ECE 118L - MICROCONTROLLER LABORATORY

Lab #14: Memory game In ARM Assembly

Created by: Puya Fard, Aryan Singh, & Michael Peter Avakian

Introduction:

The objective of the project is to create an ARM processor program, using the Seven Segment displays, Pushbuttons and Switches, to play a memory game in which the user will have to guess the numbers displayed on the LEDs and depending on the user's input the game will display if they won the game i.e. if they guessed all the numbers correctly and in sequence or if they lost the game.

Processor Information

The processor used for this project was the 800Mhz Dual Core ARM Cortex-A9 MPCore processor with a 1 GB DDR3 SDRAM (32-bit data bus), 1Gigabit Ethernet PHY with RJ45 connector, 2-port USB Host, normal Type-A USB connector, Micro SD card socket, Accelerometer (I2C interface + interrupt), UART to USB, USB Mini-B connector, Warm reset button and cold reset button, one user button and one user LED, LTC 2x7 expansion header, and this processor was used on Intel's FPGA monitor program. The Altera Cyclone device which has 64 mb SDRAM with a 16- bit data bus. The microprocessor is equipped with 4 push buttons, 10 switches, 10 LEDs along with the 6 seven segment displays. The ARM Cortex A9 is also equipped with Four 50MHz clock sources from the clock generator, 24-bit CD-quality audio CODEC with line-in, line-out, and microphone-in jacks, VGA DAC (8-bit high-speed triple DACs) with VGA-out connector, TV decoder (NTSC/PAL/SECAM) and TV-in connector, PS/2 mouse/keyboard connector, IR receiver and IR emitter, two forty pin expansion header with diode protection, A/D converter, 4-pin SPI interface with FPGA.

Theoretical Background:

To implement the program the statements and commands used were the LDR and STR commands which are used to load and store values in the registers, CMP statement which compares the values in the registers. Along with these commands, certain loops and branches were also used to branch out to different code sections and perform the functions coded in those particular sections. Labels were also used which represent the memory address of a data variable or an assembly.

Data representations were Hexadecimal and binary which are used on the Seven segment display to display the output. The .data assembler directive tells the ARM assembler to start assembling the line after the directive into the .data section of the computer program

Example for LDR and STR:

```
STR    R0, [R5, R1]          ; Store value of R0 into an address equal to
                             ; sum of R5 and R1
LDRSB  R0, [R5, R1, LSL #1]  ; Read byte value from an address equal to
                             ; sum of R5 and two times R1, sign extended it
                             ; to a word value and put it in R0
STR    R0, [R1, R2, LSL #2]  ; Stores R0 to an address equal to sum of R1
                             ; and four times R2.
```

Example for CMP:

The following forms of these instructions are available in Thumb code, and are 16-bit instructions:

`CMP Rn, Rm`

Lo register restriction does not apply.

`CMN Rn, Rm`

`Rn` and `Rm` must both be Lo registers.

`CMP Rn, #imm`

`Rn` must be a Lo register. `imm` range 0-255.

ARM Cortex A9

Procedure:

- Create a new project on the FPGA monitor program and name it Memory_game
- With the help of LDR, STR, and CMP and LOOP instructions create a program that will create a game which is going to be played by a single player.
- Format of this game is: there will be displayed 4 numbers on the 7-segment display which the player needs to keep them in their mind as the game begins. These numbers should be editable by the programmer as desired, so focus on creating a .data section for these numbers.
- The player then needs to enter these numbers using pushbuttons and insert them into the system one by one using switches.
- These numbers should be displayed on the 7-segment display as they are being entered by the player one by one.

- After all the numbers have been entered, the display should show all the entered numbers all at once then decide whether the entered numbers are correct or not with the first set of numbers shown to the players to memorize.
- After the system decides whether win/lose, it will be displayed on the 7-segment display indicating the status by typing it out saying WIN or LOSE.
- These displays should flicker the sign on and off with a slight time delay.

Tips:

1. .Data section could be used to help with this program since there will be needed variables for the pre-entered number sequence.
2. Using Hexadecimal values is easier to implement and run through the program since the 7-segment is constructed by Hex values.
3. Make sure to locate addresses of 7-segment hex display, switches, pushbuttons using addressing map or just inserting the addressing map itself using .include directive.
4. Delays are very helpful while creating this program because there will be a lot of time that the display will need to run through a delay in order to create some real time screening.
5. Make sure to name Loops, or labels correspondingly so it won't get confusing while debugging.

Review Questions:

1. How is the Polling method different from the Interrupt method?
2. How would you be able to use the interrupt method while creating this program. Would it be more efficient to use an interrupt method?