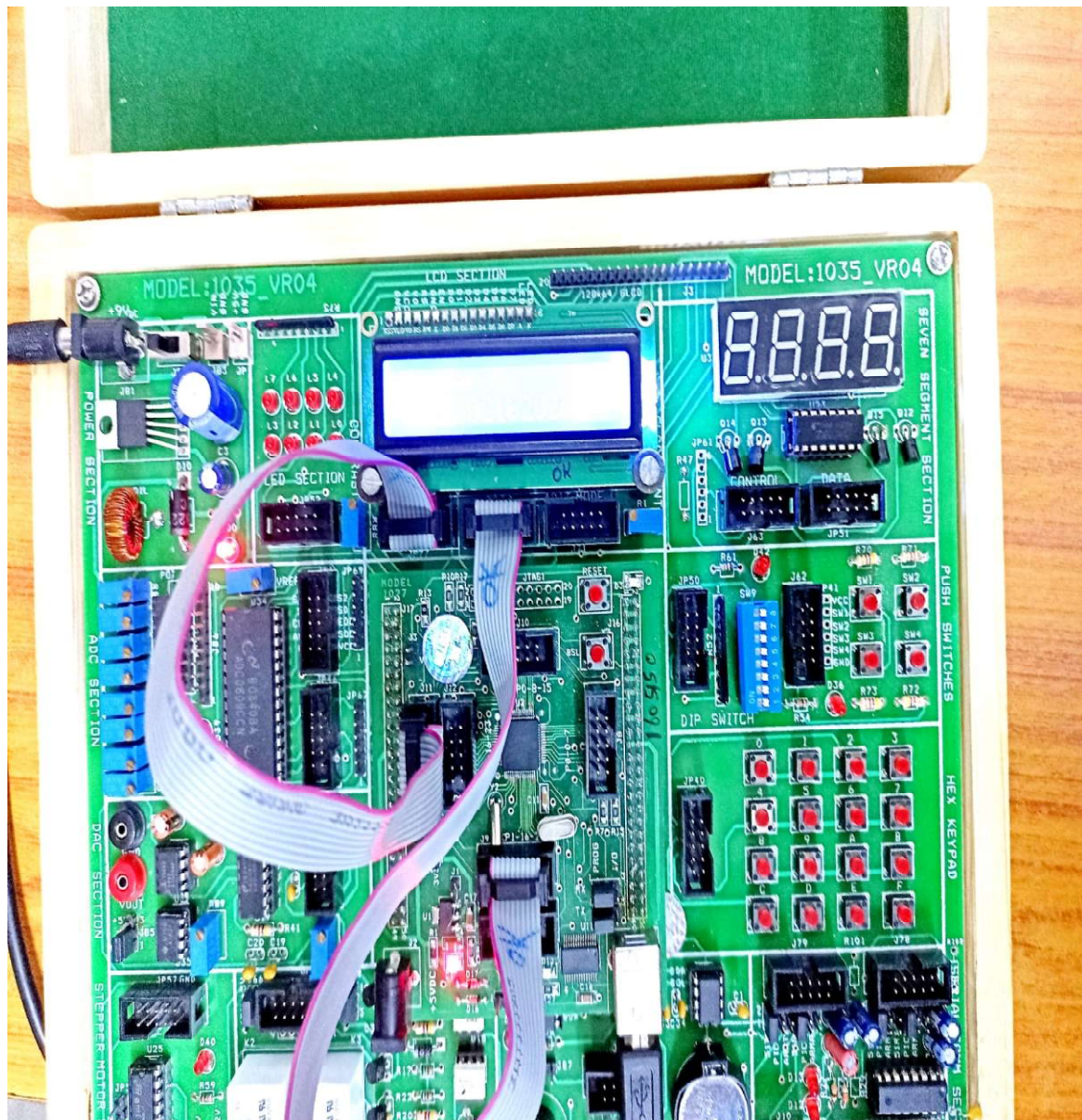
    while(1)
    {
        cmd_lcd(0x80);
        string_lcd("jessica");
        cmd_lcd(0xc0);
        string_lcd(" ** jessica 602162008** ");
    }
}
```

```

for(i=0;i<5;i++)
{
    cmd_lcd(0x1C);
    delay_ms(1000);
}
for(i=0;i<5;i++)
{
    cmd_lcd(0x18);
    delay_ms(1000);
}
}
}

```

OBSERVATION:



CODE: LEDs

```
/*******INCLUDES*****  
#include <LPC214x.h>  
#include "delay.h"  
/*******End of INCLUDES*****  
  
