# Цифровые платформы.

# Групповой проект.

# “Door game”

### Overview

Ваше задание на этот проект - это реализовать игру «Виселица» в Logisim.

* Вам предстоит реализовать 8 дисплеев 9х9, а также клавиатуру для ввода букв. Вы будете использовать дисплей, чтобы показывать количество оставшихся жизней у игрока.
* Вы так же напишите программу, которая будет обрабатывать вводимую букву и проверять её наличие в слове.

### Basic system

Ваша программа должна реализовывать игровой процесс виселицы. Она имеет очевидную стратегию. Вы должны сделать так, чтобы она:

* Обрабатывала ввод букв с клавиатуры
* Определяла, в каких местах в слове стоит эта буква и открывала соответствующие матрицы
* При вводе буквы, отсутствующей в слове, уменьшала количество жизней
* Определяла, когда игра закончена
* Определяла результат игры (победа, поражение)

### A basic hangman game

Это всем известная игра. В неё играют на поле с несколькими линиями, являющимися индикаторами букв. Правила игры в этом проекте остаются теми же:

1. В каждый свой ход игрок выбирает любую букву алфавита
2. Игрок видит на дисплее положение “открытых” букв в слове
3. При провальной попытке он теряет жизнь

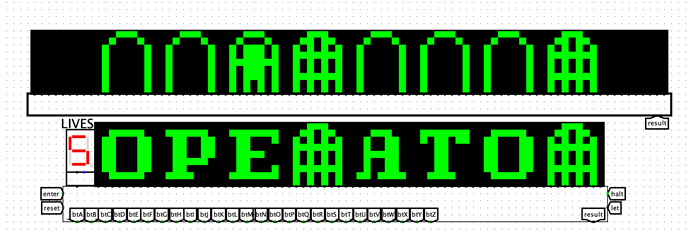
Это все правила данной игры.

### Hardware

Аппаратная часть этого проекта состоит из процессора Cdm-8 с памятью по Гарвардской системе, а также из системы контроля матриц.

### Game pad

Ниже представлен прототип панели игрока. Каждая матрица отвечает за отображение отдельной буквы в слове. Другая группа матриц выводит двери и продвижение человечка.



### Keyboard

Ввод данных должен осуществляться с помощью нажатия буквы.

Вам также необходимо правильно считать сигнал и передать его на другие модули. Затем буква будет скопирована в память.

Каждая буква соответствует определённой константе, которую и следует сохранять.

### Symbol output

Вывод символов осуществляется с помощью мультиплексора, в котором сохранены константы, соответствующие определённому рисунку на матрице. При угадывании буквы, на мультиплексоры поступает нужный адрес, иначе на выход подаются константы для закрытых дверей.

### Where does the data come from?

Данные поступают из регистра r0 Cdm-8. Поскольку регистр 8-разрядный, возникает вопрос, о том что из себя представляет каждый бит. Значение регистра означает следующее:

| bit value | meaning |
| --- | --- |
| 0 | буква, стоящая на позиции этого бита не угадана |
| 1 | буква была угадана |

### Indicating that the game is over

Указанием на то, что игра окончена служит максимальное значение r0, или сигнал, поступивший от модуля, который следит за количеством жизней.

Поскольку у игры 2 потенциальный исхода, они рассматриваются отдельно.

Игрок проиграл: такой исход возможен только в том случае, когда счётчик жизней опуститься до нуля. Для этого стоит ввести модуль подсчёта жизней. В нём сравниваются предыдущее и текущее значения регистра r0, их совпадение означает неудачную попытку игрока в угадывании слова, счётчик снижается. При достижении определённого числа неудачных попыток, данный модуль может подать сигнал о завершении игры.

Игрок выиграл: такое может произойти только в том случае, когда каждая буква будет угадана, а значит регистр r0 должен принять значение 0b11111111.

### Software

### The data structure

Загадываемое слово может быть представлено как последовательность букв, закодированных определённым образом, например как числа от одного до двадцати шести. Каждую букву следует последовательно разместить в памяти данных, начиная с адреса 0 и заканчивая 7. Каждую вводимую пользователем букву можно разместить по определённому вами адресу.

### Main loop

Основная программа должна сравнивать букву, вводимую пользователем, последовательно с каждой буквой в загаданном слове.

Ваша программа должна выглядеть примерно так:

1. Исследуйте слово, загружая каждую букву в определённый регистр, пока не встретите конец слова.
2. Сравнивайте данную букву с буквой пользователя
3. При совпадении меняйте маску, служащую индикатором отгаданных и не отгаданных букв