RAMAIAH
Institute of Technology

Autonomous Institute, Affiliated to VTU
Vidya Soudha, MSR Nagar,
MSRIT Post, Bangalore-54

Department of Information Science & Engineering

LABORATORY MANUAL

(FOR THE ACADEMIC YEAR 2019-2020)

MICROCONTROLLERS LAB – ISL48

MICROCONTROLLERS LABORATORY

Course Code: ISL48 **Credits: 0:0:1**
Prerequisite: Nil **Contact Hours: 14P**
Course Coordinator: Mr. George Philip C

Exercises

1. **Familiarizing the Keil MicrovisionV IDE**
   a. Create a project, Edit an ASM file, Build, and Debug. Observe Disassembly window, Register and Memory contents in Step mode and in Run Mode.
   b. Execute a sample ARM Assembly Language Program to add two numbers in registers and store the sum in a register.

2. **ARM Assembly Language Programming Practice using Keil MicrovisionV # I**
   a. ALP to add first 5 natural numbers. Store sum in register.
   b. ALP to add first 10 odd numbers. Store sum in register.
   c. ALP to compute sum of 5 terms of an arithmetic progression. First term is 3, common difference is 7. Store sum in register.
   d. ALP to compute sum of squares of 5 numbers starting from 1. Write and use procedure SQU. Store sum in register.

3. **ARM Assembly Language Programming Practice using Keil MicrovisionV # II**
   a. ALP to add the first n even numbers. Store the result in a memory location.
   b. ALP to generate a geometric progression with a limit n. Display the results in memory.

4. **ARM ALP # I**
   a. ALP to find the arithmetic progression with a=3, d=7.
   b. ALP to find the sum of cubes of the first n natural numbers.

5. **ARM ALP # II**
   a. ALP to count the number of zeroes and ones in a binary number.
   b. ALP to find the average of ten 16-bit numbers stored in memory.

6. **ARM ALP # III**
   a. ALP to find the factorial of a number.
   b. ALP to generate the first n Fibonacci numbers.

7. **ARM ALP # IV**
   a. ALP to find the sum of digits of a number.
   b. ALP to convert BCD number to binary.

8. **ARM ALP # V**
   a. ALP to find nCr.
   b. ALP to find nPr.

9. **ARM ALP # VI**
   a. ALP to implement Bubble Sort on an array of integers.
   b. ALP to implement Binary Search on an array of integers.

10. ARM ALP # VII

a. ALP to check whether the given number is palindrome.
b. ALP to count the number of times a substring is repeated in the string.

11. ARM C Programming Practice using Keil MicrovisionV # I

a. C program to toggle the lowest pin of Port 0 with a delay between the two states. Observe and record the waveform obtained using the Logic Analyzer in the Keil simulator.

12. ARM C Programming # I

a. C program to generate an asymmetric square wave of 120Hz and having a duty cycle of 25% using the Timer0 module.
b. C program to generate a square wave using Timer0 in the interrupt mode.

13. ARM C Programming # II

a. C program to make a LED glow at different brightness levels (low to high) with brightness levels varying over duration of 2s. Demonstrate using logic analyzer window.

14. ARM C Programming # III

a. C program to display the string ‘I LOVE ISE’ in the serial window of UART1.

Course Outcomes (COs):

At the end of the course, students will be able to-

1. Write assembly language programs for the ARM7 ISA. (PO-1,2,3) (PSO-1,2)
2. Write C programs for interfacing peripherals to the ARM7 MCU. (PO-1,2,3) (PSO-1,2)
3. Execute and Debug assembly language and C programs using a simulator. (PO-1,2,4,5) (PSO-1,2)

Exercise # 1

a)

List all the steps.

b)

```
        AREA PROG1,CODE,READONLY
ENTRY
                MOV R0,#0x78
                MOV R1,#0x21
                ADD R3,R1,R0
STOP            B       STOP
                END
```