/*******Private Macro Definitions*****  
#define LED_SET          IOSET1  
#define LED_CLR          IOCLR1  
#define LED_DIR          IODIR1  
#define LED_PIN          IOPIN1  
  
#define LED7              23  
#define LED6              22  
#define LED5              21  
#define LED4              20  
#define LED3              19  
#define LED2              18  
#define LED1              17  
#define LED0              16  
//Set data port pins  
#define LED_PORT          (unsigned  
long)((1<<LED7)|(1<<LED6)|(1<<LED5)|(1<<LED4))((1<<LED3)|(1<<LED2)|(1<<LED1)|(1<<LED0))  
/*******End of Private Macro Definitions*****  
  
int i,a,b,x,y;  
  
int main()  
{  
    LED_DIR |= (unsigned long)(LED_PORT);          //initialize D0:D7 pins as output  
    LED_CLR |= (unsigned long)(LED_PORT);          //initialize D0:D7 pins as output  
  
    while(1)  
    {  
        ////////////////////////////////// nibble on off  
        LED_CLR |= (0xf0) << LED0;  
        LED_SET |= (0x0f) << LED0;  
        delay_ms(500);  
        LED_CLR |= (0x0f) << LED0;  
        LED_SET |= (0xf0) << LED0;  
        delay_ms(500);  
        ////////////////////////////////// odd even on off  
        LED_CLR |= (0xff) << LED0;  
        LED_SET |= (0xaa) << LED0;  
    }  
}
```

```

    delay_ms(500);
    LED_CLR |= (0xff) << LED0;
    LED_SET |= (0x55) << LED0;
    delay_ms(500);
    ////////////////////////////////// left shift
    LED_PIN = (0x01) << LED0;
    delay_ms(500);
    for(i=0;i<7;i++)
    {
        LED_PIN = LED_PIN << 1;
        delay_ms(500);
    }
    ////////////////////////////////// rotate left
    LED_PIN = (0x01) << LED0;
    delay_ms(500);
    for(i=0;i<7;i++)
    {
        LED_PIN = LED_PIN << 1 | (0x01 << LED0);
        delay_ms(500);
    }
    ////////////////////////////////// right shift
    LED_PIN = (0x80) << LED0;
    delay_ms(500);

    for(i=0;i<7;i++)
    {
        LED_PIN = (LED_PIN >> 1) & (0X7F << LED0);
        delay_ms(500);
    }
    ////////////////////////////////// rotate right
    LED_PIN = (0x80) << LED0;
    delay_ms(500);

    for(i=0;i<7;i++)
    {
        LED_PIN = (LED_PIN >> 1);
        delay_ms(500);
    }
}
}
}

```

CODE: 7 Segment Display

```

/*
connection of 7-seg
    data    P0.16 to P0.23
    control P0.8 to P0.15
*/

```

```

#include"lpc214x.h"
#include"delay.h"

```

```

int main()

```



```
while(1)
{

    PORT_CLR = (1<<m1) | (1<<m2);
    PORT_SET = (1<<m1);
    delay_ms(4000);
    PORT_CLR = (1<<m1) | (1<<m2);
    delay_ms(1000);
    PORT_CLR = (1<<m1) | (1<<m2);
    PORT_SET = (1<<m2);
    delay_ms(4000);
    PORT_CLR = (1<<m1) | (1<<m2);
    delay_ms(1000);

}
}
```

RESULT:

The C codes Written above have been implemented and verified successfully.

EXPERIMENT – 14

AIM: Write a program in ARM assembly language to solve the equation

1. ax^2+by^2
2. $6(x+y)+2z+4$

APPARATUS REQUIRED: Keil

CODE:

; To solve the equation ax^2+by^2

AREA PROGRAM, CODE, READONLY

ENTRY

MAIN

```
LDR R0, VALUE1 ; Load the data x
LDR R1, VALUE2 ; Load the data y
LDR R2, VALUE3 ; Load the data a
LDR R3, VALUE4 ; Load the data b
MUL R4,R0,R0; performing  $x^2$ 
MUL R5,R4,R2; ; performing  $a(x^2)$ 
LDR R7,=0x00002000
STR R5,[R7],#4
MUL R4,R1,R1; performing  $y^2$ 
MUL R6,R4,R3; performing  $b(y^2)$ 
STR R6,[R7],#4
ADD R6,R6,R5 ; performing  $a(x^2)+b(y^2)$ 
STR R6,[R7]
NOP
```

AREA PROGRAM, DATA, READONLY

VALUE1 DCD &00000001

VALUE2 DCD &00000002

VALUE3 DCD &00000004

VALUE4 DCD &00000005

END

OBSERVATION:

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers

Register	Value
Current	
R0	0x00000001
R1	0x00000002
R2	0x00000004
R3	0x00000005
R4	0x00000004
R5	0x00000004
R6	0x00000018
R7	0x00002008
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x0000F004
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC \$	0x0000F004
Mode	Supervisor
States	15376
Sec	0.00000000

exp 14.asm*

