MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN FEDERATION
FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION OF HIGHER EDUCATION
"NOVOSIBIRSK NATIONAL RESEARCH UNIVERSITY
STATE UNIVERSITY"
(NOVOSIBIRSK STATE UNIVERSITY, NSU)

15.03.06 - Mechatronics and Robotics
Focus (profile): Artificial Intelligence

EXPLANATORY NOTE

Job topic: **«DIGITAL FLAPPY BIRD»**

Agaev Ali Malik ogly, 23934
Perminov Lev Alekseevich, 23934
Trushkin Ilya Andreevich, 23934

Novosibirsk

2024

TABLE OF CONTENTS

1 TERMS AND ABBREVIATIONS

FB	"Flappy Bird"- is a mobile game where the player controls the flight of a bird by tapping the screen, guiding it between rows of green pipes without hitting them.

2 INTRODUCTION

As part of the CS/MR Digital Platforms 2023/24 course, each student is required to design and create a game on electronic circuits using the CdM-8 processor and its assembly language.

"Digital Flappy Bird" - is a project dedicated to developing a replica of the popular game "Flappy Bird" (Screenshot №1 in the APPENDIX) (hereafter referred to as FB).
FB - is a mobile game where players control the flight of a bird by tapping the screen, maneuvering between rows of green pipes without touching them. The player's goal is to score as many points as possible.

Why we chose FB as the basis for the project:

- Simplicity of gameplay: FB is known for its simplicity and ease of understanding game mechanics.
- Popularity of the game: FB was a cult game with huge popularity. Using such a recognizable game may attract attention and interest students and teachers.
- Choosing the game FB for the project is unpopular, as few students typically choose this theme.

Purpose of the course project:
The aim of this project is to design and create a replica of the game FB based on an electronic circuit incorporating the 8-bit CdM-8 processor.

3 REVIEW OF THE ORIGINAL FB VERSION

Gameplay of FB:
The player controls a bird that needs to fly through a series of pipes without hitting them. The bird moves forward automatically, and the player must tap the screen to make the bird flap upwards and avoid colliding with the pipes. The longer the player keeps the bird in the air, the more points they earn. One mistake - and the game ends.

Interface of FB:
On the screen, a bird is displayed in the center, with green pipes above and below that the player needs to fly through. The player's score is shown at the top of the screen.

Colors of FB:

- Green pipes;
- Red one-eyed bird;
- Blue sky;
- Brown ground;
- Green grass.

Controls:
When you tap the smartphone screen, the bird makes a jump.

(A screenshot from the original game is attached as an APPENDIX).

4 DEVELOPMENT REQUIREMENTS

4.1 FUNCTIONAL REQUIREMENTS

1. Scores and ranking:
   - Keep track of scores based on the number of obstacles passed;
   - Display the player's best result (Best Score).

2. Progress saving:
   - The game should save the best result.

4.2 TECHNICAL REQUIREMENTS

1. Graphics:
   - The game should have colorful graphics, including detailed modeling of the bird.
   - The game graphics should closely resemble the original version FB.

2. Controls:
   - The game should respond to button presses to control the bird's flight.
   - There should be an option to resume the game.

3. Gameplay:
   - The player should control the bird's flight by pressing a button to make it ascend and releasing it to descend.
   - The bird must navigate obstacles in the form of pipes placed at different heights.
   - The game should end if the bird collides with an obstacle or touches the ground.

4. Organization of input and output data:
   - The program should be able to process user input (button clicks) to control the bird.
   - The program should display graphical elements such as the bird, pipes, score counter, and interface elements on the screen.
   - Display information about the current score and the player's best result.

