EE2080 Microprocessor Systems Laboratory
Experiment 2

Objective:
To familiarize ARM Cortex M0+ assembly programming using Keil software development toolset and FRDM- KL25Z development board.

Background:
Microprocessors have become so integrated to human life that they are used almost everywhere such as in computers, communication devices, consumer electronics, medical instruments, vehicles etc. Most applications use System on Chips (SoC's) or Microcontroller Units (MCUs) which integrate majority of the processing and peripheral functions into a single integrated package.

A wide variety of processor architectures are available to address different applications. One of the dominant architectures that is available today is the ARM family of processors. Therefore it is an attractive architecture to be used as a learning platform to explore microprocessor behaviors and their use. The processor used here is the ARM Cortex M0+ which is based on ARMv6-M architecture. The Cortex-M0+ processor is built on a highly area and power optimized 32-bit processor core, with a 2-stage pipeline von Neumann architecture. The processor delivers exceptional energy efficiency through a small but powerful instruction set and extensively optimized design.

Since the processors fundamentally run architecture specific machine languages to operate, in order to completely comprehend the microprocessor operations, observing and manipulating the behavior of the processors at the machine language level is necessary. But writing programs directly in machine language can be very difficult, hence assembly languages are used to provide a more friendly way to generate machine language. Therefore knowledge in assembly language is very essential.

In this lab, Keil microcontroller development kit (MDK) featuring µVision IDE will be used as the software development tool set for the Cortex M0+ processor. Keil µVision IDE combines project management, run-time environment, build facilities, source code editing, and program debugging in a single powerful environment.

The hardware experiment platform is the NXP (formerly Freescale) Freedom Development Board (FRDM-KL25Z). The KL25Z is a Cortex-M0+ microcontroller for embedded applications. The FRDM-KL25Z comes with a 32-bit ARM Cortex-M0+ core running at 48MHz, 128KB FLASH, 16KB RAM and lots of interfaces including USB Device, SPI, I2C, ADC, DAC, PWM, Touch Sensor, GPIOs, 3-axis Accelerometer, PWM Controlled RGB LED etc. It also includes a built-in USB FLASH programmer.

Useful Resources:

1. **Cortex M0+ Generic User's Guide:**
   This book is a generic user guide for devices that implement the ARM Cortex-M0+ processor.
   http://infocenter.arm.com/help/topic/com.arm.doc.dui0662b/DUI0662B_cortex_m0p_r0p1_dgug.pdf

2. **Cortex M0+ Technical Reference Manual:**
   This document describes the functionality and the effects of functional options on the behavior of the Cortex M0+ processor.
   http://infocenter.arm.com/help/topic/com.arm.doc.ddi0484b/DDI0484B_cortex_m0p_r0p0_trm.pdf

3. **FRDM KL25Z User's Manual**
   This document provides a hardware level overview of FRDM KL25Z development board.
   https://www.nxp.com/support/developer-resources/evaluation-and-development-boards/freedom-development-boards/mcu-boards/freedom-development-platform-for-kinetis-kl14-kl15-kl24-kl25-mcus:FRDM-KL25Z?&tab=Documentation_Tab&linkline=Users-Guide

4. **KL25Z Sub-family Reference manual**
   This document provides details on the KL25 series microcontroller control registers, IO assignments, etc.
   https://www.nxp.com/docs/en/reference-manual/KL25P80M48SF0RM.pdf

Exercises:

For exercises #1 to #5, execute the given assembly codes in Keil and explain what the programs do (Give comments for each line of the code). Clearly indicate how the different registers change.