```
1 ; To solve the equation ax^2+by^2
2 AREA PROGRAM, CODE, READONLY
3 ENTRY
4 MAIN
5
6 LDR R0, VALUE1 ; Load the data x
7 LDR R1, VALUE2 ; Load the data y
8 LDR R2, VALUE3 ; Load the data a
9 LDR R3, VALUE4 ; Load the data b
10 MUL R4,R0,R0; performing x^2
11 MUL R5,R4,R2; ; performing a(x^2)
12 LDR R7,=0x00002000
13 STR R5,[R7],#4
14 MUL R4,R1,R1; performing y^2
15 MUL R6,R4,R3; performing b(y^2)
16 STR R6,[R7],#4
17 ADD R6,R6,R5 ; performing a(x^2)+b(y^2)
18 STR R6,[R7]
19 NOP
20
21 AREA PROGRAM, DATA, READONLY
22 VALUE1 DCD &00000001
23 VALUE2 DCD &00000002
24 VALUE3 DCD &00000004
25 VALUE4 DCD &00000005
26 END
27
28
```

Memory 2

Address: 0x00002000

0x00002000:	00 00 00 04
0x00002004:	00 00 00 14
0x00002008:	00 00 00 18

; To solve the equation $6(x + y) + 2z + 4$

AREA PROGRAM, CODE, READONLY

ENTRY

MAIN

LDR R0, VALUE1 ; Load the data x

LDR R1, VALUE2 ; Load the data y

LDR R2, VALUE3 ; Load the data z

MOV R3,#6

MOV R4,#2

ADD R5,R0,R1; performing $x+y$

MUL R6,R5,R3 ; performing $6(x+Y)$

LDR R7,=0x00002000

STR R6,[R7],#4

MUL R8,R2,R4; performing $2z$

STR R8,[R7],#4

ADD R6,R6,R8

ADD R6,R6,#4 ; performing $6(x+y)+2z+4$

STR R6,[R7]

AREA PROGRAM, DATA, READONLY

VALUE1 DCD &00000001

VALUE2 DCD &00000001

VALUE3 DCD &00000004

END

OBSERVATION:

The screenshot displays the Keil uVision IDE interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. Below the menu is a toolbar with various icons for file operations, editing, and debugging.

The **Registers** window on the left shows the current state of the processor registers. The **Current** register set includes R0 through R15, CPSR, and SPSR. The values are as follows:

Register	Value
R0	0x00000001
R1	0x00000001
R2	0x00000004
R3	0x00000006
R4	0x00000002
R5	0x00000002
R6	0x00000018
R7	0x00002008
R8	0x00000008
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x0000F004
CPSR	0x000000D3
SPSR	0x00000000

The main window shows the assembly code for **exp 14b.asm**. The code is as follows:

```
1 ; To solve the equation 6(x+y)+2z+4
2 AREA PROGRAM, CODE, READONLY
3 ENTRY
4 MAIN
5
6 LDR R0, VALUE1 ; Load the data x
7 LDR R1, VALUE2 ; Load the data y
8 LDR R2, VALUE3 ; Load the data z
9 MOV R3, #6
10 MOV R4, #2
11 ADD R5, R0, R1; performing x+y
12 MUL R6, R5, R3 ; performing 6(x+y)
13 LDR R7, =0x00002000
14 STR R6, [R7], #4
15 MUL R8, R2, R4; performing 2z
16 STR R8, [R7], #4
17 ADD R6, R6, R8
18 ADD R6, R6, #4 ; performing 6(x+y)+2z+4
19 STR R6, [R7]
20
21 AREA PROGRAM, DATA, READONLY
22 VALUE1 DCD &00000001
23 VALUE2 DCD &00000001
24 VALUE3 DCD &00000004
25 END
26
27
28
```

A green highlight is placed over line 10, indicating the current instruction being executed. The **Memory 2** window on the right shows the memory contents at addresses 0x00002000, 0x00002004, and 0x00002008:

Address	Value
0x00002000	00 00 00 0C
0x00002004	00 00 00 08
0x00002008	00 00 00 18

RESULT:

The output of the codes executed above have been verified successfully.