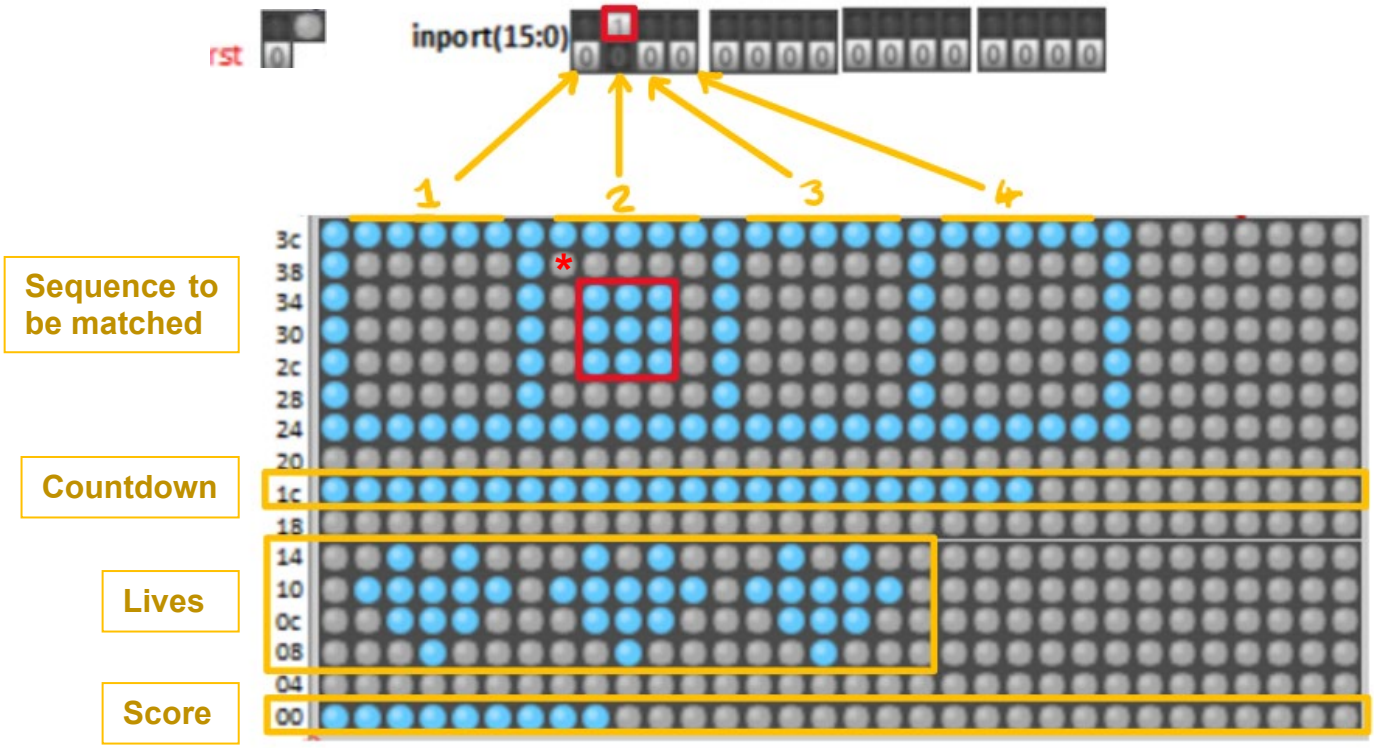
EE451-Assignment 5:
Sequence Matching Game | Spec & Instructions

Paul Kielty & Emily Busby

1 - Display:



In this document any highlighted words indicate a reference to elements on this diagram.

Sequence to be matched: Displays the pseudo-randomly generated sequence that the player is required to match in inport(15:12). In this example, elements 1, 3, & 4 are low and 2 is high*. This is correctly matched in the state of inport(15:12).

Countdown: A line across the display that shortens as LEDs are de-asserted with regular “ticks”, starting from the right-hand side. The countdown is finished when the leftmost bit is de-asserted. This loops continuously through the running of the game.

