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

Project: 20 points Date: March 8, 2021

Obstacle Avoidance Game on Pt-51

1. [20 points] In this project, you will be writing a obstacle avoidance game. It is inspired by the Dinosaur Game in Google Chrome `https://en.wikipedia.org/wiki/Dinosaur_Game`. The player will type on a keyboard connected to the Pt-51 board via UART.

   - When the game starts, the LCD displays the character `D` on the first row and third column as shown below. There are no obstacles at the start of the game.

     ```
       D
     ```

   - A sequence of obstacles start moving to the left with a delay of 1 second between movements. The obstacles will be represented using `*` characters. For example, an obstacle may appear as shown in the frame sequence below.
     - At $t = 1$ second

       ```
         D                                *
       ```

     - At $t = 2$ seconds

       ```
         D                                 *
       ```

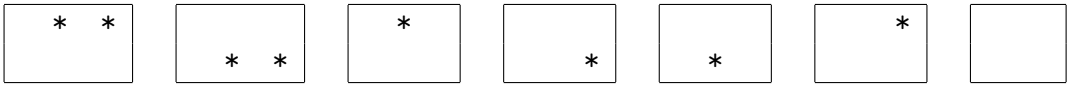
     - At $t = 3$ seconds

       ```
         D                                  *
       ```

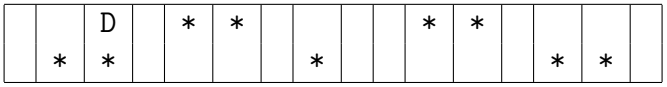
   - To avoid the obstacles, the player types characters `q` and `a` to move the `D` character up or down. The `D` character always remains in the third colum. The position of the `D` character should be updated immediately after the player types a `q` or an `a`.
   - If the player types `q` when the `D` is in the first row or if she types `a` when the `D` is in the second row, these keystrokes are ignored.
   - The game ends when the `D` character gets hit by an obstacle, i.e. the position of the `D` character coincides with the position of an obstacle character.
   - 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 obstacles should appear random. At the same time, the obstacle sequence should allow the player to escape.
   - We consider 7 possible obstacle configurations each of which occupies 2 rows and 3 columns, as shown below. Note that the first column in each configuration is empty to allow escape. The last configuration is empty without any `*` characters.



- One of the above configurations will enter the LCD from the left hand side, one column at a time after each second. After the configuration has occupied the last three columns of the LCD, another configuration will begin appearing. At some point in the game, the LCD might look like the following where the lines have been drawn between columns for clarity.



- We can use a linear feedback shift register (LFSR) to generate pseudorandomness. Consider a LFSR with 3 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 3 bits is (b_2, b_1, b_0), the next state is

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

  where $\oplus$ represents bitwise XOR. With this next-state rule, the LFSR will cycle through all the non-zero 3-bit states with a period of 7. This is an example of a maximal-length LFSR `https://en.wikipedia.org/wiki/Linear-feedback_shift_register`.

- Associate a unique 3-bit sequence with each of the 7 obstacle configurations.
- Initialize the LFSR state (b_2, b_1, b_0) to a non-zero value (this is done only once when the program loads for the first time).
  - Choose the first obstacle configuration to be the one associated with (b_2, b_1, b_0).
  - Calculate the next state of the LFSR. Let it be (b'_2, b'_1, b'_0).
  - Choose the second obstacle configuration to be the one associated with (b'_2, b'_1, b'_0).
  - And so on, until the game ends.
- Preserve the last state of the LFSR for the next game.