МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»  
(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Высший колледж информатики

Кафедра интеллектуальных систем теплофизики

Направление подготовки 15.03.06 Мехатроника и робототехника

Направленность (профиль): Мехатроника и робототехника

**КУРСОВАЯ РАБОТА**

**Беспалова Сергея Вячеславовича**

**Грищенко Александра Михайловича  
Солопова Ильи Руслановича**

Тема работы:

**ИГРа «RUSH Space»**

|  |  |
| --- | --- |
| **«К защите допущена»** | **Руководитель ВКР** |
| Заведующий кафедрой, | ученая степень, звание |
| ученая степень, звание | должность, место работы в НГУ |
| ………………/………….. | ………………/………... |
| (ФИО) / (подпись) | (ФИО) / (подпись) |
| «……»………………20…г. | «……»………………20…г. |
|  |  |

Новосибирск, 2022

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc103514653)

[1 ПОСТАНОВКА ЗАДАЧИ 4](#_Toc103514654)

[2 АНАЛОГИ 5](#_Toc103514655)

[3 АППАРАТНАЯ ЧАСТЬ 7](#_Toc103514656)

[4 ПРОГРАММНАЯ ЧАСТЬ 10](#_Toc103514657)

[ЗАКЛЮЧЕНИЕ 13](#_Toc103514658)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ 14](#_Toc103514659)

[ПРИЛОЖЕНИЯ 15](#_Toc103514660)

ВВЕДЕНИЕ

По заданию к курсовому проекту по дисциплине «Digital platforms» нужно было спроектировать и создать игру в жанре Shoot 'em up на электронных схемах, использовав процессор CdM-8 и его ассемблерный язык.

Shoot 'em up – это вид видеоигр, в котором управляемый игроком персонаж чаще всего представлен в виде космического корабля или другого транспортного средства, основная цель которого – победить множество врагов, используя стрельбу. Врагами в таких играх, как правило, выступают различные инопланетяне или монстры, атакующие игрока, стреляя в него или как-то иначе. Традиционно в таких шутерах используется вид сверху или сбоку, а для успешного прохождения важна хорошая реакция, чтобы уклоняться от вражеского огня.

В соответствие с поставленной целью необходимо было решить следующие задачи:

* Изучить примеры игр жанра Shoot 'em up (аналоги);
* Изучить и проанализировать информацию о процессоре, его возможностях, командах и инструкциях;
* Определить функциональные требования.

1 ПОСТАНОВКА ЗАДАЧИ

Цель данной курсовой работы – проектирование и создание игры жанра Shoot 'em up на основе электросхемы с включенным в неё 8-битным процессором CdM-8.

Ниже представлены функциональные требования:

1. Управляемое движение игрока;
2. Неуправляемое движение противников;
3. Стрельба игрока;
4. Стрельба монстров;
5. Уничтожение монстров;
6. Управление с клавиатуры (движение игрока, стрельба игрока);
7. Возможность выигрыша;
8. Возможность проигрыша.

Методы решения поставленных задач…

2 АНАЛОГИ

В процессе создания проекта нами были изучены игры жанра Shoot 'em up. Рассмотрим некоторые из них.

1. Японская автоматная аркада «Space Invaders», выпущенная в 1978 году. Её можно назвать одной из первых в данном жанре. Данная игра имеет все заданные в предыдущем пункте функциональные требования, а также дополнительные, например, счётчик жизней и количества набранных очков, звуковые эффекты, бесконечное количество «волн» врагов, защита в виде «бункеров» от выстрелов инопланетян.

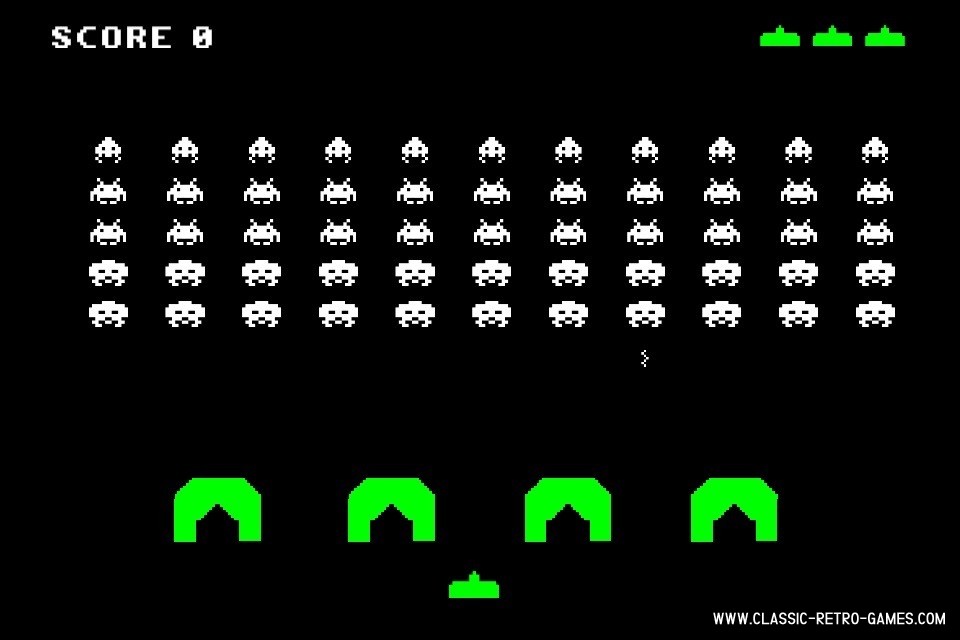


Рисунок 1 – скриншот игрового процесса «Space Invaders»

1. Еще одна японская аркада, выпущенная для игровых автоматов в 1981 году – «Galaga». Её можно назвать улучшенной и более современной версией предыдущей игры. В «Galaga» были созданы новые игровые механики, такие как притягивающий луч инопланетян, способный лишить игрока возможности управления на короткий промежуток времени и одной жизни, пикирование монстров на игрока по различным траекториям. Также это одна из первых игр с цветной RGB-графикой.

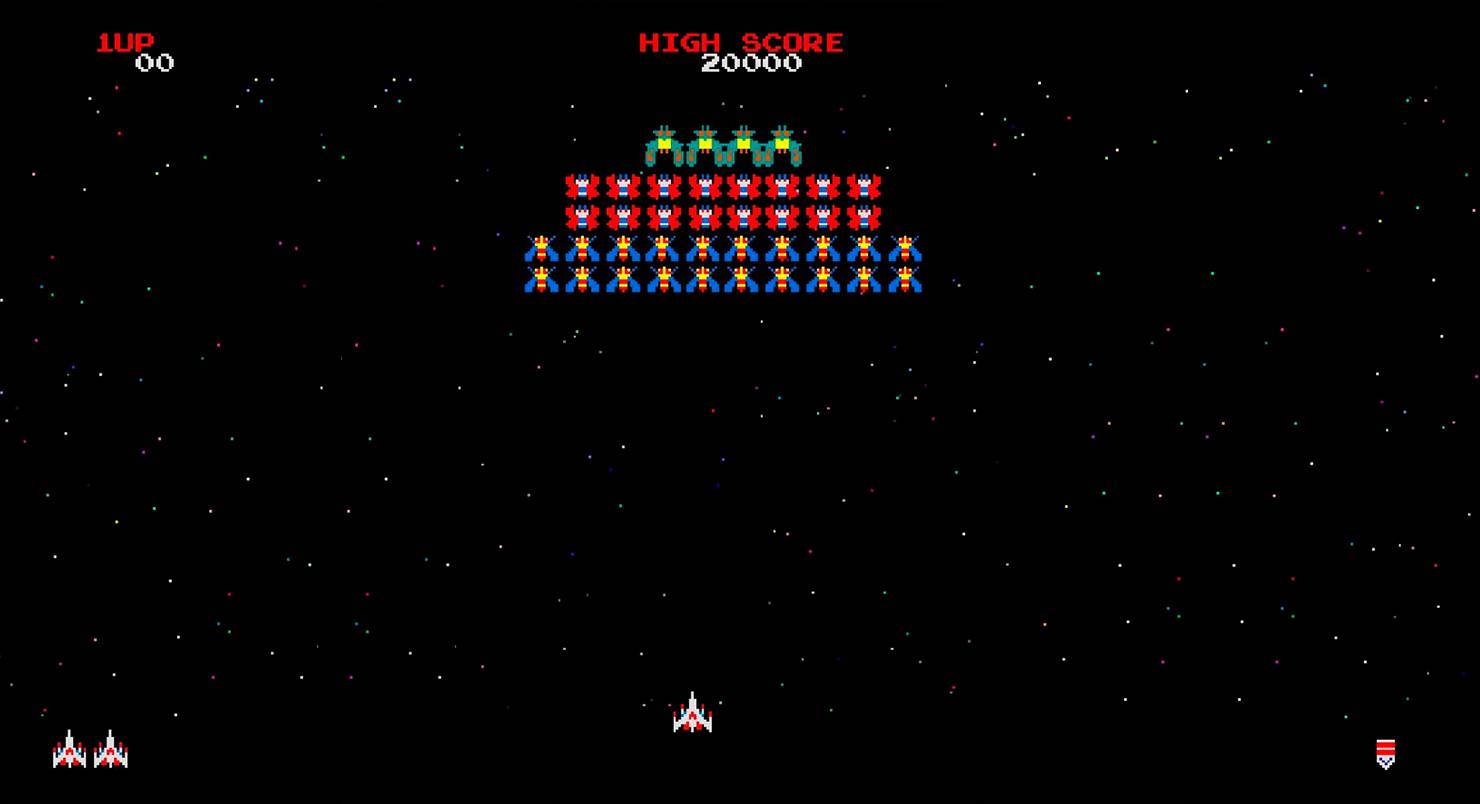


Рисунок 2 – скриншот игрового процесса «Galaga»

Изучив и проанализировав данные примеры, нашей командой было принято решение создавать проект, основываясь на дизайне и наполнении игры «Space Invaders». Из-за ограниченного количества времени, выделенного на разработку нам нужен был не очень сложный и ресурсозатратный вариант, который было бы возможно реализовать, используя CdM-8 и наши знания в области схемотехники.

3 АППАРАТНАЯ ЧАСТЬ

Аппаратная часть нашего проекта представляет из себя логические электрические схемы, созданные в программе Logisim, которая позволяет моделировать и редактировать их с помощью удобного графического интерфейса. Рассмотрим наши разработки.