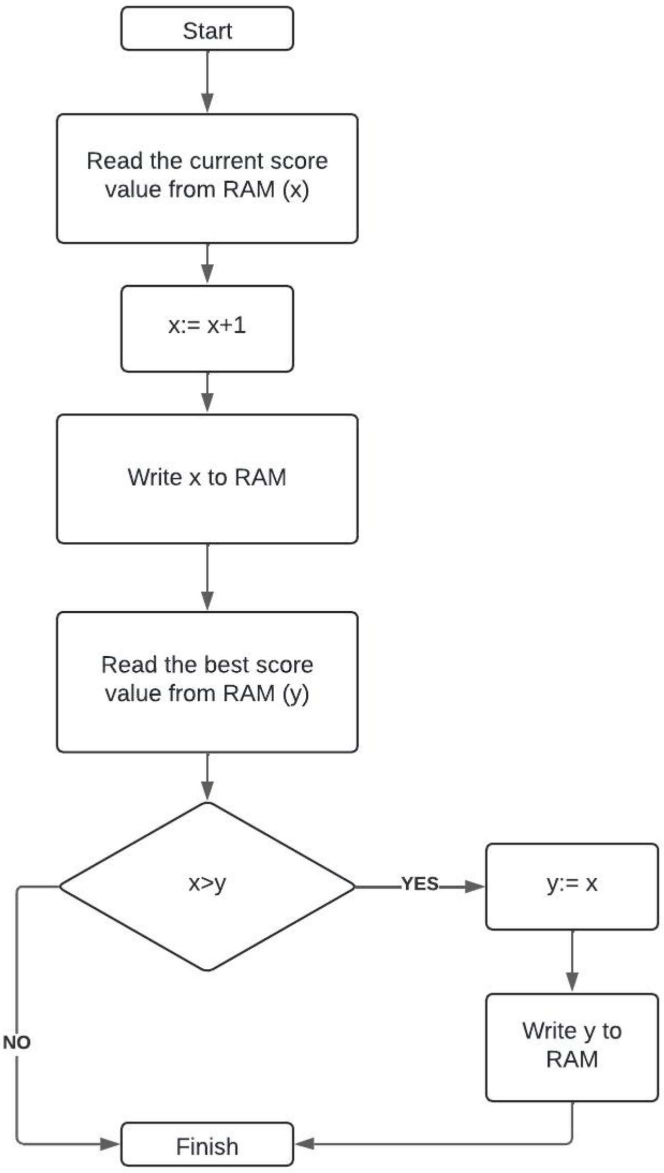
5. Similarities to the original version :
   - The game should visually resemble the original FB game version, specifically: similar to the phone aspect ratio, using original colors.

6. Interface elements:
   - The game should have a Start Screen.
   - The game should have a Fail Screen.

5 FUNCTIONAL CHARACTERISTICS

To implement the software part of our game, we used the assembly language of the CdM-8 processor in the integrated development environment CocoIDE, which is specifically designed for developing code that will be executed by this processor.

The processor in the project is needed to calculate the player's score. The block diagram of the algorithm is shown below.


Start
Read the current score value from RAM (x)
x:= x+1
Write x to RAM
Read the best score value from RAM (y)
x>y
YES
y:= x
Write y to RAM
NO
Finish

