Laboratory Exercise 9

Input/Output in ARM assembly

Revision of November 23, 2021

The purpose of this lab is to:

1. learn how to interact with memory-mapped devices
2. learn how to use polling to detect external events

1 Preparation

Please refer to the Lab 7 handout for resources related to the ARM processor and the cpulator simulator.

Note: This lab will be marked entirely in-person during your scheduled practical session via a demo to your TA. There is no automarker for this lab.

2 Part I

Write an assembly-language program that turns on two LEDR lights at a time on the DE1-SoC board. First, the lights LEDR9 and LEDR0 should be on, then LEDR8 and LEDR1, then LEDR7 and LEDR2, and so on. When you get to LEDR5 and LEDR4 being on, the direction should be reversed. Only two LEDR lights are ever on at one time. The effect should be a two lights sweeping first towards the centre, then outwards towards the edges, and so on.

Use a delay so that lights move every 0.25 seconds. To implement the delay, use the MPCORE Private Timer described in the lectures (and also in Section 2.4.1 of the Intel documentation on the Quercus website.) When KEY3 is pressed and released, the sweeping motion should stop. Pressing and releasing KEY3 again should restart the motion of the LEDR lights from where they left off. When the restart occurs, the current LEDRs should remain on for a delay of 0.25s before continuing their motion.

To synchronize with the timer device in your main program, use polled I/O by repeatedly reading the F bit in the timer's status register. Also use polled I/O to deal with the push button KEY3.

1. All necessary code should be contained in one file: part1.s.
2. You should test your code using the CPUlator simulator or using the DE1-SoC board.
3. There is no tester provided for Part 1.

3 Part II – Optional for bonus marks

In Part I, 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 KEY3 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. Both the timer and the KEY need to be handled through Interrupts.

You are given template code on Quercus called part2_template.s. There is a LOT of code here. At the top of the file you will see a number of .equ lines. These are similar to #define in C and can be used as a convenience for you in your code if you wish.

You will need to fill in the missing parts including:

- Provide code for all exception handlers. Specifically look for "FILL IN CODE HERE"
- Complete the code for the PRIV_TIMER_ISR. It has to modify the global variables LEDR_DIRECTION and LEDR_PATTERN that are declared in the main program below. 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.
- Complete the code for KEY_ISR. This is the push button KEY interrupt service routine. It has to stop and resume the timer when KEY3 is pressed.
- Test your code in the cpulator simulator or the DE1-SoC board. No autotester is provided.
- You should comment your code so the TA can understand what you are doing.

An outline of the main (_start) program that you need to use for this lab 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
```

4 Submission

There is no submission for this lab. You will demo your working code for a TA during your practical session. TA will test the functionality and ask you questions about your code.

Part II is optional and can be completed and demo'd for to 1% bonus added to the final course mark.