**Main Interface**

Служит интерфейсом игры. Представляет собой схему *Pentamenu*, подключенную к матрице и элементам управления: клавиатуре, кнопкам «ON/OFF» и «START». Также здесь присутствует схема *Cosmsirc*, которая выполняет только эстетическую функцию – рисунок. На рисунке 3 представлен скриншоты схемы Main Interface.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 – скриншот схемы Main Interface

Для управления интерфейсом игры используется схема *Pentamenu*, которая содержит в себе *Menumega* и *Circ*, содержащая в себе уже *Gameover* и *Menu.*

**Menu**

В схеме Menu используется только вход in. Он отвечает за то, какая из строк экрана «Press Start» будет отображена. Декодер получает со входа пятибитовое значение, и поднимает бит с номером, равным этому значению. Дальше этот бит активирует один из 32 буферов, который выводит в out нужную строку (она будет либо нулем, либо какой-то константой). На рисунке 4 представлен скриншот схемы Menu.

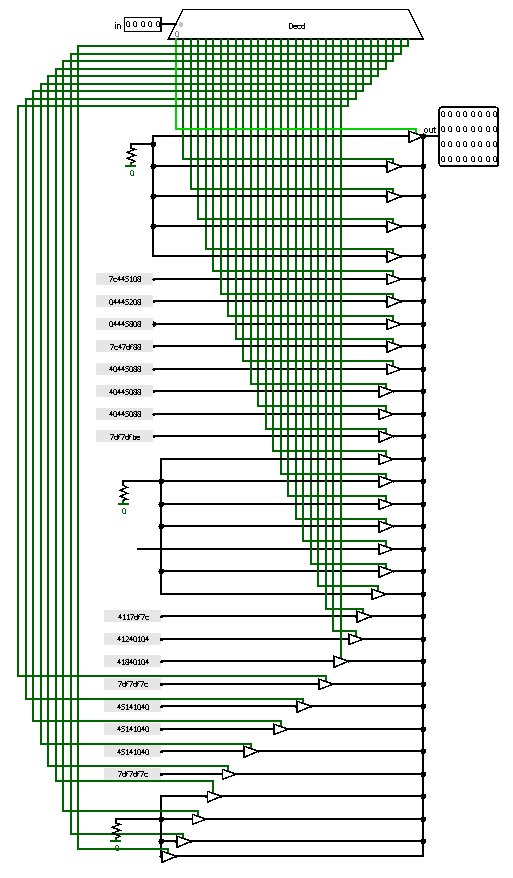


Рисунок 4 – скриншот схемы Menu

**Gameover**

Эта схема работает аналогично предыдущей. Используется декодер, поднимающий один из своих выходов в зависимости от входа in. Результатом является одна из 32 строк экрана «Game Over» или экрана «You win». Также здесь присутствует дополнительный вход isgameover, который изменяет выходящее значение с мультиплексора: если он поднят, то мультиплексоры выдают значение со входа 1, иначе со входа 0. Таким образом получается либо строки «Game Over», либо «You win». На рисунке 5 представлен скриншот схемы Gameover.

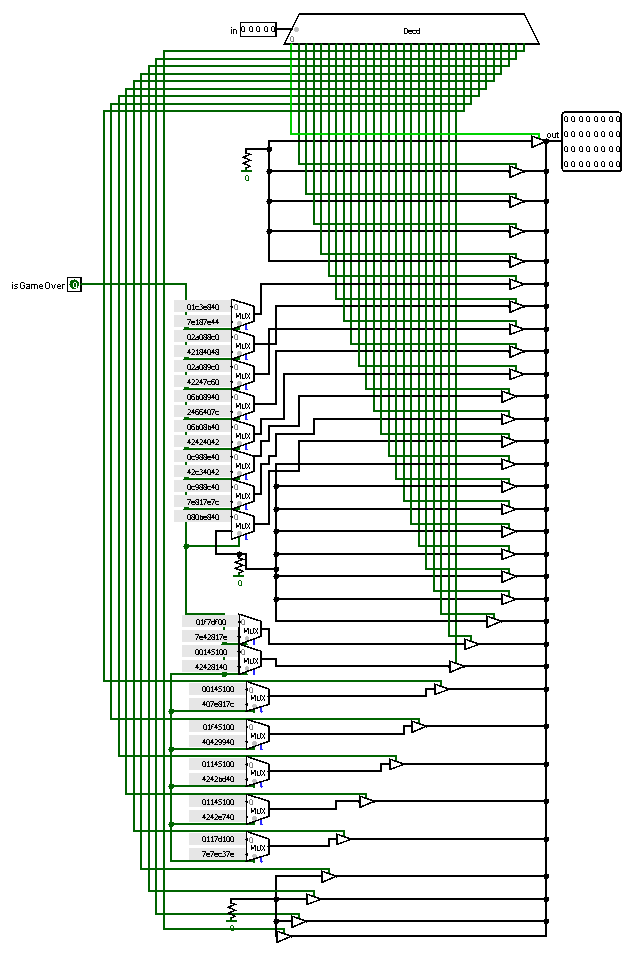


Рисунок 5 – скриншот схемы Gameover

**Circ**

В этой схеме регулируется отображение четырех различных строк: строки игры, строки из экрана «Game Over», строка из экрана «You win», либо строка из экрана «Press Start. Для этого используются четыре регулирующих входа: on, start, isgameover и endgame. Если поднят вход on, то отображается экран «Press Start», если start, то на выход схемы подается 32 битный вход game. Если поднят endgame, то выдается одна из строк экрана «You Win», если же подняты одновременно endgame и isgameover, то выдается одна из строк экрана «Game Over». Таким образом, в зависимости от поданного числа на схему числа и данных входов, мы можем получить 4 различных варианта экрана. На рисунке 6 представлен скриншот схемы Circ.

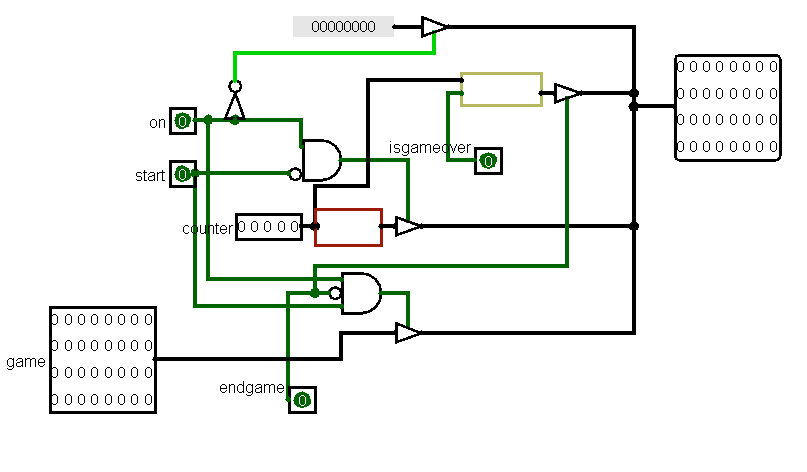


Рисунок 6 – скриншот схемы Circ

**Menumega**

В этой схеме используются 8 схем Circ для отображения 8 строк одного из 4 экранов. Выбор экрана зависит от входов gameover, endgame, on, start. Инкрементируется вход beg, определяющий какая строка является начальной. Выход end выводит число, оставшееся после инкрементирования (т.е. умноженное на 8). На рисунке 7 представлен скриншот схемы Menumega.

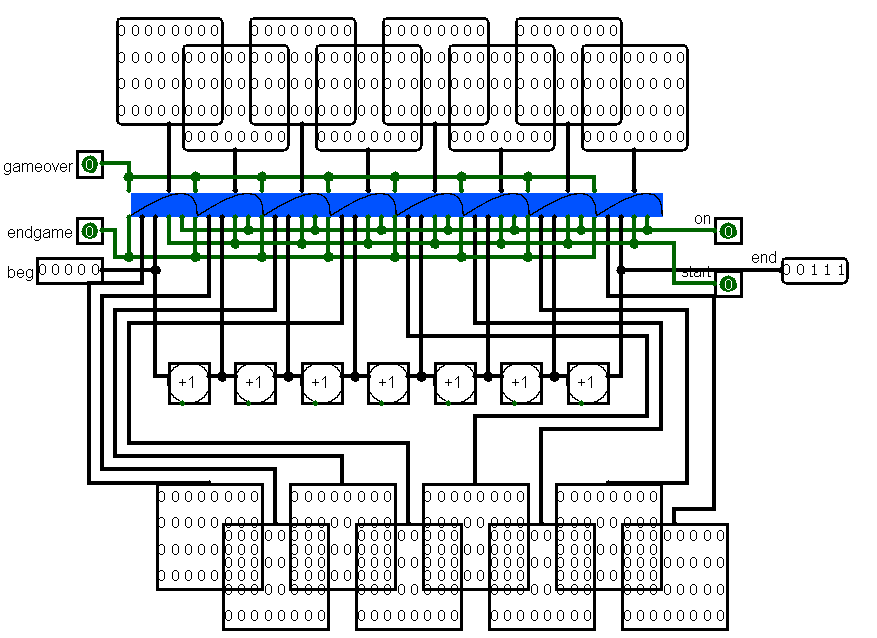


Рисунок 7 – скриншот схемы Menumega

**Pentamenu**

В этой схеме используется 4 схемы Menumega, имеющие по 8 входов (в сумме 32). Также используются 4 входа: gameover, endgame, start, on. Нулевой Menumega мы передаем нуль, и последовательно получаем по 8 строк каждого экрана от каждой схемы. Далее в зависимости от входов получаем 32 строки экрана. Если поднят on, то выводится экран «Press Start», если поднят одновременно с этим start, то выводится экран самой игры.

Экраны «Game over» и «You win» управляются внешними входами от схемы main с помощью регистров и буферов. Выходной контакт gameover от схемы main блокирует с помощью буфера отображение экрана «You win» (поднимаются одновременно значения в туннелях go и eg). Однако, если наступает состояние, когда игрок уничтожил всех противников, то поднимается триггер от win, и он блокирует с помощью буферов состояние проигрыша (поднимается, только eg). Также в схеме присутствуют пятибитовый вход control, передающийся в схему main, и выход clk, подающий такты на все остальные схемы и клавиатуру, находящуюся в Main Interface. На рисунке 8 представлен скриншот схемы Pentamenu.