Exercise # 2

a)

```
        AREA PROG2,CODE,READONLY
ENTRY
                MOV R0,#0
                MOV R1,#0
BACKK           ADD R0,R0,#1
                ADD R1,R1,R0
                CMP R0,#5
                BNE BACKK
GO              B       GO
                END
```

b)

```
        AREA PROG3,CODE,READONLY
ENTRY
                MOV R1,#1
                MOV R2,#9
                MOV R3,#1
BACKK           ADD R3,R3,#2
                ADD R1,R1,R3
                SUBS R2,R2,#1
                BNE BACKK
GO              B       GO
                END
```

c)

```
        AREA PROG4,CODE,READONLY
ENTRY
                MOV R3,#0
                MOV R1,#3
                MOV R2,#0
BACKK           ADD R3,R3,R1
                ADD R1,R1,#7
                ADD R2,R2,#1
                CMP R2,#5
                BNE BACKK
GO              B       GO
                END
```

d)

```
        AREA PROG5,CODE,READONLY
ENTRY
                MOV R7,#0
                MOV R2,#1
LOOP            BL SQU
                ADD R7,R7,R4
                ADD R2,R2,#1
                CMP R2,#6
                BNE LOOP
GO              B       GO
SQU             MUL R4,R2,R2
                MOV PC,LR
                END
```

Exercise # 3

a)

```
        AREA PROG6,CODE,READONLY
N               RN 1
RESULT          RN 2
EVEN_NUMBER     RN 3
ENTRY
            MOV N,#5
            MOV RESULT,#0
            MOV EVEN_NUMBER,#2
            MOV R4,#0x40000000
LOOP        ADD RESULT,RESULT,EVEN_NUMBER
            ADD EVEN_NUMBER,EVEN_NUMBER,#2
            SUBS N,N,#1
            BNE LOOP
            STR RESULT,[R4]
STOP        B STOP
END
```

b)

```
        AREA PROG7,CODE,READONLY
A           RN 1
D           RN 2
N           RN 3
ENTRY
            MOV A,#1
            MOV D,#2
            MOV N,#10
            MOV R5,#0x40000000
LOOP        MUL R6,A,D
            MOV A,R6
            STR A,[R5],#4
            SUBS N,N,#1
            BNE LOOP
STOP        B STOP
END
```

Exercise # 4

a)

```
          AREA PROG8,CODE,READONLY
ENTRY
                    MOV R1,#3
                    MOV R2,#1
                    LDR R3,=PRO
                    STR R1,[R3]
                    ADD R1,R1,#7
BACKK               STR R1,[R3,#4]!
                    ADD R1,R1,#7
                    ADD R2,R2,#1
                    CMP R2,#10
                    BNE BACKK
GO                  B GO
          AREA PROGRESSION,DATA,READWRITE
PRO                 SPACE 10
                    END
```

b)

```
          AREA PROG9,CODE,READONLY
N                             RN 1
NPLUSONE                      RN 2
TEMP                          RN 3
RESULT                        RN 4
ENTRY
                    MOV R5,#0x40000000
                    LDR N,=3
                    ADD NPLUSONE,N,#1
                    MUL TEMP,N,NPLUSONE
                    MOV TEMP,TEMP,LSR #1
                    MUL RESULT,TEMP,TEMP
                    STR RESULT,[R5]
STOP                B STOP
                    END
```

Exercise # 5

a)

```
          AREA PROG10,CODE,READONLY
NUMBER                      RN 1
NUMONES                     RN 10
NUMZEROES                   RN 11
ENTRY
                MOV R5,#0x40000000
                LDR NUMBER,=0xA
                MOV NUMONES,#0
                MOV NUMZEROES,#0
LOOP            LSRS NUMBER,#1
                ADDCS NUMONES,#1
                ADDCC NUMZEROES,#1
                CMP NUMBER,#0
                BNE LOOP
                STR NUMONES,[R5]
                STR NUMZEROES,[R5,#4]
STOP            B STOP
                END
```

b)

```
          AREA PROG11,CODE,READONLY
ENTRY
                LDR R7,=TABLE
                MOV R0,#9
                LDRH R1,[R7]
BACKK           LDRH R2,[R7,#2]!
                ADD R1,R1,R2
                SUBS R0,R0,#1
                BNE BACKK
                MOV R3,#10
                MOV R4,#0
                MOV R5,R1
BACKK1          SUBS R5,R5,R3
                ADDPL R4,R4,#1
                BPL BACKK1
          ADDMI R5,R5,R3
GO        B GO
TABLE     DCW 1000,2564,8936,344,5667,908,786,654,9871,456
          END
```