Lives: Shows a number of hearts corresponding to how many lives the player has remaining, initially 3.

Score: Increments one LED at a time from left to right whenever the player scores a point.

2 – How to Play:

1. Toggling reset (rst) high to low at any point will clear the display and restart the game.

2. When the game starts, there will be one initial countdown with a blank sequence to be matched. This is for the player to get ready.

3. When this initial countdown finishes, the first sequence to be matched will be generated and another countdown is started.

4. All 4 elements in this sequence need to be matched with their corresponding inport(15:12) switch before the current countdown reaches 0.

5. If the sequence is matched correctly, the score is increased by 1 and the next countdown is sped up slightly
.
6. If the sequence is **not** matched correctly, the player loses one of their lives (one heart is removed). They are then given a grace period of one countdown before the next sequence is generated.

7. Steps 3-6 are looped continuously until one of two conditions are met:

    7.1 If the player loses all of their lives, the game ends and their final score is held. As another indicator of this state, the value “DEAD” is written to the first 4 bytes of register x1.

    7.2 If the player manages to fill up the score line (i.e. 32 points) before running out of lives, they win the game and are given a “WINNER!” message on the screen. Good luck!

3 - Primary Functions:

Setup:

This is the first code section run whenever the program is started. It preforms necessary once-off tasks, such as:

- Initialises registers where required.
- Clears the LED array memory (to clean up previous runs of the game).
- Draws the frame on the LED array where the randomly generated sequences are shown.
- Calls one initial countdown to give the player time

Countdown:

This function draws the countdown line on the display and de-asserts 2 LEDs at a time (this is one “tick”) from right to left. Initially one full countdown takes approximately 3 seconds, but this speeds up as the player’s score increases. (Note: At the current running speed Vicilogic cannot consistently show each “tick” two LED’s at a time, however the overall countdown time is sufficiently consistent.)

GenerateSequence:
Based on the current tick delay, peripheral counter value, inport value, and the previously generated sequence, generates a pseudo-random sequence and saves it to the frame on the LED array. Also, in a separate register, saves the inport value required by the player to match this sequence.

CheckCorrectInput:
Compares the value generated by generateSequence to the inport set by the player. If matching, calls increaseScoreAndDifficulty, if not matching, calls removeOneLife.

IncreaseScoreAndDifficulty:
Increments the score of the player by one, and then applies a fixed decrement to the countdown “tick” delay. This gives the player less time for the next countdown, making the game more difficult as they play better.

RemoveOneLife:
Decrements the player’s number of lives remaining by 1 and removes one heart from the display. After this, if the player still has any lives remaining, a “pity countdown” is given. Meaning an extra countdown is called before the next sequence is generated, so the player has time to catch up after missing an input. This was deemed necessary as in testing it was seen that the player would often lose all 3 lives back to back as a new sequence would appear while they were correcting their previous mistake.

MainLoop:
This is the primary loop that calls generateSequence, countdown, and checkInput respectively. Additionally, in each loop the player’s score is checked to see if the game should enter the “win” state, and then their number of lives remaining is checked to see if it should enter the “lose” state. If the win state is entered, a “WINNER!” message is displayed on the LED array. If in the lose state, “DEAD” is written to the upper 4 bytes of register 1. In both cases the player’s final score is held on the display and the game is stopped until a reset.

4 - Status:
Referring back to the specification submitted at the beginning of assignment 5, the current implementation of the game meets the requirements in full. The only changes were redesigning the display to make the required inputs clearer, and the addition of a win state when the player reaches a certain score (this wasn’t outlined in the initial spec).

All of the functions have been implemented and tested thoroughly, and no bugs have been found in their operation. The only small snags were found in the countdown, specifically in that Vicilogic does not refresh the display fast enough to accurately show each 2 LED “tick”, and the time taken for each delay count to be decremented to 0 can be inconsistent. Note that neither of these points are a result of the code.