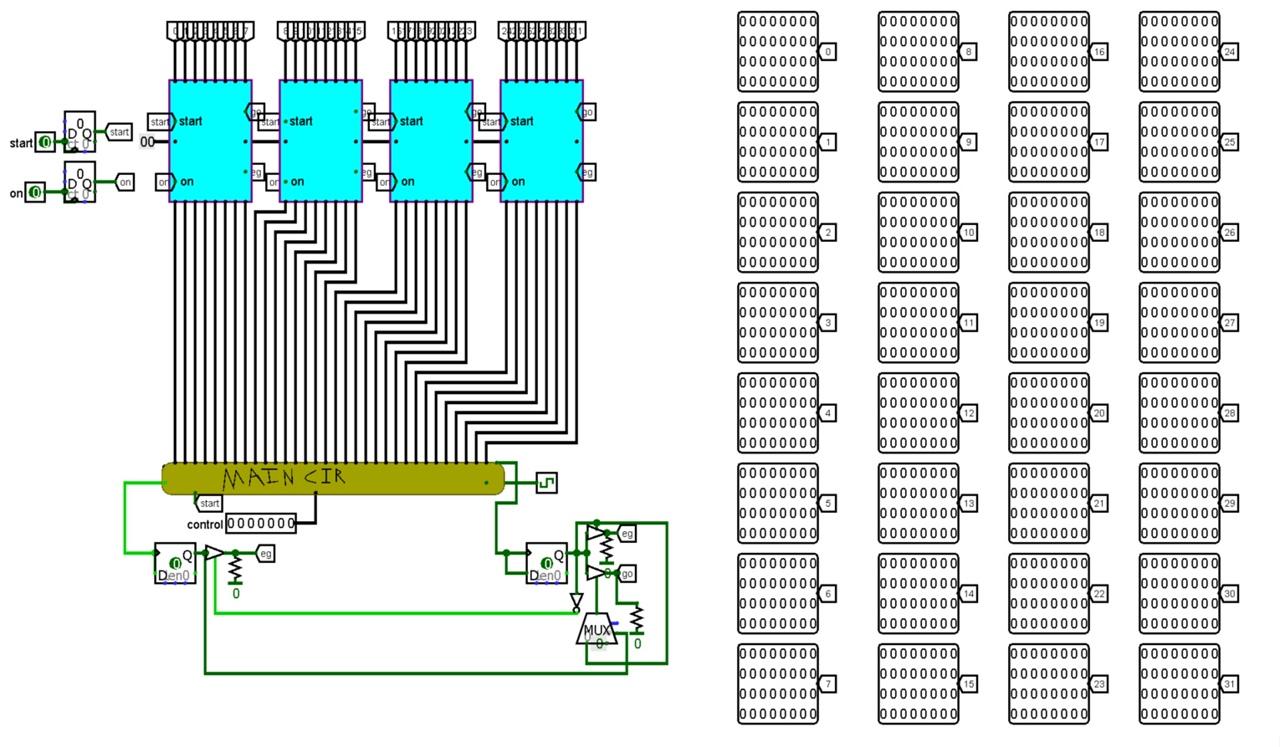


Рисунок 8 – скриншот схемы Pentamenu

**Main**

Входами являются initgame, moving и clk. Clk подаётся на все схемы, которым необходима генерация тактов, initgame подаётся на все схемы, требующие инициализацию. Вход moving реагирует на три четко определенных значения: 61, 64 и 20. Это реализовано с помощью компараторов. 61 – это код клавиши «a» и движение влево, 64 – это код клавиши «d» и движение вправо, 20 – код клавиши «SPACE» и выстрел. Если значение на входе равно 61 или 64, то происходит перемещение пушки с помощью схемы Player Move.

В Main также реализовано взаимодействие между всеми вложенными схемами.

32 выхода со схемы Shooting поданы на 32 входа к схеме enemies, обеспечивающей обработку попадания в противников, их стрельбу и передвижение.

Также есть выход boom, который в случае попадания в противника, блокирует на время стрельбу игрока для того, чтобы дать процессору время на обработку попадания.

Отображение на экран пули игрока, противников и их пуль, подключено через логическое «или» к выходам с 31 по 5, описывающих строки экрана.

На схему Walls поданы туннели 4’ и 5’’, переносящие соответственно пулю противника и пулю игрока. Из нее отходят два выхода, обозначающие попадание либо от противника, либо от игрока, в таком случае Walls блокирует работу одной из схем.

Схема Enemies имеет выход win, который обозначает, что все враги были уничтожены, и выход с 4 строки передвижения монстров. если хоть один бит там равен 1, то сразу же поднимается выход gameover в Main. Также, в gameover через или содержит в себе логические «и» от пуль монстров и передвижения игрока. Если хоть где-то есть 1, то в игрока было попадание, и поднимается gameover. На рисунке 9 представлен скриншот схемы Main.

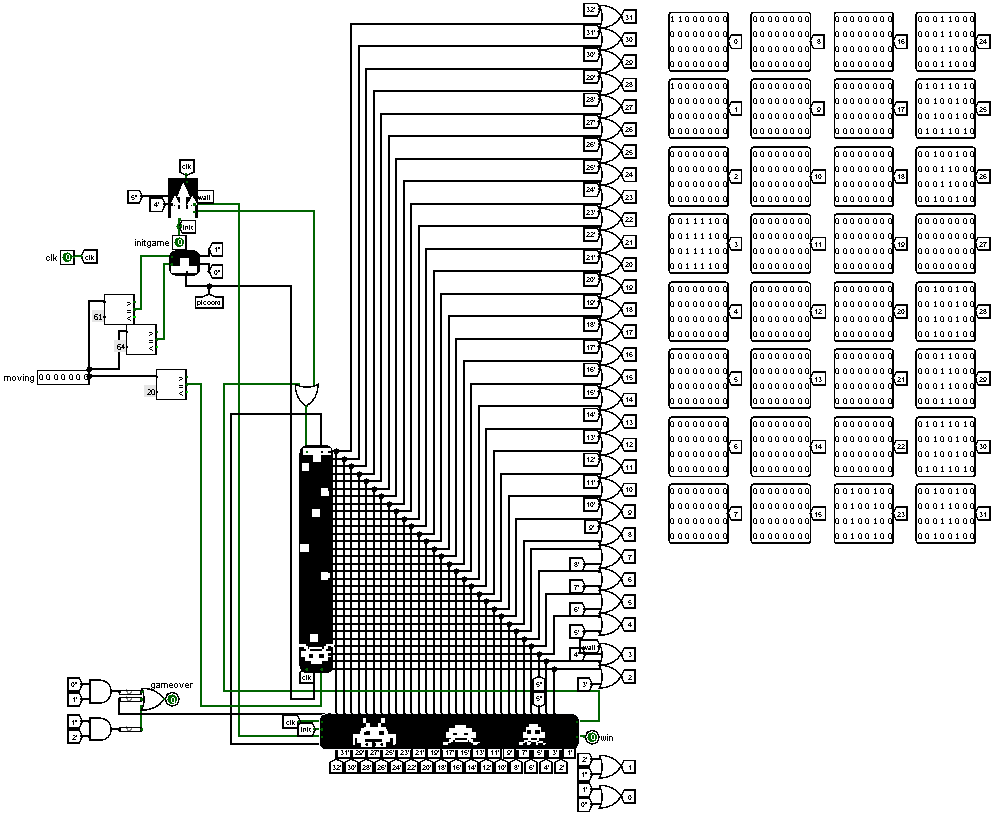


Рисунок 9 – скриншот схемы Main

**CDMEnemShoot**

Эта схема принимает на вход двухбитовое число, сгенерированное генератором случайных чисел rand и байт сдвига врагов shift, со значением -1 (сдвиг врагов влево), либо 1 (сдвиг врагов вправо). Вход clk задает частоту процессора и элементов памяти. На основе входных данных rand и shift CdM-8 генерирует номер столбца матрицы, куда будет помещена пуля (res). Выходной контакт out выдает res.

Так как код должен выполн иться несколько раз, он зациклен. Нам нужно останавливать процессор и запускать его снова. Для этого используется демультиплексор, который подает на питание тактовый генератор при выключенном триггере состояния PC, равном 1 (байт кода начала основной подпрограммы). Этот триггер управляется триггером состояния ручного запуска, который активируется при срабатывании входного контакта En и отключается по завершению программы. Выходной контакт Ready говорит окружению, завершена ли обработка программы. На рисунке 10 представлен скриншот схемы CDMEnemShoot.

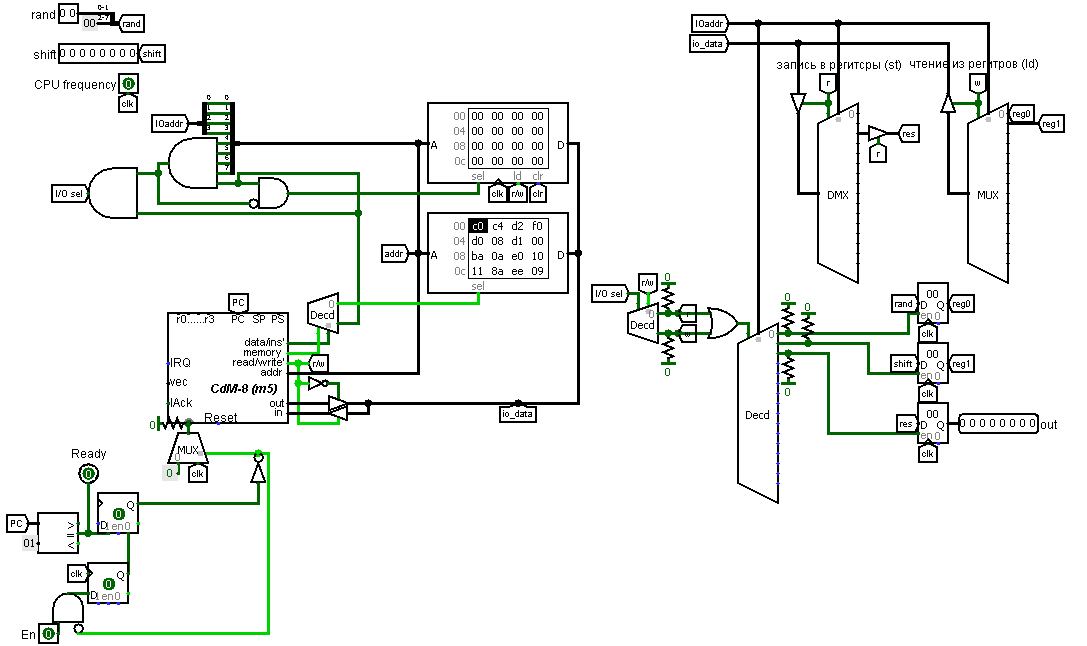


Рисунок 10 – скриншот схемы CDMEnemShoot

**Inc 5bit**

На этой схеме реализован инкремент с помощью двух вентилей: пятибитового «и» и исключающего «или». Значение пятибитового входа x подаётся на xor и на «и». Выход с вентиля «и» разбивается разветвителем, нулевой бит числа заменяется единицей, а старший передаётся на выход Cout. На рисунке 11 представлен скриншот схемы Inc 5bit.