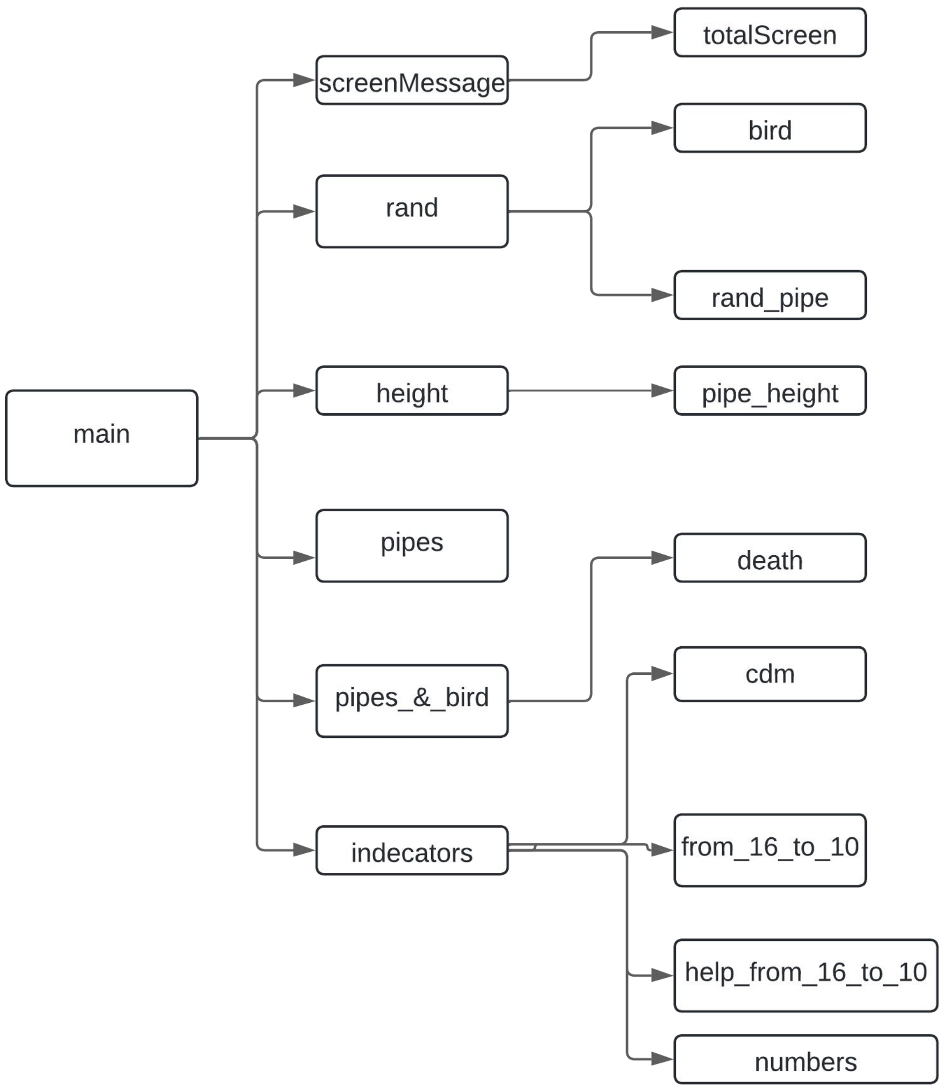
Scheme №1 «Block diagram of the scoring algorithm»

The detailed assembly code can be found in the explanatory note appendix.

6 TECHNICAL CHARACTERISTICS

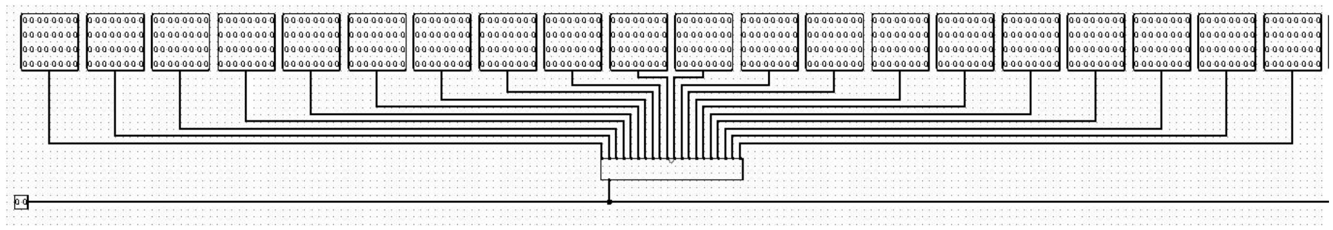
The hardware part of our project consists of interconnected digital logic circuits created in the Logisim program. Let's look at our developments.

Below is a diagram of the connections between subcircuits in "Digital Flappy Bird"



Scheme №2 «Connection of subcircuits between each other»

6.1 screenMessage



Screenshot №1 part of the «screenMessage»

Input screenMessage circuit:

- single 2-bit number.

