EE 337: Microprocessors Laboratory (Spring 2021)
Indian Institute of Technology Bombay

Project: 20 points Date: March 8, 2021

Whac-A-Mole Game on Pt-51

1. [20 points] In this project, you will be writing a game program for a reaction game. It is inspired by the Whac-A-Mole arcade game `https://en.wikipedia.org/wiki/Whac-A-Mole`. The player will type on a keyboard connected to the Pt-51 board via UART.

   - When the game starts, the LCD displays the string `Get Ready` on the first line for two seconds. Then the following is shown on its two lines.

     ```
     ********
     ********
     ```

     Note that there are 8 `*` characters in each line.

   - After a 3 second delay, an `m` appears in place of one of the 16 `*` characters. For example, the display might look like the following:

     ```
     ********
     ****m***
     ```

     The `m` character represents the mole to be whacked.

   - A player whacks the mole by pressing an alphabet key. The mapping from LCD positions to the alphabet key is shown below.

     ```
     qwertyui
     asdfghjk
     ```

     Note that the keys are taken from the first two rows of a QWERTY keyboard. The player would have to press the `g` key to whack the mole in the example shown above.

     - If the key pressed by the player matches the location of the `m` character, then:
       - The `m` character is replaced by a `*` character.
       - An `m` character appears in a new location on the LCD by replacing one of the `*` characters.
       - Note that the new location should not be the same as the old location of the `m` character.
     - If the key pressed by the player does not match the location of the `m` character, then the LCD contents remains the same.

   - The game proceeds for 10 seconds and after that the number of correct matches is displayed on the first LCD line and the highest score so far is displayed on the second LCD line, for 3 seconds. For example, if the player whacks 11 moles in the current game and has a high score of 18, then the display will look like the following.

     ```
     Score:          11
     High Score:     18
     ```

- The goal of the player is whack as many moles as possible in the 10 seconds.
- To make the game interesting, the successive locations of the moles should appear random.
- We associate a unique integer from 0 to 15 with each of the 16 positions as shown below.

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15

- We can use a linear feedback shift register (LFSR) to generate pseudorandomness. Consider a LFSR with 4 registers (each containing a bit). We can use a single nibble to store the current state of the LFSR. If the current state of the 8 bits is (b_3, b_2, b_1, b_0), the next state is

$$(b_3 \oplus b_0, b_3, b_2, b_1),$$

  where $\oplus$ represents bitwise XOR. With this next-state rule, the LFSR will cycle through all the non-zero 4-bit states with a period of 15. This is an example of a maximal-length LFSR `https://en.wikipedia.org/wiki/Linear-feedback_shift_register`.
- Initialize the LFSR state (b_3, b_2, b_1, b_0) to a non-zero value (this is done only once when the program loads for the first time).
  - Place the first `m` at the location whose binary representation is $b_3b_2b_1b_0$. For example, if $b_3b_2b_1b_0 = 0101$ then place the `m` at location 5 (the sixth location on the first line of the LCD). If $b_3b_2b_1b_0 = 1010$ then place the `m` at location 12 (the fifth location on the second line of the LCD).
  - If the player whacks the mole by pressing the correct key, calculate the next state of the LFSR. Let the next state be $b'_3b'_2b'_1b'_0$.
  - Let B be the decimal integer corresponding to $b_3b_2b_1b_0$ and B' be the decimal integer corresponding to $b'_3b'_2b'_1b'_0$.
  - Place the next `m` at the location $B + B' \bmod 16$. Since B' is the integer representation of the LFSR state, it will be non-zero. So the next `m` location will be different from the previous location.
  - If the player whacks the mole again by pressing the correct key, calculate the next state of the LFSR and use it to once again calculate the next `m` location.
  - And so on, until the 10 seconds of the game have passed.
- Preserve the last state of the LFSR for the next game.