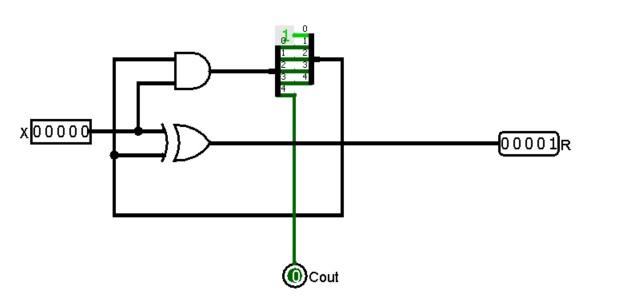


Рисунок 11 – скриншот схемы Inc 5bit

**Player Move**

Отображение движения игрока реализуется с помощью декодера и счётчика. Значение счётчика увеличивается или уменьшается в зависимости от входов l и r. Далее оно передаётся на декодер, который подаёт единицу в бит под номером, пришедшим на вход. После этого данный бит сдвигается на 1 в лево и в право от начальной позиции и с помощью операции «или» между этими битами получается часть выводимой на экран пушки. Результат подаётся на выход 0. Значение с декодера без сдвигов подаётся на выход 1 для получение полного изображения пушки.

Также на схеме есть выход counter. На него подаётся нынешнее значение со счётчика. На рисунке 12 представлен скриншот схемы Player Move.

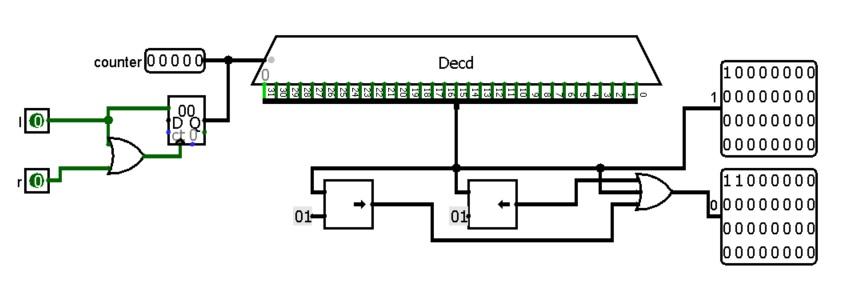


Рисунок 12 – скриншот схемы Player Move

**Transpon Field**

Схема реализует транспонирование игрового поля. На ней находится 32 тридцатидвухбитных входа и 32 тридцатидвухбитных выхода. Из туннелей ко входам от 0 до 31 с помощью разветвителя вычленяется сначала нулевой бит и подаётся на выход 0. Затем выделяется первый бит, а выходы с разветвителей объединяются и подаются на выход 1. Аналогичные операции проводятся с каждым битом. На рисунке 13 представлен скриншот схемы Transpon Field.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 13 – скриншот схемы Transpon Field

**Walls**

Входные контакты: init указывает началась ли игра, clk – тактовый генератор, 2 тридцатидвухбитных входа bull для различных пуль (игрока и монстров). Результатом пропускания входов bull через схему для определения попадания будет являться число, в котором бит, равный единице обозначает место попадания. Также, при попадании, в 32-битный регистр загружается значение, показывающее место попадания. Это происходит только, если выход с фиксирования попадания пули противника или игрока не равен нулю. Если было попадание монстра в стенку, то поднимается триггер с выходом на BOOMmon!. Если было попадание игрока или пришельца, то поднимается триггер с выходом на BOOM!.

Выход BOOMmon! используется только для отслеживания, было ли попадание противника, чтобы на время обработки отключить врагам стрельбу. Также есть тридцатидвухбитный выход на значение самих стенок (их состояние хранится с помощью регистра).

Управление процессором происходит по аналогии со схемой CDMEnemShoot, но управление Триггером ручного запуска зависит от состояний BOOM! и StartGame. Состояние BOOM! хранится в триггере, который срабатывает от контакта trigger схемы Collision и обнуляется при состоянии DONE. Состояние DONE хранится в регистре, который обнуляется при срабатывании DONE и выдает единицу, когда счетчик количества обращений к 3 стенке равен 2. На рисунках 14 и 15 представлены скриншоты схемы Walls.

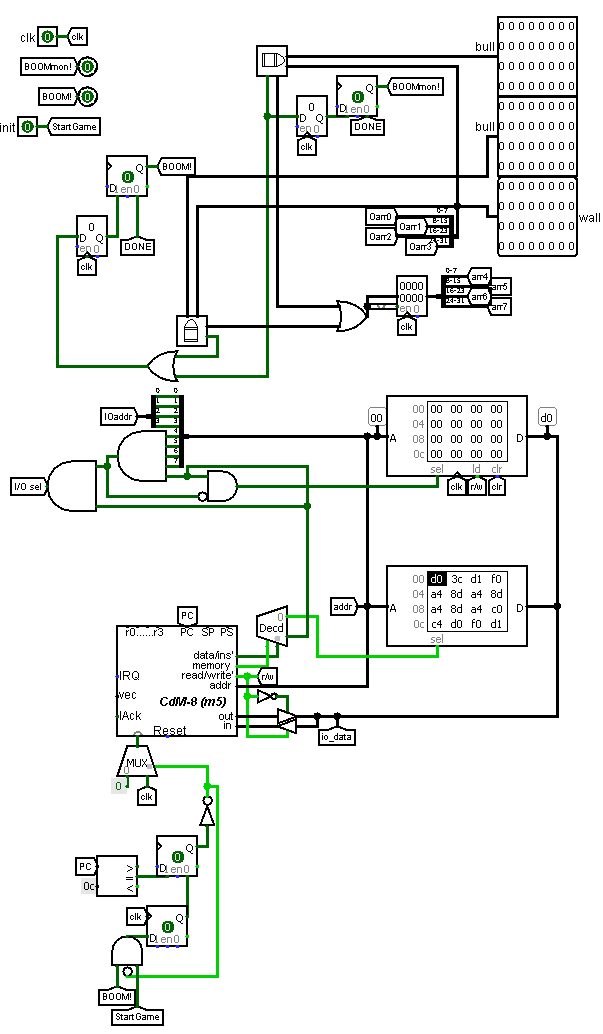


Рисунок 14 – скриншот части схемы Walls

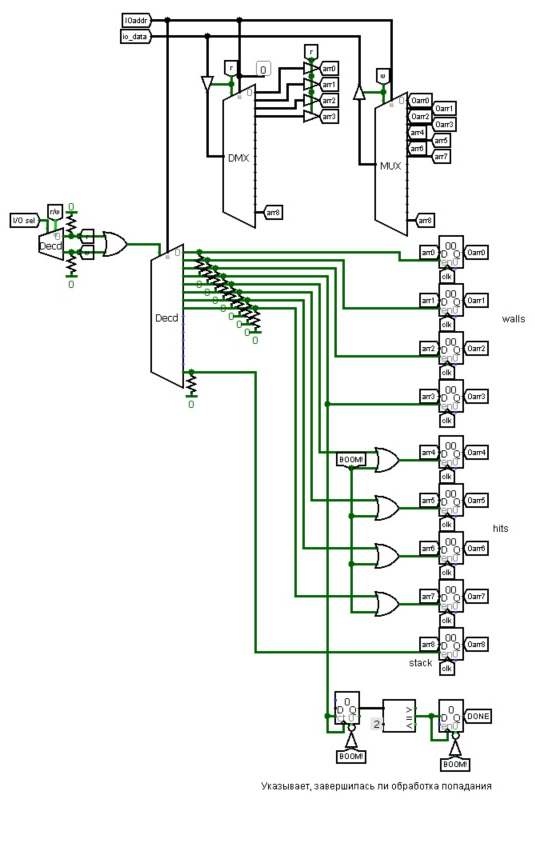


Рисунок 15 – скриншот части схемы Walls

**Collision**

Координаты двух объектов на экране совпадают, когда один и тот же бит в разных строках равен единице. Данная схема реализует проверку столкновения путём выполнения операции «и» между двумя тридцатидвухбитными входами A и B. Результат операции подается на выход OUT. Также на схеме есть выход Trigger, на который подаётся значение с вентиля «и», пропущенное через схему Or32. На рисунке 16 представлен скриншот схемы Collision.

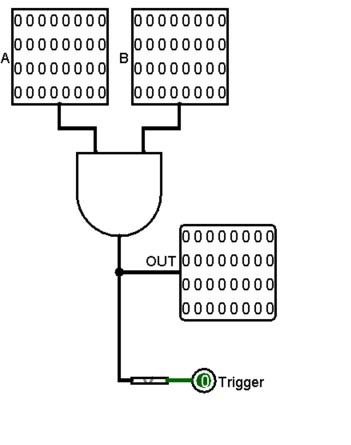


Рисунок 16 – скриншот схемы Collision

**Or 32**

На схеме расположен один традцатидвухбитный вход, разбивающийся разветвителем на 32 однобитных значения, который проходят через вентиль «или» и подаются на выход out. На рисунке 17 представлен скриншот схемы Or 32.