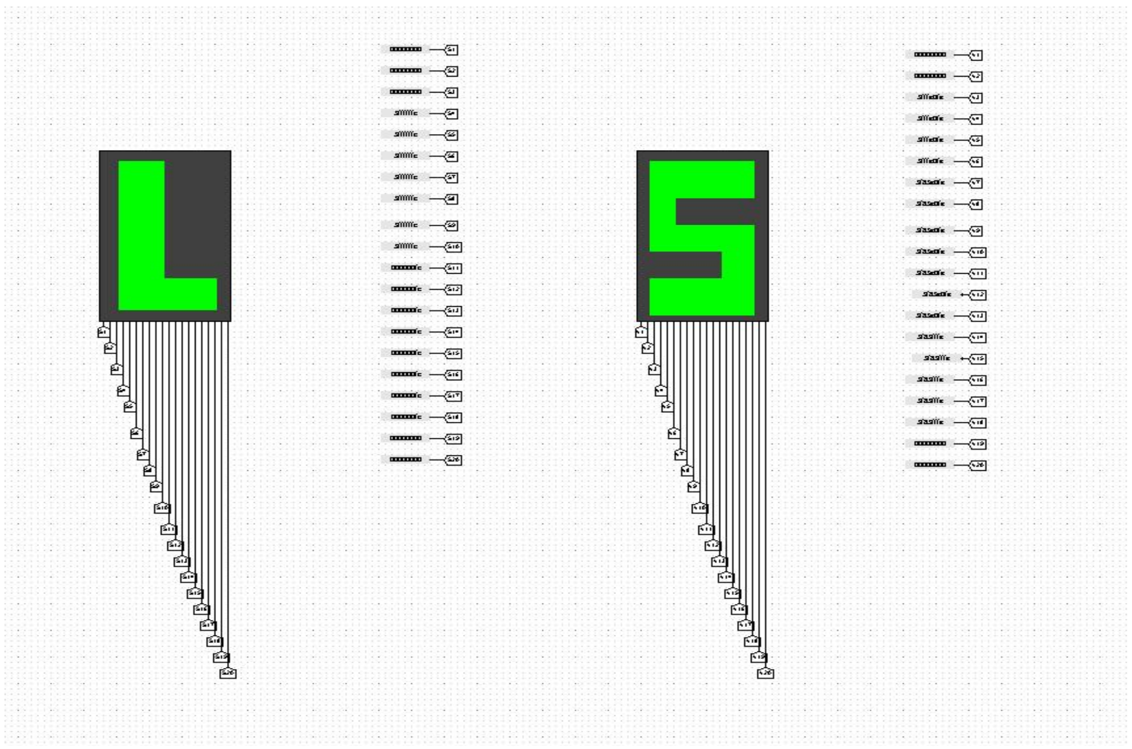
Outnput screenMessage circuit:

- one hundred 32-bit numbers;

This circuit generates a message about the state of the game according to the input parameters. It is possible to output two messages to the five matrices of the information display of the scheme "main": "START" or "LOSS". It is also possible to turn off the information display matrices.

