# Game of Life

* The place of action of the game is a plane marked up into cells, which can be unlimited, limited or closed.
* Each cell on this surface has eight neighbors surrounding it, and can be in two states: being "alive" (filled) or "dead" (empty).
* The distribution of living cells at the beginning of the game is called the first generation. Each next generation is calculated based on the previous one according to standard rules:
  + Neighbours are nearby cells (like a king in chess).
  + 3 neighbours — cell can be born.
  + 2 or 3 neighbours — cell continues to live.
  + Otherwise, the cell dies.

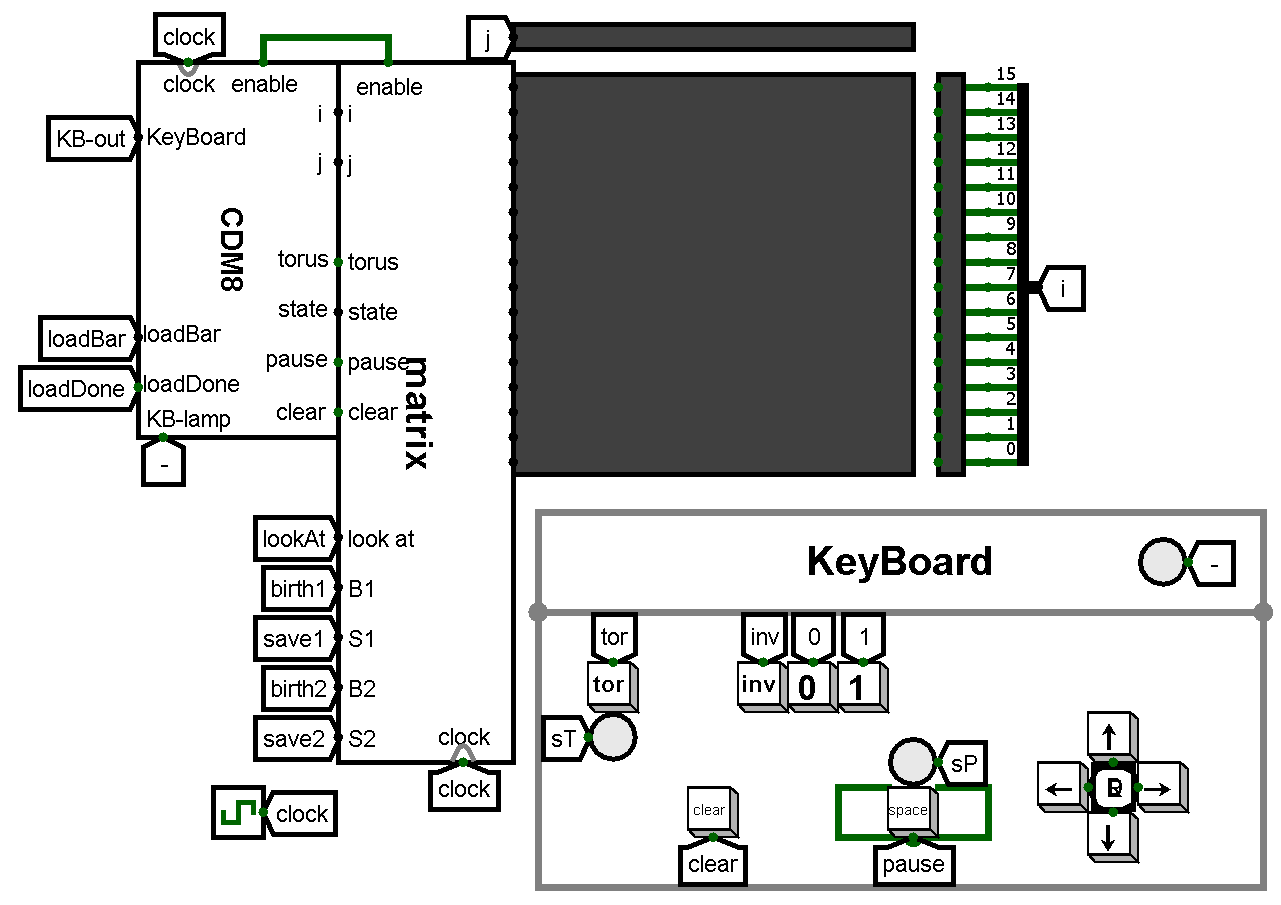
The game is very popular, described in Wikipedia (Russian and English):

* [https://ru.wikipedia.org/wiki/Игра\_«Жизнь»](https://ru.wikipedia.org/wiki/%D0%98%D0%B3%D1%80%D0%B0_%C2%AB%D0%96%D0%B8%D0%B7%D0%BD%D1%8C%C2%BB)
* <https://en.wikipedia.org/wiki/Conway's_Game_of_Life>
* A number of sites are dedicated to the game, for example <https://conwaylife.com/>

The recommended implementation approach consists of a 16x16 matrix (display), a keyboard and a processor that provides first-generation editing. You will need to add several buttons to edit, start and stop the game.

On the Internet, you can find various prototypes of the implementation of the Game “Life” in other languages or a fully hardware implementation of the game “Life” in Logisim. Excessive borrowing of solutions from this implementation is punishable by disqualification.

Hardware Design:



Software Design:

**asect** 0

# Implement the game initialization procedure.

**init**:

# write your code here

# Implement a game loop.

**game**:

# write your code here

**halt**

# Implement other procedures necessary for the operation of the chip:

**up**:

# write your code here

**down**:

# write your code here

**right**:

# write your code here

**left**:

# write your code here

**set0**:

# write your code here

**set1**:

# write your code here

**setInv**:

# write your code here

**end**.

Necessary changes from the original implementation of the game:

1. How to handle the edges of the field?

The toroidal field is 16x16. Cells on the edges are considered neighbors of cells on the opposite edge of the field.

1. Editing the initial position.

It is forbidden to edit the initial position with a matrix of buttons in which each button corresponds to one pixel. Therefore, it is necessary to implement a joystick or keyboard to control the cursor, which will edit each cell of the field.

The joystick/ keyboard should have buttons for moving, pausing, clearing, changing the state of the cell (alive or dead) and turning on the toroidal field. If possible, flags can be implemented that reflect the state of the toroidal field and pause buttons.

The team decides on its own how to connect the keyboard to the processor.

1. The rules editor.

It is necessary to implement changes to the rules of the game. The user should be able to change the rules at will.

To implement the rules editor, you can use the Logisim logic scheme: several contacts (“input pins”) that allow the player to change the conditions of the game.