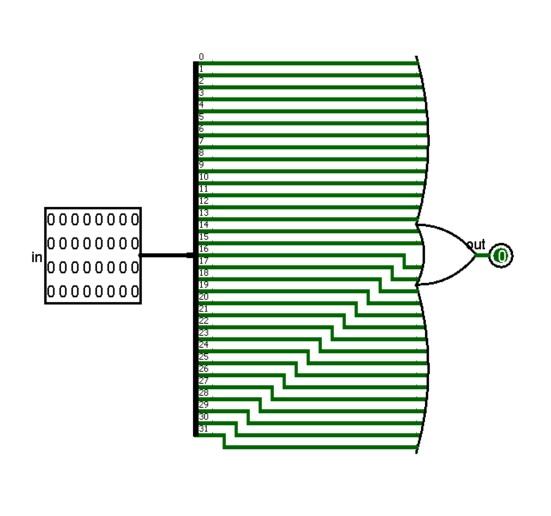


Рисунок 17 – скриншот схемы Or 32

**Shooting**

На схеме присутствует 4 входа: fire, reset, coord и clk. С помощью них осуществляется управление стрельбой игрока. Если была нажата кнопка выстрела, то активируется вход fire, который пропускается к D-триггеру, если reset не равен единице. Вход reset необходим для блокировки стрельбы на время обработки попадания в игровой объект. Вход coord подается на вход к управлению выходом из демультиплексора. Столбец из пуль подаётся в столбец, где находится игрок. Выходы с демультиплексора подключены через схему Transpon Field для работы со столбцами. После поднятия fire поднимается D-триггер, выход которого через ИЛИ с инвертированным входом подан на обнуление регистра. Таким образом, регистр обнуляется в случае, если выход fire не поднят, либо поднят вход reset. Регистр увеличивает свое значение каждый раз, когда поднимается вход clk. С помощью декодера поднимается бит, который каждый раз смещается на 1 из-за увеличения значения счётчика. Выходы с декодера собираются в 32 битное число, которое передается на вход к демультиплексору, передающему его на один из 29 выходов. Также на схеме есть пятибитовый выход для использования другими схемами. На рисунке 18 представлен скриншот схемы Shooting.

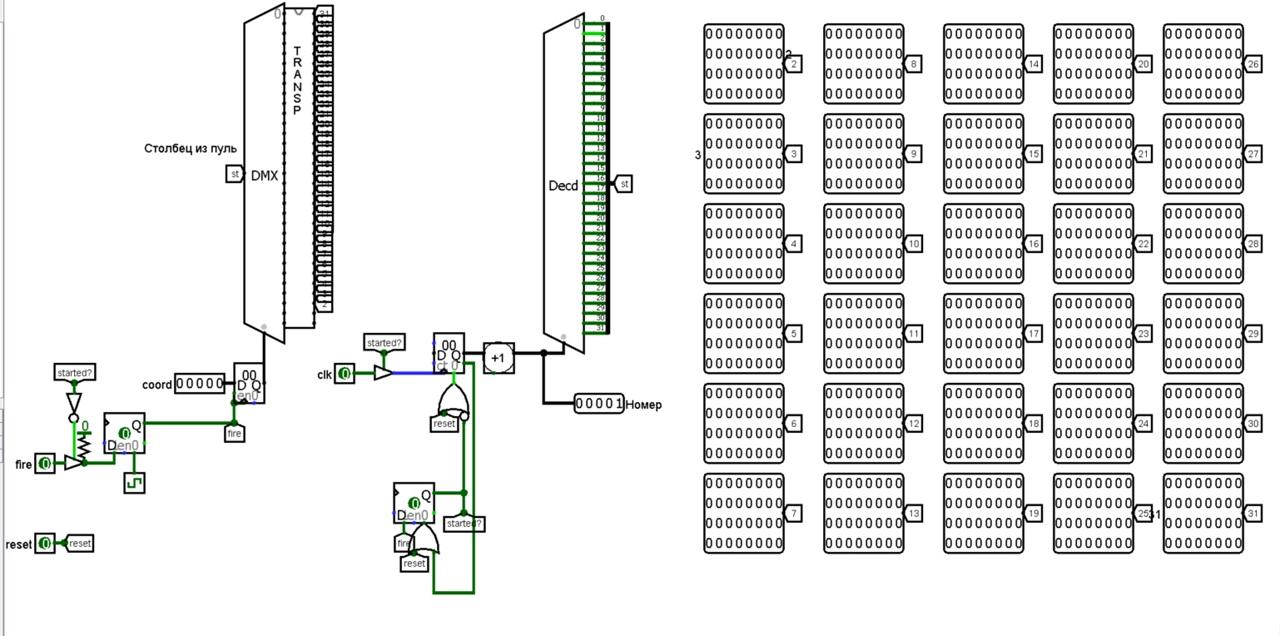


Рисунок 18 – скриншот схемы Shooting

**Shifts**

Эта схема реализует управление движениями пришельцев. Для этого используется счетчик, принимающий значения от 0 до 5: 1 – сдвиг влево, 2 и 5 – сдвиг вниз, 3 – обнуление сдвига влево, 4 – сдвиг вправо, 6 – обнуление сдвига вправо.

Сдвиги вычисляются путём пропускания 9 входных строк через ShifterLeft или ShifterRight, управляемые счетчиками, в которых хранится величина сдвигов, и ShifterDown, который сдвигает полученный результат вниз на значение своего счетчика.

Выходные данные – это новый слой сдвинутых врагов (2e – 31e), значение сдвига shift (влево (-1) вправо (1) или его отсутствие (0)) и 5-битовое значение сдвига вниз height. На рисунке 19 представлен скриншот схемы Shifts.

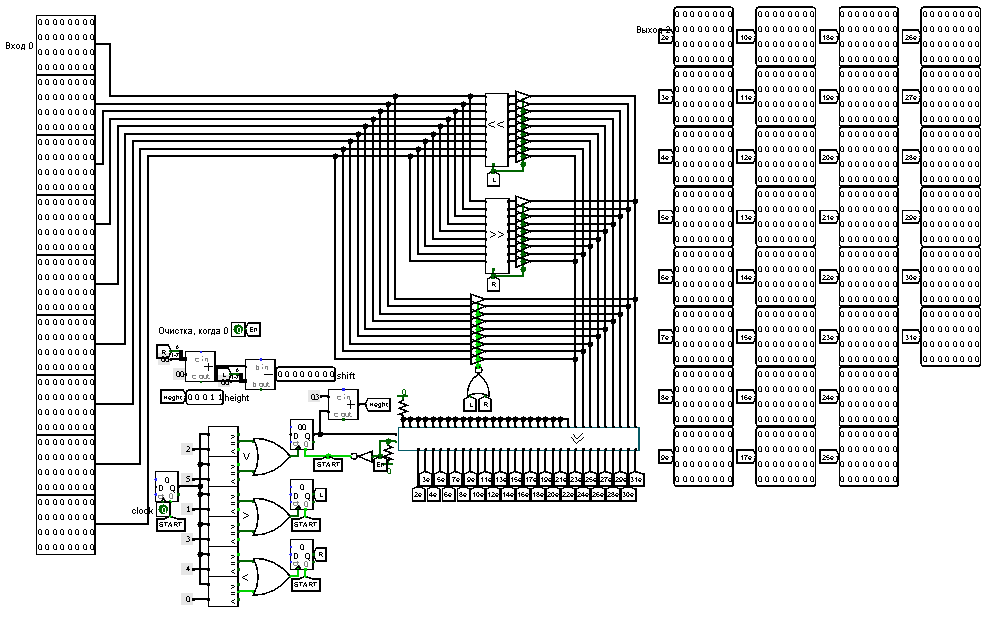


Рисунок 19 – скриншот схемы Shifts.

4 ПРОГРАММНАЯ ЧАСТЬ

Программная часть нашего проекта была реализована с помощью ассемблерного языка процессора CdM-8 в интегрированной среде разработки CocoIDE, созданной специально для разработки кода, выполняемого этим процессором. Рассмотрим наши алгоритмы.

**Wall destruction**

Алгоритм, реализующий инициализацию и разрушение защитных стенок при попадании в них пуль монстров. Входные данные – это два четырёхбайтовых массива: строка с текущим состоянием стенок и нулевая строка с единичным битом попадания. Выходные данные – строка с состоянием стенок после попадания или промаха.

Первым шагом инициализируются регистры, находящиеся по адресам F0-F3, значением 3С (00111100 в двоичной системе). Эта константа – выводимая на экран стенка из 4 пикселей. На втором шаге в регистры общего назначения загружаются входные данные, инициализируется счётчик. Затем в цикле, пока счётчик больше нуля, выполняется операция исключающего «или» между соответствующими байтами входных массивов, и результат сохраняется в регистр, в который были загружены стенки. Таким образом, итог – это битовая строка с состоянием «защит» после попадания или промаха. По окончании цикла происходит переход на второй шаг алгоритма. На рисунке \_ представлена блок-схема алгоритма инициализации и разрушения стен.

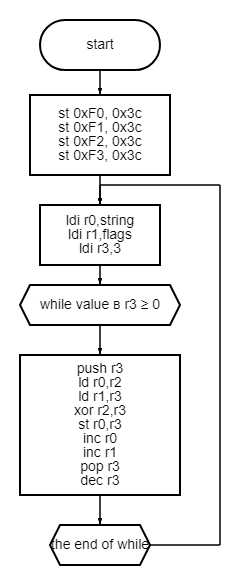


Рисунок \_ - блок-схема алгоритма инициализации и разрушения стен

**Monsters shooting**

Алгоритм, реализующий вычисление координаты (номера столбца), в которой монстр произведёт выстрел. Входные данные – два однобайтовых числа: номер пришельца, совершающего выстрел (от 0 до 3) и значение смещения монстра (-1 или 1). Выходные данные – однобайтовый номер столбца.

Первым шагом в регистры общего назначения загружаются номер пришельца, который будет использоваться в качестве счётчика, а также константы 8 и 0. Затем после вычислений в цикле получится значение, равное номеру монстра, совершающего выстрел, умноженное на 8. После этого к нему прибавляется значение смещения и константа 4, а результат (получившийся номер столбца) сохраняется в регистр. Далее происходит переход на первый шаг алгоритма. На рисунке \_ представлена блок-схема алгоритма вычисления номера столбца, в котором будет произведён выстрел.

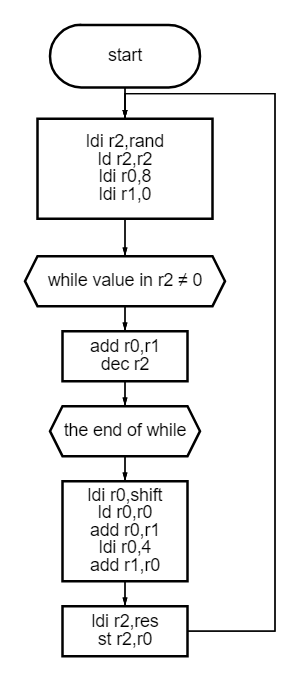


Рисунок \_ - блок-схема алгоритма вычисления номера столбца, в котором будет произведён выстрел

**Hit**

Алгоритм, реализующий инициализацию состояния пришельцев обработку попадания пули игрока в них. Входные данные – байт, хранящий состояния инопланетян (если бит равен единице, то монстр жив, если нулю, то нет) и четырёхбайтовый массив – нулевая строка с единичным битом попадания. Выходные данные – байт, хранящий новые состояния врагов.