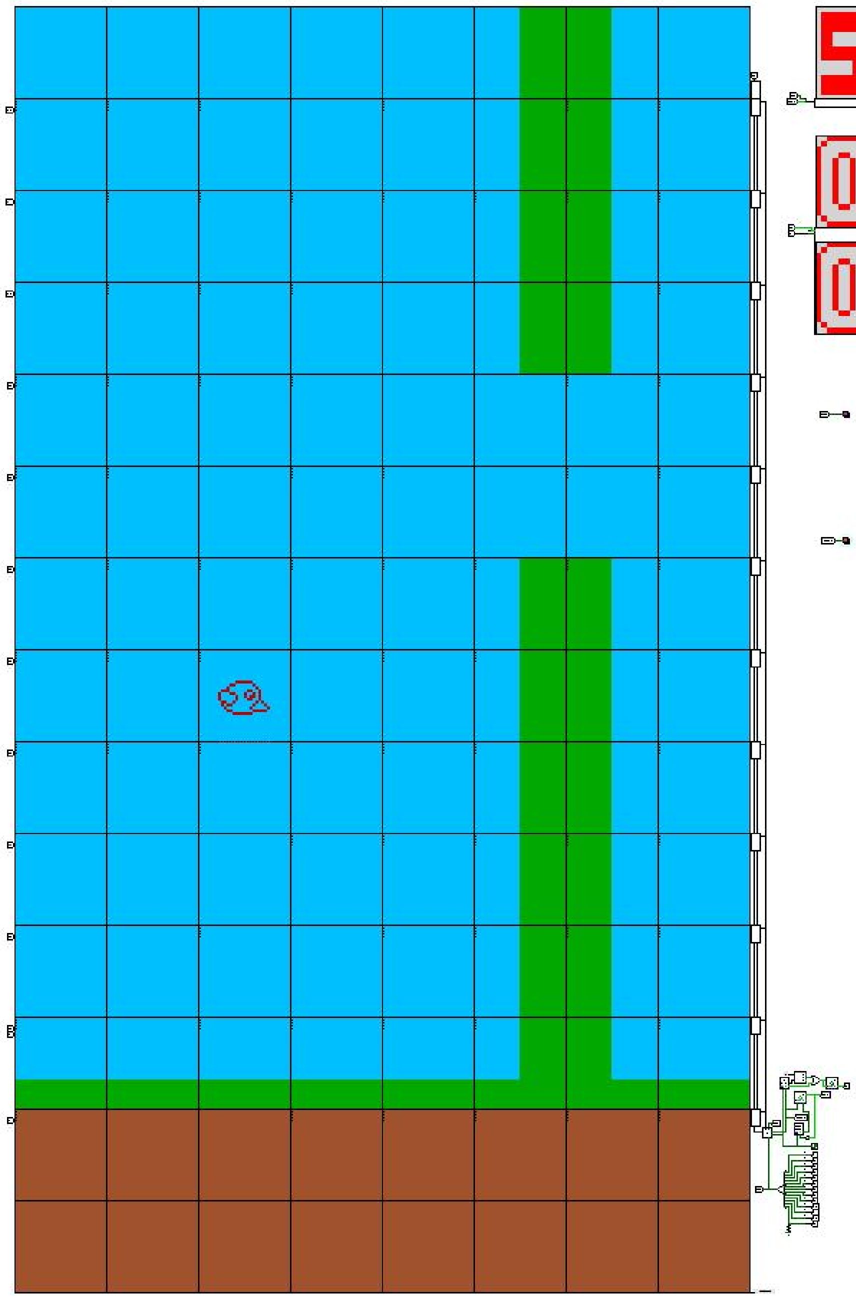
On a separate matrix of the information display of the "main" circuit one of the predetermined characters is displayed.

The values that form one letter on the information display matrix are initially set in the corresponding subsegments. ("S/L", "t/o", "a/s", "r/s", "t/!").



Screenshot №2 part of the «S/L»

6.2 main



Screenshot №3 «main circuit»

Scheme main serves as the interface of the Digital Flappy Bird game.

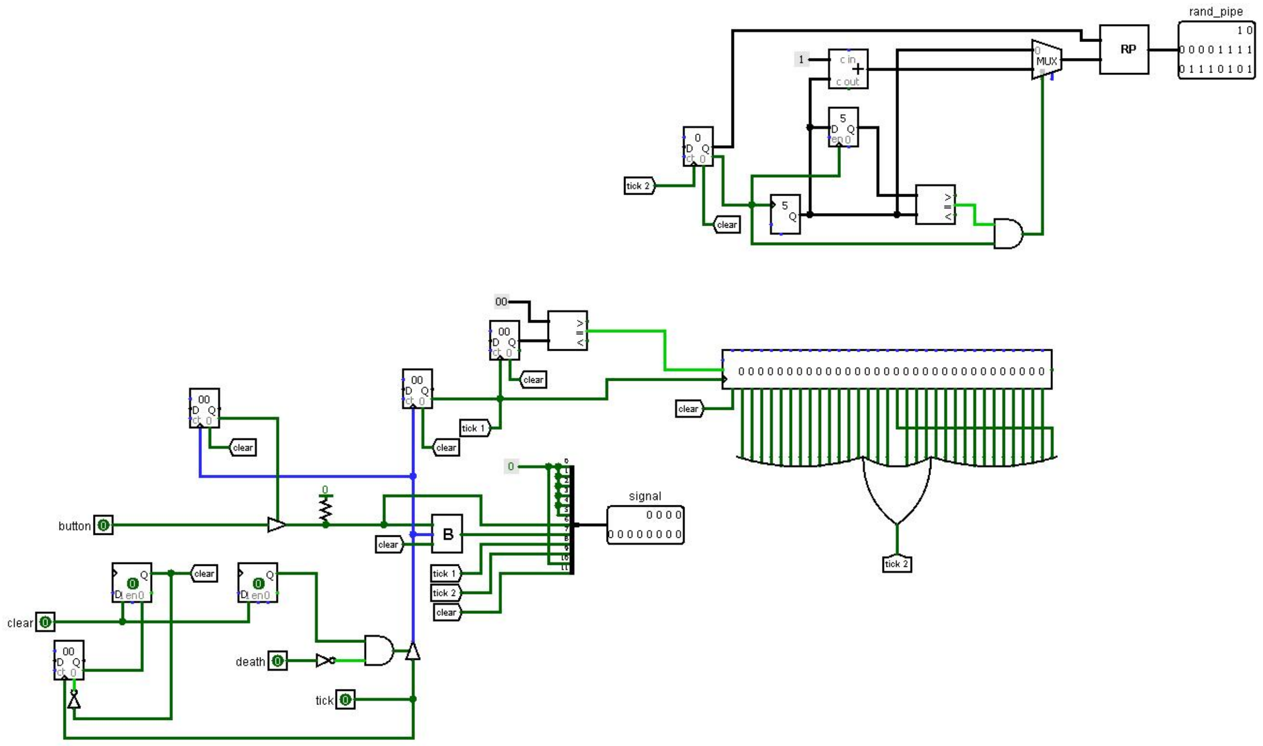
main consists of :