Exercise # 6

a)

```
        AREA PROG12,CODE,READONLY
N               RN 1
FACT            RN 2
ENTRY
                MOV N,#10
                MOV FACT,#1
LOOP            MUL FACT,N,FACT
                SUBS N,N,#1
                BNE LOOP
STOP            B       STOP
                END
```

b)

```
        AREA PROG13,CODE,READONLY
ENTRY
                MOV R1,#1
                LDR R2,=TABLE
                LDR R3,=NUMFIBONACCI
                LDRB R6,[R3]
                STRB R1,[R2],#1
                MOV R3,#0
                MOV R4,#0
                MOV R5,#1
                SUB R6,R6,#1
BACKK           ADD R4,R3,R1
                STRB R4,[R2],#1
                MOV R3,R1
                MOV R1,R4
                ADD R5,R5,#1
                CMP R5,R6
                BLS BACKK
GO              B GO
```

```
NUMFIBONACCI     DCB    0x0A
         AREA NUMBER,DATA,READWRITE
TABLE           SPACE 60
                END
```

Exercise # 7

a)

```
          AREA PROG14,CODE,READONLY
DIVIDEND              RN 1
DIVISOR               RN 2
QUOTIENT              RN 3
REMAINDER             RN 4
RESULT                RN 5
ENTRY
                LDR DIVIDEND, =12345
                MOV DIVISOR,#10
                MOV RESULT,#0
LOOP            BL DIV
                ADD RESULT,REMAINDER,RESULT
                CMP QUOTIENT,#0
                MOVNE DIVIDEND,QUOTIENT
                BNE LOOP
STOP            B STOP

DIV             MOV QUOTIENT,#0
LOOP2           SUBS DIVIDEND,DIVIDEND,DIVISOR
                ADDPL QUOTIENT,QUOTIENT,#1;QUOTIENT
                BPL LOOP2
                ADDMI REMAINDER,DIVIDEND,DIVISOR
                BX LR
                END
```

b)

```
        AREA PROG15,CODE,READONLY
RADIX                   RN 0
LOWERNIBBLEMASK         RN 10
UPPERNIBBLEMASK         RN 11
LOWERNIBBLE             RN 3
UPPERNIBBLE             RN 4
RESULT                  RN 5
NUMBYTES                RN 6
BYTE                    RN 2
ENTRY
            MOV RADIX,#10
            MOV LOWERNIBBLEMASK,#0x0F
            MOV UPPERNIBBLEMASK,#0xF0
            MOV RESULT,#0
            MOV NUMBYTES,#4
            LDR R1,=NUMBER
            ADD R1,R1,NUMBYTES
            SUB R1,R1,#1
LOOP        LDRB BYTE,[R1]
            SUB R1,R1,#1
            AND LOWERNIBBLE,BYTE,LOWERNIBBLEMASK
            AND UPPERNIBBLE,BYTE,UPPERNIBBLEMASK
            LSR UPPERNIBBLE,#4
            MLA RESULT,RADIX,RESULT,UPPERNIBBLE
            MLA RESULT,RADIX,RESULT,LOWERNIBBLE
            SUBS NUMBYTES,NUMBYTES,#1
            BNE LOOP
STOP        B STOP

NUMBER      DCD 0x00000127
            END
```

Exercise # 8

a)

```
          AREA PROG16,CODE,READONLY
DIVIDEND              RN 1
DIVISOR               RN 2
QUOTIENT              RN 3
REMAINDER             RN 4
N                     RN 10
R                     RN 11
NDR                   RN 12
ENTRY
                MOV N,#6
                MOV R,#3
                LDR R5,=0X40000000
                SUB NDR,N,R
                MOV DIVIDEND,N
                BL FACT
                MOV N,DIVISOR
                MOV DIVIDEND,NDR
                BL FACT
                MOV DIVIDEND,N
                BL DIV
                STR QUOTIENT,[R5]
STOP            B STOP

FACT            MOV DIVISOR,#1
LOOP2           MUL DIVISOR,DIVIDEND,DIVISOR
                SUBS DIVIDEND,DIVIDEND,#1
                BNE LOOP2
                BX LR

DIV             MOV QUOTIENT,#0
LOOP3           SUBS DIVIDEND,DIVIDEND,DIVISOR
                ADDPL QUOTIENT,QUOTIENT,#1
                BPL LOOP3
                ADDMI REMAINDER,DIVIDEND,DIVISOR
                BX LR
                END
```

