**Conway’s Game of Life Project Specification**

This document outlines the requirements for a project implementing Conway’s Game of Life.

Conway’s Game of Life is simulated on a toroidal field measuring 32 by 32 cells, which can be either alive or dead. The "birth" and "survival" of each cell are determined by the state of its 8 neighbors, according to user-defined rules. The rules for birth and survival are set by the user in advance.

The game has been described countless times in all sorts of materials and has representations in various variations:

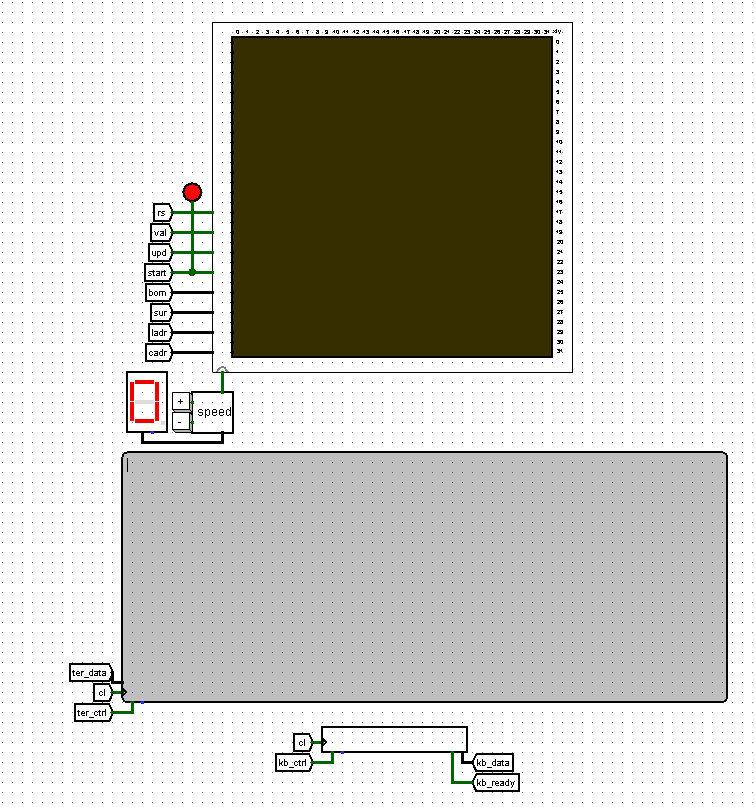
<https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life>

<http://www.dataved.ru/2012/08/game-of-life.html>

<https://playgameoflife.com/>

It should be noted that Conway’s Game of Life exists in a great many languages, and there are also ready implementations of this project in Logisim. Excessive copying or copying of solutions will be punished by disqualification.

Hardware Implementation:



Software Implementation:

The program must process entered commands by calling a separate file with command processing functions, outputting corresponding messages to the terminal, and sending signals to circuit sections for subsequent game simulation.

Program Project:

asect 0

\_start:

# code here

\_main:

# here should be a call to the parse function,

# which calls the necessary handler from parser.c

# and correctly handles errors

\_kb\_isr:

# code here

\_print:

# code here

Necessary changes compared to the original game implementation:

1. **Toroidal field of 32 by 32.** Edge cells are considered neighbors of cells on the opposite edge.
2. **CdM-16 processor.** The project used the CdM-16 processor for full and fast processing of commands and arising errors.
3. **Position editing.** The user can edit the value of any cell at any point during the game simulation.
4. **Rule editor.** The user can change the rules for cell birth and survival at any point during the game simulation.
5. **Command processing.** Almost all user interaction occurs via commands entered using the keyboard. Through them, you can start and stop the game, clear the field, set a glider and other pre-prepared patterns, fill a rectangular area of the field, change the value of a specific cell, and change the rules.
6. **Input validation.** The software part performs validation of user input: number of arguments, string length, command existence, and adherence of argument values to allowed ranges.