- Clock;
- Display consisting of 14*8 (112) matrices of dimension 32*32;
- 5 matrices of 32*20 pixels for displaying the message "START/LOSS" to the user about the state of the game;
- 3 matrixes of 32*20 pixels for displaying information about the current score;

- 3 matrices of 32*20 pixels to output information about the best score since the simulation was turned on;
- Button to control the bird's jump (then we call the "jump" button);
- Button to start/restart the game (then we call the "start/restart" button);
- Block for tracking the state of the game.(Screenshot 3 of the application)
- 1 rand scheme
- 13 schemes height
- 84 schemes pipes
- 12 schemes pipes_&_bird
- 1 scheme totalScreen (scheme is located in circ library "screenMessage")
- 1 scheme indicators

The user's work with the game takes place in the main scheme. The rules for starting the game will be described in the explanatory note appendix.

6.3 rand



Screenshot №4 «rand circuit»

Input rand circuit:
- Clocks;
- Signal from the "jump" button;
- Signal from the "start/restart" button;
- Death flag.

Output rand circuit:

- 18 bit number. By this number, a particular pipe in the pipe_height scheme will be rendered;
- 12 bit number, which is a combination of one 4 bit number and 8 single bit values:
    a) 0-3 bit number is used in to denote the matrix layer number;
    b) 4 and 5 bits are used to generate the bird movement signal;
    c) 6 bit to track the “jump” button;
    d) 7 bit to track the bird's fall and jump velocity clock cycles;
    e) 8 bit to track the clock speed of pipe movement;
    f) 9 bit to monitor the appearance of the pipe on the display ;
    g) 10 bit to track the death of the bird;
    h) 11 bit to track the “start/restart” button.

When you first press the "start/restart" button, the "Restart" flag is turned on for a certain time, which clears all counters and registers, as well as goes to the subcircuit "bird" and the output signal. The flag (right D-trigger) is also turned on, which, when the flag "death" is off, starts the game. When the "start/restart" button is pressed repeatedly, the same thing happens, except that the game start flag (right D-trigger) is activated.