b)

```
        AREA PROG17,CODE,READONLY
DIVIDEND                RN 1
DIVISOR                 RN 2
QUOTIENT                RN 3
REMAINDER               RN 4
N                       RN 10
R                       RN 11
NDR                     RN 12
ENTRY
                LDR R5,=0X40000000
                MOV N,#6
                MOV R,#3
                SUB NDR,N,R
                MOV DIVIDEND,N
                BL FACT
                MOV N,DIVISOR
                MOV DIVIDEND,R
                BL FACT
                MOV R,DIVISOR
                MOV DIVIDEND,NDR
                BL FACT
                MOV DIVIDEND,N
                MUL DIVISOR,R,DIVISOR
                BL DIV
                STR QUOTIENT,[R5]
STOP            B STOP
FACT            MOV DIVISOR,#1
LOOP2           MUL DIVISOR,DIVIDEND,DIVISOR
                SUBS DIVIDEND,DIVIDEND,#1
                BNE LOOP2
                BX LR
DIV             MOV QUOTIENT,#0
LOOP3           SUBS DIVIDEND,DIVIDEND,DIVISOR
                ADDPL QUOTIENT,QUOTIENT,#1
                BPL LOOP3
                ADDMI REMAINDER,DIVIDEND,DIVISOR
                BX LR
                END
```

Exercise # 9

a)

```
        AREA PROG18,CODE,READONLY
ENTRY
            MOV R0,#13
            LDR R1,=NUMS
            LDR R2,=0X40000000
LOOP1       LDR R3,[R1],#4
            STR R3,[R2],#4
            SUBS R0,R0,#1
            BNE LOOP1
            MOV R12,#13
            LDR R11,=0X40000000
LOOP3       MOV R1,R11;INITIALISING I
            SUBS R12,R12,#1
            MOVNE R0,R12
            BEQ STOP
LOOP2       ADD R2,R1,#4
            LDR R3,[R1]
            LDR R4,[R2]
            CMP R3,R4;
            STRPL R3,[R2]
            STRPL R4,[R1]
            ADD R1,R1,#4
            SUBS R0,R0,#1
            BNE LOOP2
            CMP R12,#0
            BNE LOOP3
STOP        B STOP
NUMS        DCD 20,19,18,17,16,15,14,13,12,11,10,9,8
            END
```

b)

```
        AREA PROG19,CODE,READONLY
ENTRY

STORAGE             EQU 0X40000000
                    LDR SP, =STORAGE
                    LDR R3, =STORAGE + 200

NUM                 EQU 11
SIZE                EQU 1

                    ADR R6, ARRAY
                    MOV R1, #0
                    MOV R2, #NUM - 1
                    MOV R5, #17                          ;search value
                    STMDB R3!, {R6,R1,R2,R5, R0}

MAIN                BL FINDIT
                    B MAIN

FINDIT
                    STMDB SP!, {R4,R7,R8,R9,R10,R11,R12,LR}
                    LDMFD R3!, {R11,R7,R8,R10, R0}
                    CMP R7, R8
                    BGT STOP

                    ADD R9, R7, R8
                    MOV R9, R9, ASR #1
                    LDR R12, [R11, R9,LSL #2]
                    ADD R11, R9, LSL #2
                    MOV R0, R11
                    ADD R4, R9, LSL #1
                    CMP R12, R10
                    SUBGT R8, R9, #1
                    ADDLE R7, R9, #1
                    LDR R11, [R3, #-4]
                    STMFD R3!, {R11,R7,R8,R10, R0}
                    LDMIA SP!, {R4,R7,R8,R9,R10,R11,R12,PC}

                    MOV PC, LR

STOP                B STOP

ARRAY               DCD 3,6,8,12,17,22,45,67,99,2089,30001
                    END
```