На первом шаге инициализируется байт состояний монстров значением FF (11111111 в двоичной системе). Вторым шагом в регистры общего назначения загружаются нулевая строка с единичным битом попадания, а также константа 0, инициализируется счётчик. Затем в цикле ищется не равный нулю байт в массиве и вычисляется номер пришельца, в которого было осуществлено попадание. После этого путём побитовых сдвигов влево вычисляется однобайтовая маска, где единственный единичный бит имеет индекс, равный вычисленному ранее номеру инопланетянина, который находится в нижнем ряду. На следующем шаге происходит сравнение этой маски и байта, хранящего состояния врагов, путём выполнения операции побитового «и». Если результат равен нулю, то монстра в нижнем ряду нет, поэтому происходит еще 4 побитовых сдвига влево. Далее выполняется операция исключающего «или» между новой маской и байтом состояний, а результат сохраняется в регистр. После сохранения происходит переход на второй шаг. На рисунках \_ и \_ представлены части блок-схемы алгоритма инициализации состояния пришельцев и обработки попадания в них.

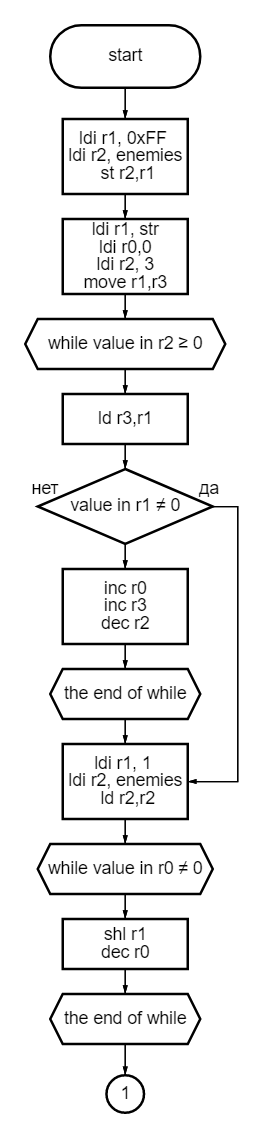


Рисунок \_ - первая часть блок-схемы алгоритма инициализации состояния пришельцев и обработки попадания в них

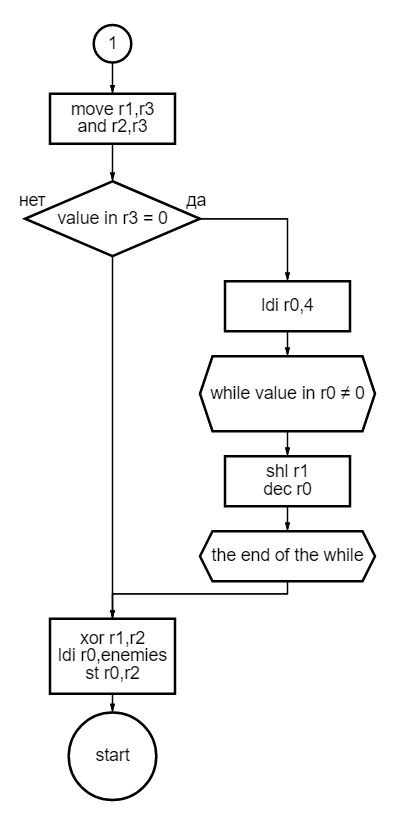


Рисунок \_ - вторая часть блок-схемы алгоритма инициализации состояния пришельцев и обработки попадания в них

Таким образом, разработанная игра – это результат взаимосвязанной работы аппаратной и программной частей проекта, описанных ранее.

ЗАКЛЮЧЕНИЕ

В результате проделанной работы у нас получилось создать игру «Rush space» в жанре Shoot ‘em up (руководство пользователя см. приложение А). Она реализована, используя электронные схемы, в которых мы задействовали процессор CdM-8, выполняющий написанные нами коды. Задачи, поставленные перед нами в начале проекта, были успешно выполнены. Мы смогли реализовать все обозначенные функциональные требования: движения и стрельба игрока и противников, управление с кнопок клавиатуры, возможности выигрыша и проигрыша в игре. Создавая проект, мы получили знания в области создания электросхем, работы с процессором построенном на Гарвардской архитектуре и его программирования, написания проектной документации, а также опыт работы в команде.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Computing platforms: учебник / A.Shafarenko, S.P.Hunt. – 2015.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А

**Руководство пользователя**

При открытии игры пользователь видит электросхему, стилизованную под аркадный автомат. На ней расположены две кнопки и клавиатура, с помощью которых игрок может осуществлять управление игрой. На рисунке А.1 представлен скриншот начального экрана игры.

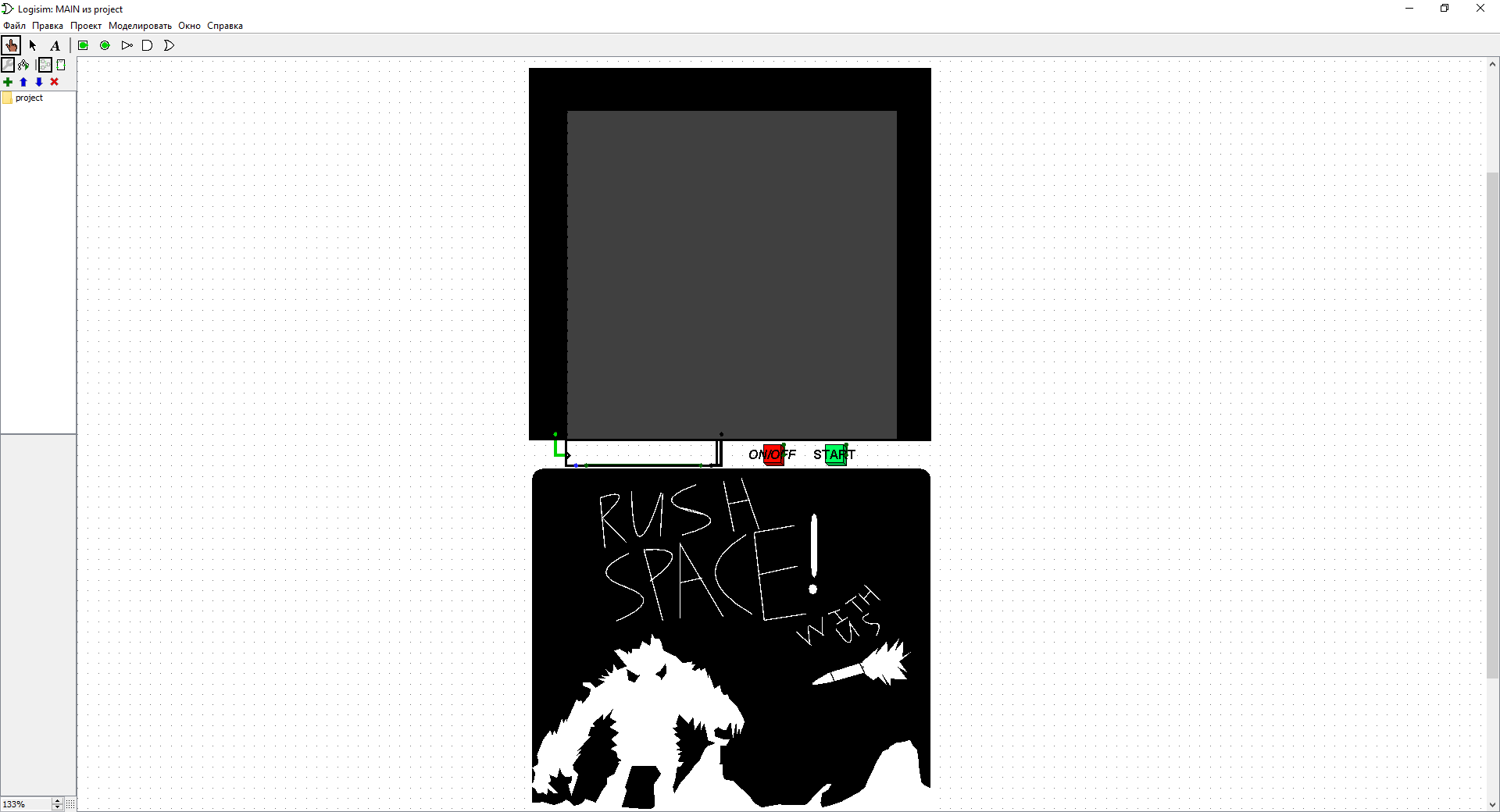


Рисунок А.1 – скриншот начального экрана игры

Для включения игры пользователю нужно нажать кнопку «ON/OFF». После этого игрок увидит на экране сообщение о том, что для начала игрового процесса ему необходимо нажать кнопку «START». На рисунке А.2 представлен скриншот экрана с этим сообщением.

Изображение выглядит как текст, часы

Автоматически созданное описание

Рисунок А.2 – скриншот экрана с просьбой нажать кнопку «START»

При нажатии на кнопку, запускающую игру, на экран выводится поле и начинается игровой процесс. Игрок управляет пушкой, находящейся в нижней части экрана, в горизонтальной плоскости с помощью клавиатуры, а именно клавиш “w”, “a”, “s”, “d”. Также на поле пользователь видит 4 статичные стенки, и 8 монстров, движущихся неуправляемо по горизонтали и вертикали. На рисунке А.3 представлен скриншот игрового поля в момент начала игры.

