ECE253 – Digital and Computer Systems

University of Toronto

Lab 10: Interrupts

The goal of this lab exercise is to continue to gain familiarity with assembly language programming, to learn more about subroutines, and to communicate with an I/O device by using interrupts.

Preparation and In-Lab Marks

Preparation marks: Write the required assembly language programs. Print them and hand to a TA when you enter the lab. **Note:** your programs have to be commented well enough to enable another person to understand them.
In-lab marks: You are required to demonstrate the lab exercise functionality to a TA. You should also demonstrate that you can set a breakpoint in an interrupt service routine.

What to Do
In Lab 9 you used the timer to create a delay, and used polled I/O to synchronize with the timer. Also, you used polled I/O to check when KEY_3 was pressed. In this part of the exercise, you are to create the same application, but relying entirely on interrupts instead of polled I/O. Interrupts have to be generated by both the timer and KEY port.

An outline of the main program that you need to use for this part is shown below.

```
        .text
        .global  _start
_start:
        . . . initialize the IRQ stack pointer . . .
        . . . initialize the SVC stack pointer . . .

        BL       CONFIG_GIC                  // configure the ARM generic interrupt controller
        BL       CONFIG_PRIV_TIMER           // configure the MPCore private timer
        BL       CONFIG_KEYS                 // configure the pushbutton KEYs

        . . . enable ARM processor interrupts . . .

        LDR      R6, =0xFF200000             // red LED base address
MAIN:
        LDR      R4, LEDR_PATTERN            // LEDR pattern; modified by timer ISR
        STR      R4, [R6]                    // write to red LEDs

        B        MAIN

/* Configure the MPCore private timer to create interrupts every 0.25 second */
CONFIG_PRIV_TIMER:
        LDR      R0, =0xFFFEC600             // Timer base address
        . . . code not shown
        MOV      PC, LR                      // return

/* Configure the KEYS to generate an interrupt */
CONFIG_KEYS:
        LDR      R0, =0xFF200050             // KEYs base address
        . . . code not shown
        MOV      PC, LR                      // return

        .global  LEDR_DIRECTION
LEDR_DIRECTION:
        .word    0                           // 0 means moving to centre, 1 means moving to outside

        .global  LEDR_PATTERN
LEDR_PATTERN:
        .word    0x201                       // 1000000001
```

Store your main program in a file, for example *main.s*. Create three other files, called *exceptions.s*, *timer_ISR.s*, and *key_ISR.s*. The *exceptions.s* file should initialize the exception vector table and provide code for all exception handlers. It should also include the subroutine *CONFIG_GIC*, which initializes the ARM Generic Interrupt Controller

The file *timer_ISR.s* is the MPCORE Private Timer interrupt service routine. It has to modify the global variables *LEDR_DIRECTION* and *LEDR_PATTERN* that are declared in the main program above. The first of these variables specifies if the LEDR lights are currently sweeping to the centre or to the outside, and the second variable sets which lights are currently turned on. Finally, the file *KEY_isr.s* is the pushbutton KEY interrupt service routine. It has to stop and *resume* the timer when KEY_3 is pressed.

Skeleton assembly language files are provided on the course Piazza site with the lab writeup. Fill in the missing parts of these files to create your program.