Lab 5: Interfacing Display Peripheral with Microcontroller

Goals

1. Understanding of the fundamental concepts of Assembly Language
2. Able to write Simple Assembly program using ARMv7 ISA to interface with simple I/O devices (Switch, Push Button)
3. Develop familiarity with a I/O system of a Computer System Simulator

Peripheral Device:

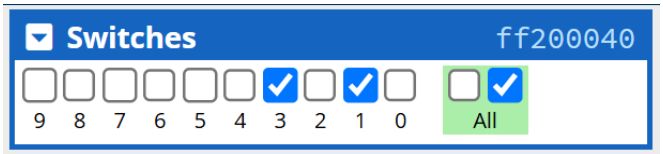
In CPUlator, peripherals are simulated hardware components that allow you to interact with the ARM system by performing input/output operations or interacting with memory-mapped hardware devices. The ARM version of CPUlator typically includes peripherals found in embedded systems such as Switches, Push Buttons and etc.

Switches:

From the right-side panel of the CPULator, we can see switches. We use switched to take an input. The input from the user goes to the registers in the code segment. Previously, we learned that for every peripheral device, we need to use the address of that device. We also learned that to load a value in text segment we use LDR instruction. So, we will use that idea to do code.

From the following picture we can see, each of the switches has two options, on or off. So, the switch will always input a binary value. Suppose, you want to input 10 using switches, you will use 1010 in switch from the rightmost.

A sample code given below:



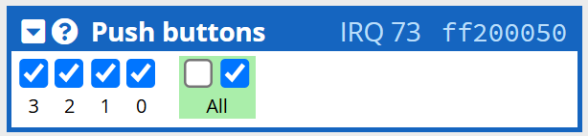
Syntax:

```
.global _start

.equ Switch_add, 0xFF200040

.text

_start:

LDR R0, =Switch_add @ loading address of switches

LDR R1, [R0] @loading value from switches (input)
```

Push Buttons:

It is a button and works the same as switches. So, the syntax for the Push button is also similar to switches. The only difference in the address of the peripheral device. So, we need to be careful when loading the address of the peripheral device.

From the following picture we can see, each of the Push buttons has two options, on or off. So, the PB will always input a binary value. Suppose, you want to input 15 using PBs, you will use 1111 in PB from the rightmost.



Syntax:

```
.global _start
.equ PB_add, 0xFF200050
.text
_start:
LDR R0, =PB_add @ loading address of Push Buttons
LDR R1, [R0] @loading value from push buttons (input)
```

Assembly program example:

```
/*****************************************************************************
This program demonstrates the use of parallel ports in the DE1-SoC Computer
It performs the following:
1. displays the SW switch values on the red lights LEDR
2. displays a rotating pattern on the HEX displays
3. if KEY[3..0] is pressed, uses the SW switches as the pattern
*****************************************************************************/
org 1000 @program start at memory address 1000
.text /* executable code follows */
.global _start
_start:
 MOV R0, #31 // used to rotate a bit pattern: 31 positions to the
```

```
/ right is equivalent to 1 position to the left
LDR R1, =0xFF200000 // base address of LEDR lights
LDR R2, =0xFF200020 // base address of HEX3_HEX0 7-segs
LDR R3, =0xFF200040 // base address of SW switches
LDR R4, =0xFF200050 // base address of KEY pushbuttons
DR R5, HEX_bits // load the initial pattern for the HEX displays
DO_DISPLAY: LDR R6, [R3] // load SW switches
 STR R6, [R1] // write to red LEDs
LDR R7, [R4] // load pushbutton keys
MP R7, #0 // check if any key is presssed
EQ NO_BUTTON
 MOV R5, R6 // copy SW switch values onto HEX displays
AIT:
LDR R7, [R4] // load pushbuttons
MP R7, #0
 BNE WAIT // wait for KEY release
O_BUTTON: STR R5, [R2] // store to HEX3 ... HEX0
ROR R5, R0 // rotate the displayed pattern to the left
DR R6, =50000000 // delay counter
SUB_LOOP: SUBS R6, R6, #1
 BNE SUB_LOOP
 B DO_DISPLAY
HEX_bits:
.word 0x0000000F // initial pattern for the HEX displays
end
```

CSE331L/EEE453L: Microprocessor Interfacing and Embedded System Lab
North South University, Dhaka

Lab Report 5

Lab 5: Interfacing Display Peripheral with Microcontroller

Student Name:

ID: **Section:**

1. You can use ROR instruction to move between seven segment displays? How many ROR instruction you used and why?
2. Write a pseudo algorithm for ARM assembly that can simulate a trafic signal with LEDs. Consider the irst LED as Green, second one as yellow and third one as red.
3. You can use NOP instruction and loop to simulate a delay function. Write a pseudo algorithm that with changing one parameter we can change the delay function. (Hint: you can use .equ pragma).