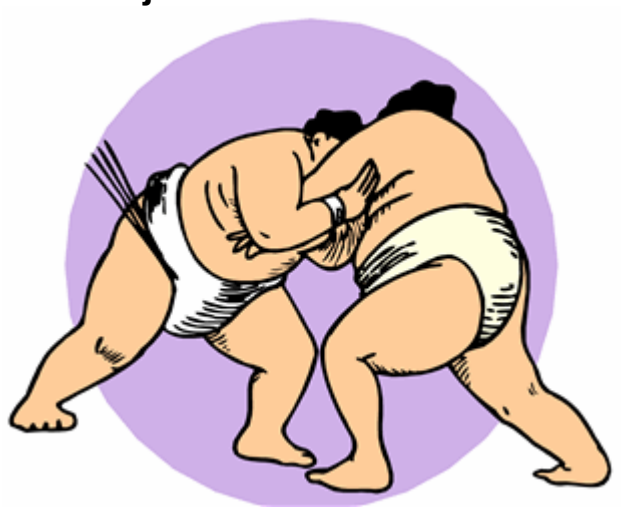
Department of Electronics Engineering

Project 1 – Sumo Wrestler



Introduction.

In this project, you will design a two-player game using the bar graph and two push buttons. Each player will control one of two sumo wrestlers and each will try to push the other out of the ring.

As the match goes on, the wrestlers periodically shove each other apart, and the first to regain his balance is able to push the other a little closer to the edge of the ring.

Using the hardware you have already developed for Lab 3, write an assembly program that meets the requirements on the next page. Document your code well. Write a file header that briefly describes your program. Each subroutine should have a header that describes its inputs, outputs and side effects. There should be no unexplained “magic” numbers, and 40% or more of the lines of code should either have or be a comment.

The project must be demonstrated to your lab instructor, who will verify that it meets the requirements during your lab section. A copy of your assembly code is due during the following class period. Due dates will be announced in class.

Requirements.

The requirements for this project are as follows:

1. The display shall consist of a 10-LED bar graph mounted horizontally. The LEDs shall be numbered from 1 to 10, starting from the right.
2. There shall be 2 buttons, each in the proximity of a different end of the bar graph. The button on the left is used by player 1 and the button on the right is used by player 2.
3. Three DIP switches shall be used to configure the starting position of the players. The three switches shall be interpreted as a 3-bit binary number, *N*, and shall select one of seven possible starting positions. (*N* = 0 is invalid.)
4. The buttons shall be sampled with a period of 10ms (milliseconds).
5. After the system is reset, LED numbers *N*+2 and *N*+1 shall be lit. These LEDs represent players 1 and 2, respectively.
6. One second after reset, the leftmost lit LED shall move one spot to the left and the rightmost lit LED shall move one spot to the right. This event starts the move.
7. After the move starts, each player races to press and release his button. As soon as a button is pressed, the corresponding player's lit LED moves back to its prior position (except as described in line 8, below).
8. If the first player to press his button releases it before his opponent presses his button (and moves his lit LED), the quicker player's lit LED shall move again and become adjacent to his opponent's lit LED.
9. If, after the move concludes, either LED 1 or LED 10 is lit, the game is over and the LEDs shall not change until the system is reset.
10. Otherwise, at some random time at least 1/2 second but no more than 1 second after the move concludes, the next move shall start by leftmost lit LED moving one spot to the left and the rightmost lit LED moving one spot to the right.

Hints:

Random delays are easy if you key them off human interaction. Have the function that does the 10 ms delay increment a modulo 50 counter, and whenever either button is pressed, add 50 to that counter and store the result in a variable (i.e. rand_num). Then when you need a 0.5-1.0 second delay, call the 10 ms delay (rand_num) times.

To make a modulo 50 counter, put this code in before or after the delay:

```
        djnz R7,carry_on
        mov R7,#50
carry_on:
```