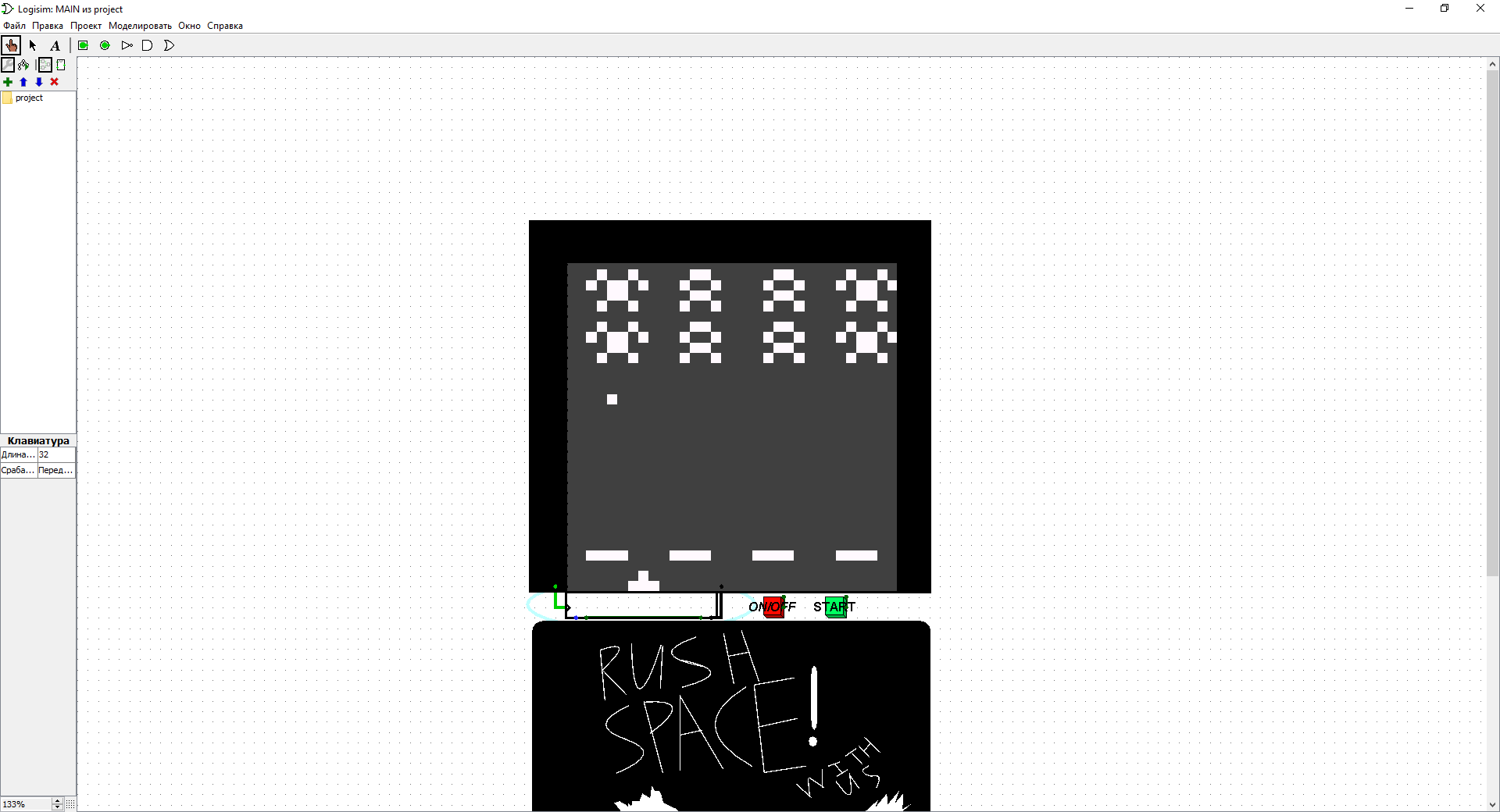


Рисунок А.3 – скриншот игрового поля в момент начала игры

Также пользователь может стрелять, используя клавишу "SPACE” на клавиатуре. При нажатии из пушки вылетает пуля, летящая вертикально вверх до того момента, пока не попадет в пришельца, либо до того, как выйдет за пределы игрового поля. На рисунке А.4 показан скриншот игрового поля в момент выстрела.

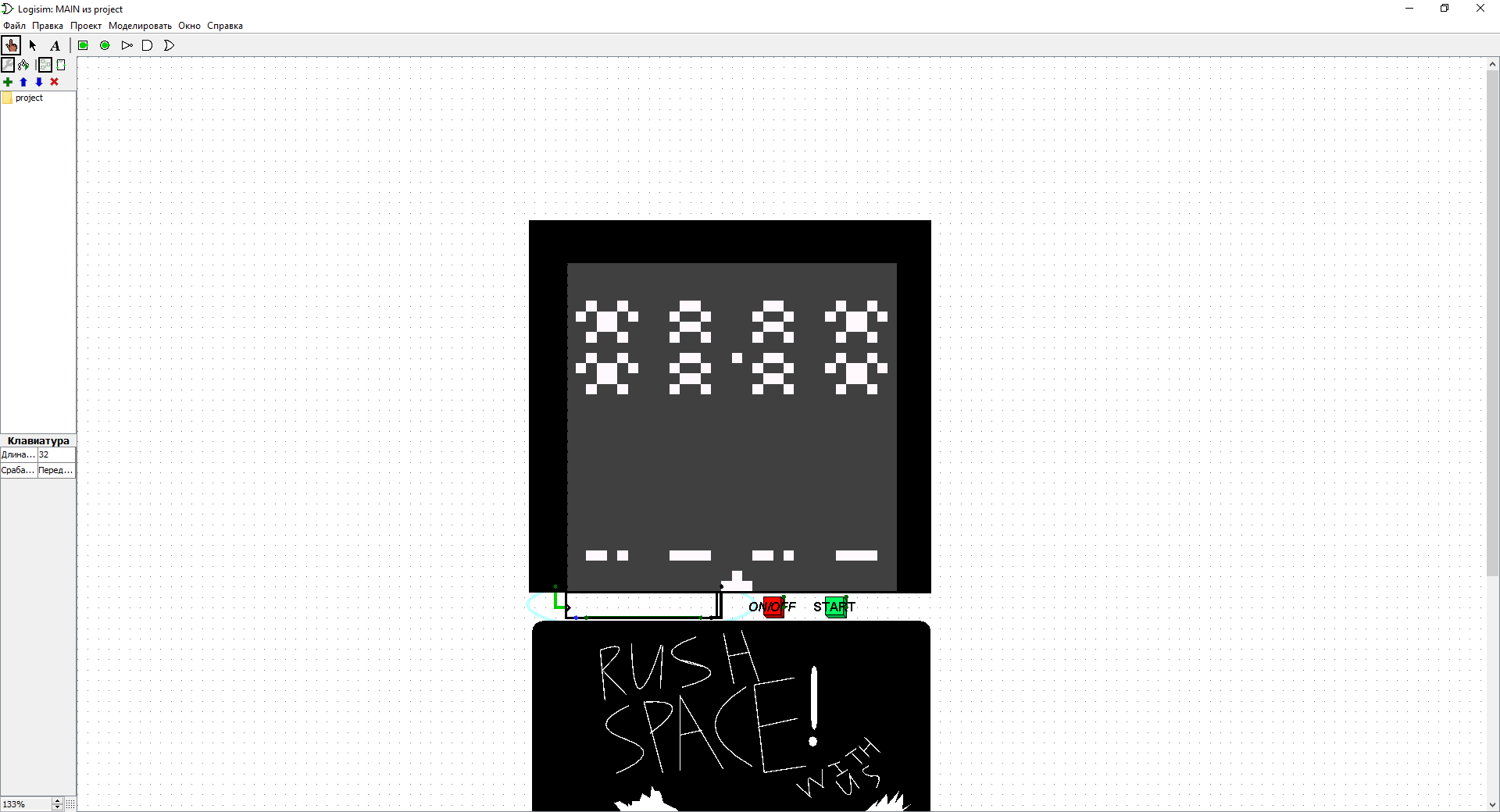


Рисунок А.4 – скриншот игрового поля в момент выстрела

Монстры тоже осуществляют стрельбу. Каждые несколько секунд случайный пришелец выпускает пулю, летящую вертикально вниз до того момента, пока не попадет в игрока, либо в стенку, либо до того, как выйдет за пределы игрового поля. Стенка, защищающая игрока, при попадании в неё пули частично разрушается. Каждая «защита» выдерживает 4 попадания, прежде чем полностью разрушится. Важно заметить, стенка может быть атакована не только инопланетянами, но и самим игроком. На рисунке А.5 представлен скриншот игрового поля с частично разрушенными стенками.

Изображение выглядит как текст, часы, снимок экрана

Автоматически созданное описание

Рисунок А.5 – скриншот игрового поля с частично разрушенными стенками

При попадании пули пришельца в игрока он проигрывает, и на экране появляется сообщение о конце игры. На рисунке А.6 представлен скриншот сообщения о конце игры.

Изображение выглядит как текст, часы, снимок экрана

Автоматически созданное описание

Рисунок А.6 – скриншот экрана с сообщением о конце игры

Игрок может проиграть и при другом условии. Каждые несколько секунд все монстры перемещаются ближе к пушке. Если хотя бы один пришелец сможет выжить и добраться до стенок, то игра будет закончена и на экране появится сообщение о её конце. На рисунке А.7 представлен скриншот игрового поля в состоянии близком к проигрышу.