Exercise #1:

```
;Provide a title for the program mentioning its purpose
equate_val EQU 0xAAAAAAAA
      AREA code_area, CODE, READONLY
      EXPORT __main
__main
      LDR R1,const_val
      LDR R0,=const_val
      LDR R1,[R0]
      LDRH R1,[R0]
      LDRB R1,[R0]
      LDR R0,=equate_val
      LDR R1,=const_val
      STR R0,[R1]
```

```
      MOV R2,R0
      MOVS R2,#0
      B __main

const_val DCD 0X55555555
      AREA data_area, DATA, READWRITE
      END
```

Exercise #2:

```
;Provide a title for the program mentioning its purpose
value1 EQU 50
value2 EQU 123
value3 EQU 0xFFFFFFF0
      AREA code_area, CODE, READONLY
      EXPORT __main
__main
      MOVS R0,#0
      LDR R1,=value1
      LDR R2,=value2
      LDR R3,=value3
      MSR APSR,R0
      ADDS R2,R1
      SUBS R2,R1
      ADDS R3,R1
      SUBS R3,R1
      MSR APSR,R0
      ADD R3,R1
      CMP R1,R2
      CMP R2,R1
      CMP R1,R1
      CMP R1,#0x40
      CMP R2,#0x40
      CMN R1,R3
      CMN R1,R3
      B __main

      AREA data_area, DATA, READWRITE
      END
```

Exercise #3:

```
;Provide a title for the program mentioning its purpose
      AREA code_area, CODE, READONLY
      EXPORT __main
__main
```

```
pos1
      B pos3
pos2
      B pos4
pos3
      B pos2
pos4
      B pos1

      AREA data_area, DATA, READWRITE
      END
```

Exercise #4:

```
;Provide a title for the program mentioning its purpose
AREA code_area, CODE, READONLY
      EXPORT __main
__main
rst_cnt
      MOVS R0,#3
dec_cnt
      SUBS R0,#1
      BNE dec_cnt
      B rst_cnt

      AREA data_area, DATA, READWRITE
      END
```

Exercise #5:

```
;Provide a title for the program mentioning its purpose
     AREA code_area, CODE, READONLY
      EXPORT __main
__main ;
loop
      LDR R0,=value1
      BL change_value
      LDR R0,=value2
      BL change_value
      B loop

change_value
      PUSH {R1,R2}
      LDR R1,[R0]
      ADDS R1,#1
      EORS R1,R0
      MOVS R2,#0xFF
```

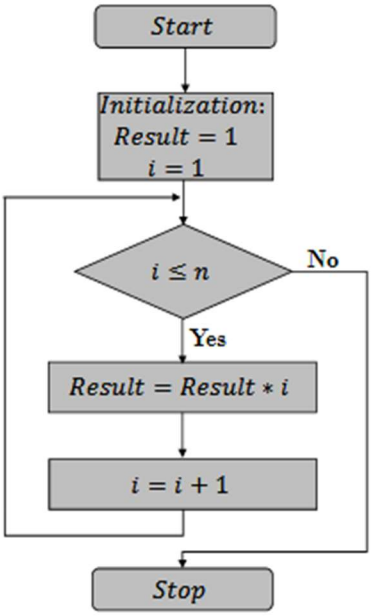
```
        ANDS R1,R2
        STR R1,[R0]
        POP {R1,R2}
        BX LR

        AREA data_area, DATA, READWRITE
value1  SPACE 4
value2  SPACE 4
        END
```

For exercises #6 to #8, write and simulate the assembly codes in Keil and explain the implemented codes by providing comments wherever necessary. Draw your conclusions on how the values are stored in memory and how the registers change.

Exercise #6:

With the help of the flow chart given below, find the factorial of a non-negative number N (N!=1x2x3……x(N-2)x(N-1)x(N).



Exercise #7:

Find out the parity bit value and the number of 1's in a given value n. Assume ODD parity if even number of ones and EVEN parity if odd number of ones.

Exercise #8:

With the help of the flow chart given below, find the maximum value and its location in the given array. After debugging using simulator, also load the code onto the FRDM-KL25Z board and illustrate the debug operation.

Make the following assumptions:

- Linear search
- Given an array[ ] - can contain signed numbers
- r3= Array_Size
- r0= Max_Value
- r2 = Counter
- r1= Array_Index
- Each number in the array is a word size
- Array DCD 1,2,3,-1, …
- Size DCD 4