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

Project: 20 points Date: March 8, 2021

Tetris Game on Pt-51

1. [20 points] In this project, you will be writing a Tetris game `https://en.wikipedia.org/wiki/Tetris`. The player will type on a keyboard connected to the Pt-51 board via UART.

   - When the game starts, the LCD is empty.
   - Then a tile starts moving to the left with a delay of 1 second between movements. The tiles will be made of `*` characters. For example, a tile may appear as shown in the frame sequence below.
     - At $t = 1$ second
       ```
                          * *
       
       ```
     - At $t = 2$ seconds
       ```
                       * *
       
       ```
     - At $t = 3$ seconds
       ```
                      * *
       
       ```
   - To move the tiles up and down (if possible), the player types characters `q` and `a`. To rotate a tile, the player types the `r` character. The position or orientation of the tile should be updated immediately after the player types `q`, `a`, or `r`.
   - There are three types of tiles as shown below.
     ```
     * *     *       *
             * *
     ```
   - The rotation of the tile with a single `*` character has no effect.
   - The rotation of the tile with two `*` characters switches the tile state between the following two configurations.
     ```
     * *       *
               *
     ```
   - The rotation of the tile with three `*` characters switches the tile state between the following four configurations. Note that the rotation is clockwise.
     ```
       *     *       * *     * *
     * *     * *     *         *
     ```
   - If an up or down move will cause the tile characters to move out of the LCD, then such a move is not allowed. In such a scenario, the character typed by the player is ignored.
     - Suppose the state of the LCD is the following.
       ```
                      * *
       
       ```
       If the player types the `q` character, it is ignored. If the player types the `a` character, the LCD should be updated to the following.

```
                
           * *  
```

Now if the player types the `r` character, the LCD should be updated to the following.

```
            *   
            *   
```

Now if the player types the `r` character, the LCD should be updated to the following.

- The tile keeps moving to the left until it reaches the end of the LCD or it cannot move further because other tiles are blocking their movement.
- After a tile stops moving, the next tile appears on the right hand side of the LCD.
- The game ends when no more tiles can enter the LCD without getting blocked.
- When the game ends, the player's score is the number of seconds she was able to play the game. The player's score 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 has played for 20 seconds in the current game and has a high score of 30, then the display will look like the following.

```
Score:                20
High Score:           30
```

- The game restarts after the 3 seconds of score display.
- To make the game interesting, the sequence of tiles should appear random.
- 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 4 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`.
- Associate the integers 0, 1, 2 with each of the 3 tile configurations.
- 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).
  - Calculate the remainder when the integer corresponding to $b_3b_2b_1b_0$ is divided by 3.
  - Choose the tile whose corresponding integer is this remainder value as the first tile.
  - Calculate the next state of the LFSR. Let it be (b'_3, b'_2, b'_1, b'_0).
  - Calculate the remainder when the integer corresponding to $b'_3b'_2b'_1b'_0$ is divided by 3.

  - Choose the tile whose corresponding integer is this remainder value as the second tile.
  - And so on, until the game ends.
- Preserve the last state of the LFSR for the next game.