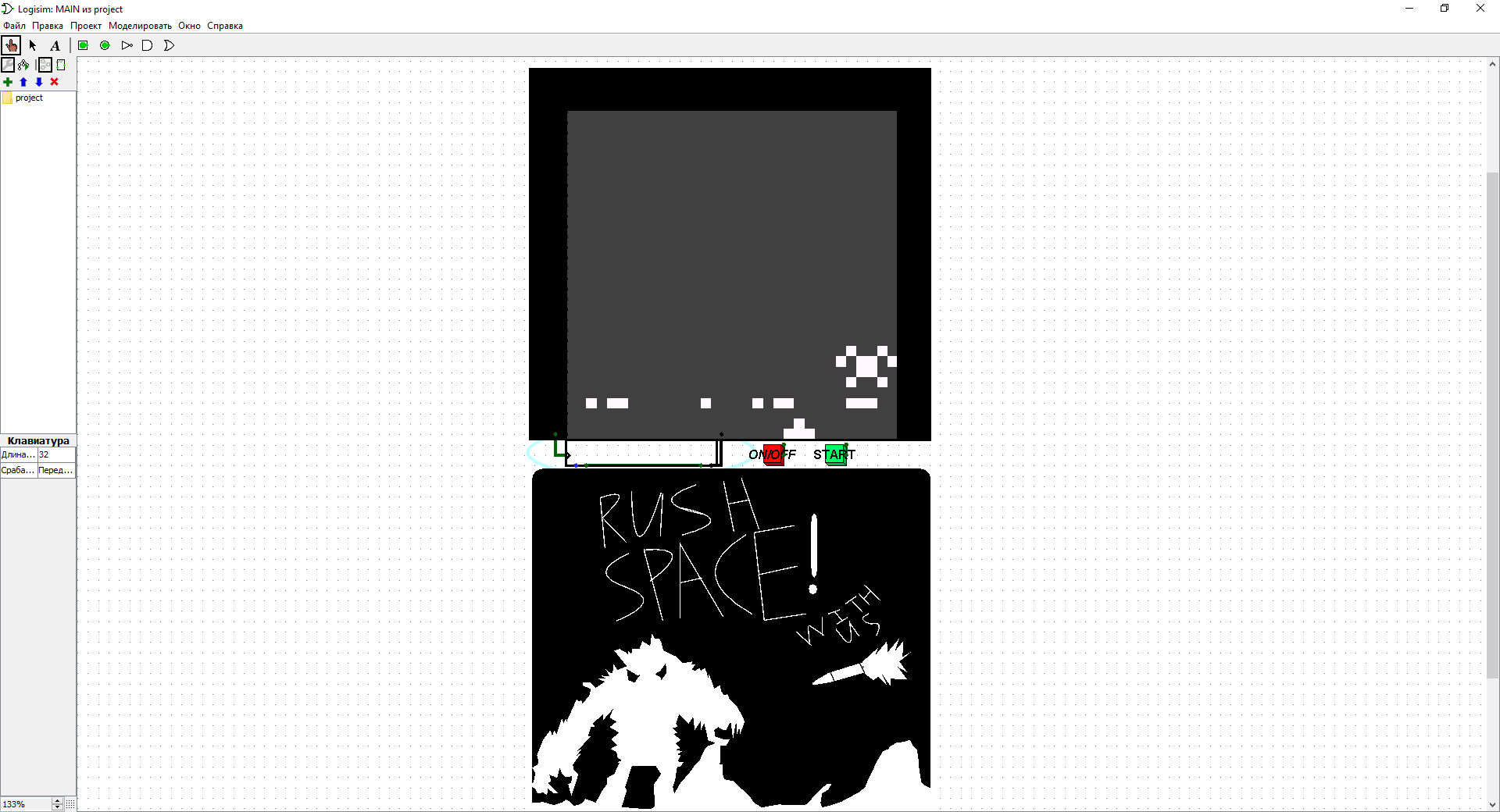


Рисунок А.7 – скриншот игрового поля в состоянии близком к проигрышу

Для победы пользователь должен уничтожить всех монстров. На рисунке А.8 представлен скриншот игрового поля после попадания в большинство инопланетян.

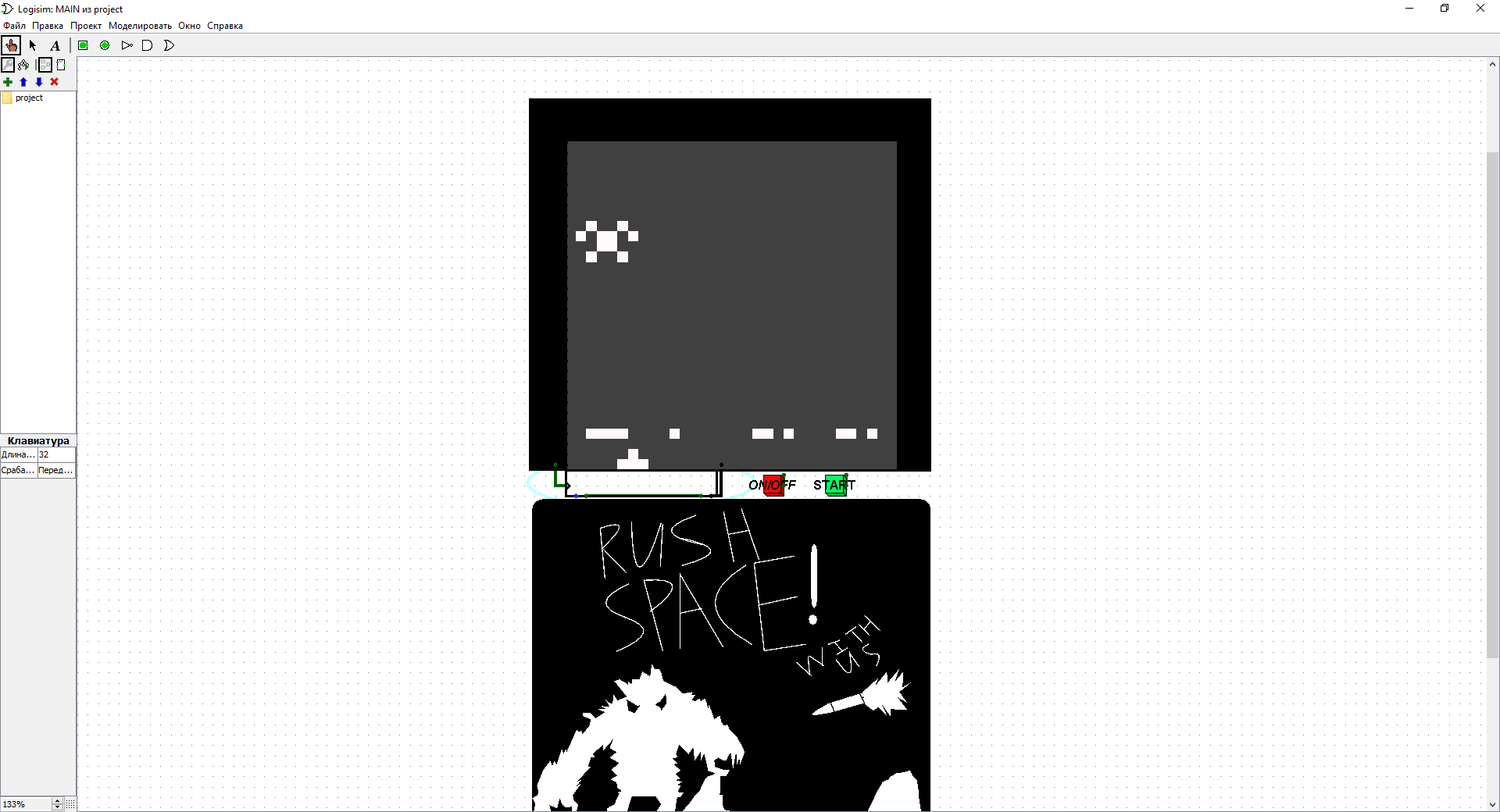


Рисунок А.8 – скриншот игрового поля после уничтожения большинства инопланетян

После уничтожения всех пришельцев на экране появится сообщение о победе игрока. На рисунке А.9 представлен скриншот сообщения о победе.

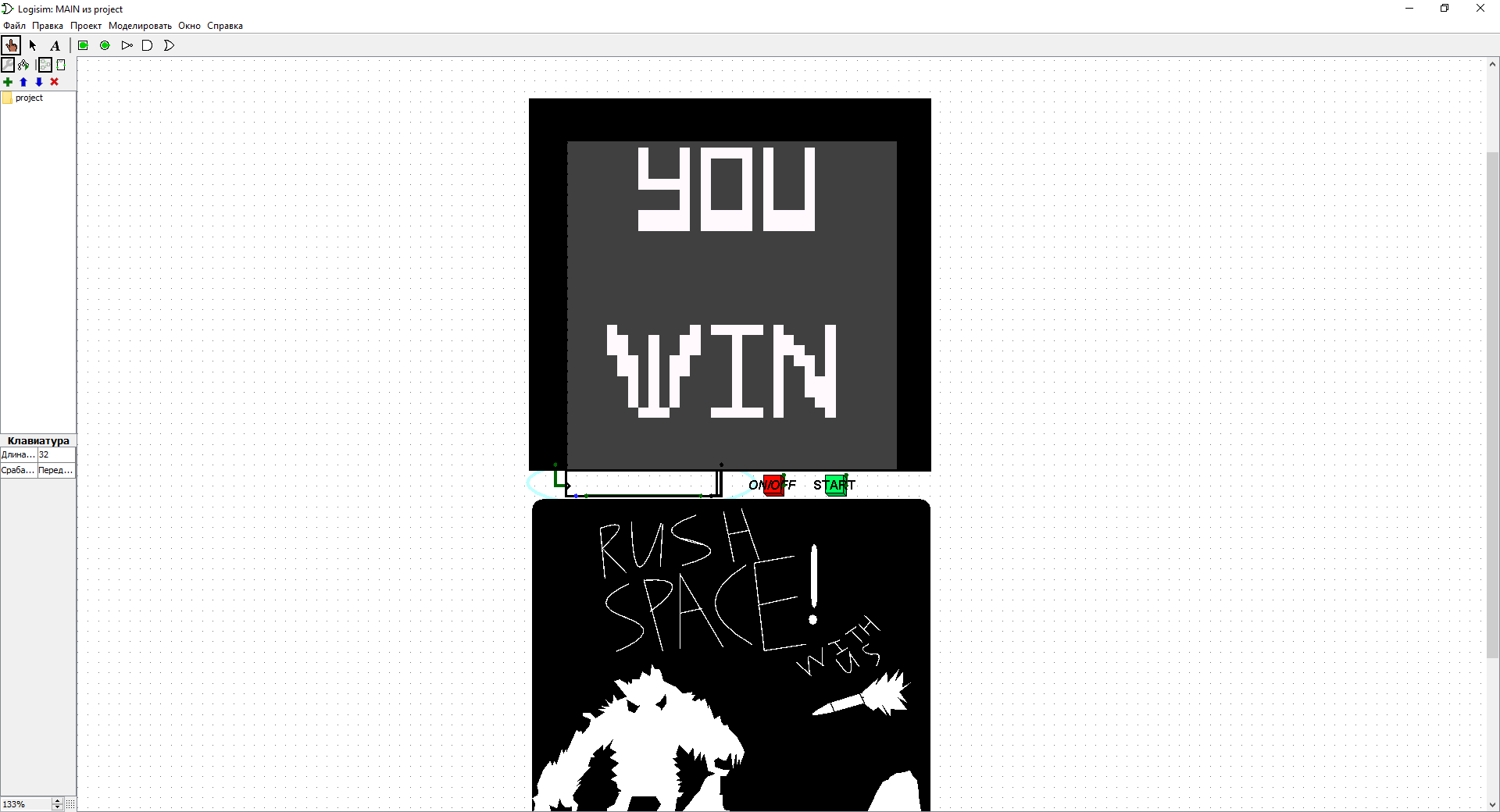


Рисунок А.9 – скриншот сообщения о победе