Exercise # 10

a)

```
        AREA PROG20,CODE,READONLY
ENTRY
            LDR R1, = 12321
            MOV R6,R1
            MOV R2,#10
            MOV R5,#0
            MOV R10,#10
LOOP        BL DIV
            MLA R5,R10,R5,R4
            CMP R3,#0
            MOVNE R1,R3
            BNE LOOP
            CMP R5,R6
            MOVEQ R7,#1
            MOVNE R7,#0
STOP        B STOP

DIV         MOV R3,#0
LOOP2       SUBS R1,R1,R2
            ADDPL R3,R3,#1
            BPL LOOP2
            ADDMI R4,R1,R2
            BX LR
            END
```

b)

```
        AREA PROG21,CODE,READONLY
CNT RN 7
ENTRY
                LDR R1,=M
                LDR R2,=S
                MOV R12,R2
                MOV CNT,#0

LOOP            LDRB R3,[R1]
                LDRB R4,[R2]

                CMP R4,#0
                ADDEQ CNT,CNT,#1
                MOVEQ R2,R12
                BEQ LOOP

                CMP R3,R4
                ADDEQ R2,R2,#1
                MOVNE R2,R12
                ADD R1,R1,#1
                BEQ LOOP

                CMP R3,#0
                BEQ STOP
                BNE LOOP

STOP            B STOP

M               DCB "ABCABC",0
S               DCB "ABC",0
                END
```

Exercise # 11

a)

```
#include<LPC214X.h>
void delay(int);
int main()
{
        IODIR0 = 0x00000001;
        while(1){
                IOSET0 = 0x00000001;
                delay(500);
                IOCLR0 = 0x00000001;
                delay(500);
                }


}
                void delay(int n)
        {
                inti =0;
                for(i = 0;i<n;i++);
        }
```

Exercise # 12

a)

```
#include<lpc214x.h>
//124373 * 0.25 = 31093 = 7975H
//124373 * 0.75 = 93280; 93280/2 = 46640 = B630
voidon_delay(void){
        T0MR0=0x7974;
        T0PR=0;
        T0TCR=1;
        while(T0TC!=T0MR0);
        T0TCR=2;
        T0TC=0;
}
voidoff_delay(void){
        T0MR0=0xB630;
        T0PR=1;
        T0TCR=1;
        while(T0TC!=T0MR0);
        T0TCR=2;
        T0TC=0;
}
int main(void){
        T0MCR=4;
        IODIR1=0x00010000;
        while(1){
                IOSET1=1<<16;
                on_delay();
                IOCLR1=1<<16;
                off_delay();
        }
}
```

b)

```
#include<LPC214X.h>
unsigned int x = 0;
__irq void Timer0_ISR (void){
	x ^= 1;
	if(x)
		IOSET1 = 1 << 20;
	else
		IOCLR1 = 1 << 20;
		T0IR = 0x01;
		VICVectAddr = 0x00000000;
}
int main(){
	IODIR1 = 0x0FFFFFFF;
	T0MCR = 0x00000003;
	T0MR0 = 0x3456FF;
	VICVectAddr4 = (unsigned)Timer0_ISR;
	VICVectCntl4 = 0x00000024;
	VICIntEnable = 0x00000010;
	T0TCR = 1;
	for(;;);
}
```

Exercise # 13

a)

```
# include <lpc214x.h>
voidpwm_init(void)
{
PINSEL0|=0x00000002;
PWMPR= 0x2;
PWMPCR=0x00000200;
PWMMR0=0xC37F;
PWMMCR=0x00000002;
PWMTCR=0x00000009;
}
        int main()
{
        inti;
        pwm_init();
        while(1)

                {  for(i=0;i<10;i++)

        {PWMMR1=0xFFF+(0xFF5*i);
        PWMLER=0x02;

        }}}
```

Exercise # 14

a)

```
#include<LPC214X.h>
voidinit()
{
        PINSEL0=0x05;
        U0FCR=0x07;
        U0LCR=0x83;
        U0DLL=0x5D;
        U0DLM=0x00;
        }
void delay()
{
        inti;
        for(i=0;i<250;i++);
}
int main()
{
        unsigned char p[]="I LOVE ISE\n";
        int z;
        init();
        for(z=0;z<=24;z++)
                                {
                                        U0THR=p[z];
                                        while(!(U0LSR&0x20));
                                        delay();
                                }
                                while(1);
}
```