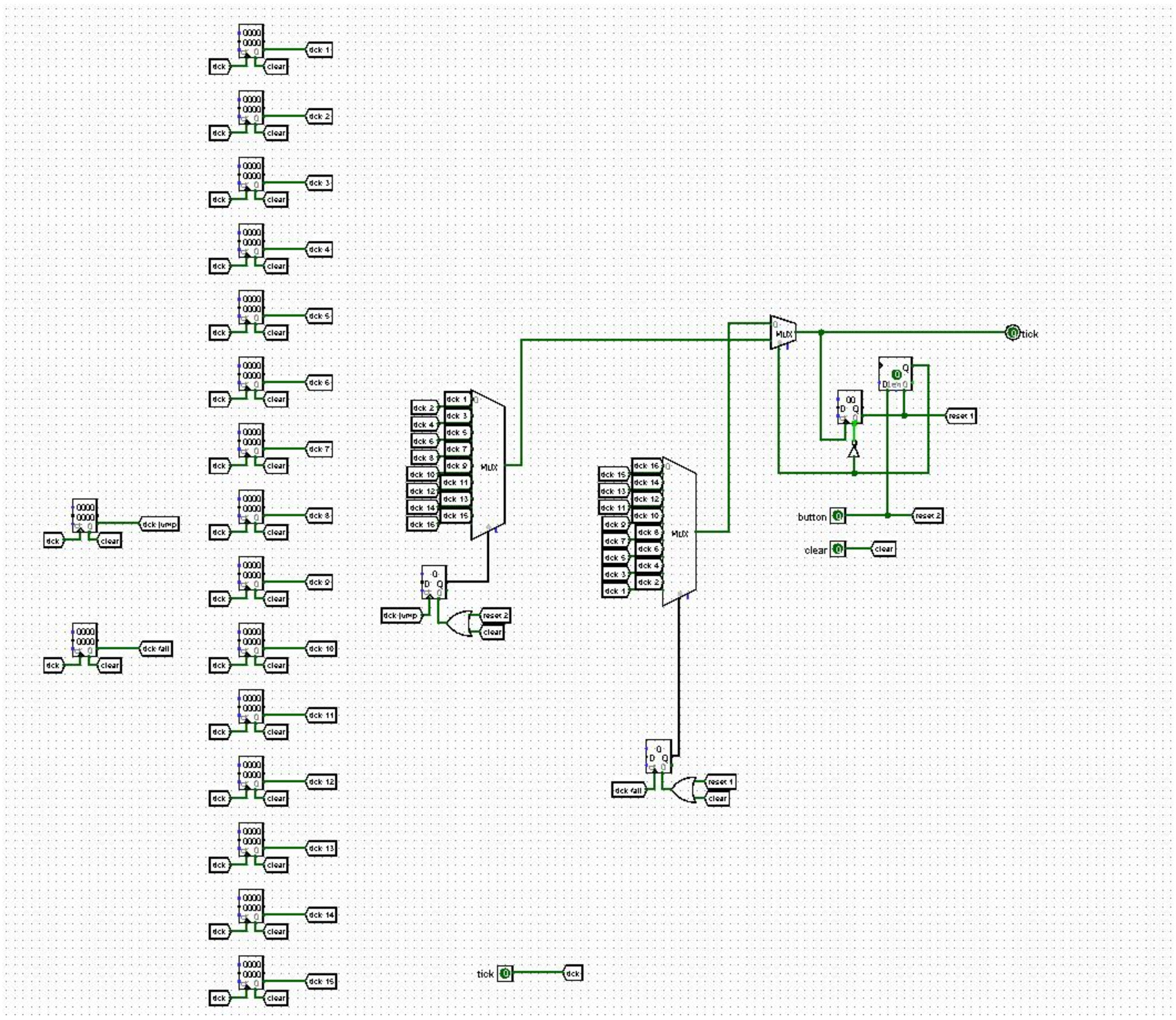
We made sure that during the first 255 clock cycles after the start/restart of the game, pressing the "jump" button will have no result (thanks to the counter). On further presses, the "jump" button signal is sent to the "bird" subcircuit and the output signal.

With the help of the clock frequency divider we get the clock cycles that set the speed of pipe movement on the display of the circuit "main".

With the help of the second divider we get the frequency of appearance of pipes.
With the help of a shift register also the thickness of pipes.

With the help of the counter and random number generator, the input values for the rand_pipe scheme are obtained. Since the random number generator works with 3 bit values, there are often situations of repetition. To solve this problem, a register storing the previous value of the random number is provided. When the previous value and the newly obtained value are equal, the latter is incremented.

6.4 Bird



Screenshot №5 «bird circuit»

Input bird circuit:

- Clocks;
- Signal from the "jump" button;
- Signal from the "start/restart" button;

Output bird circuit:

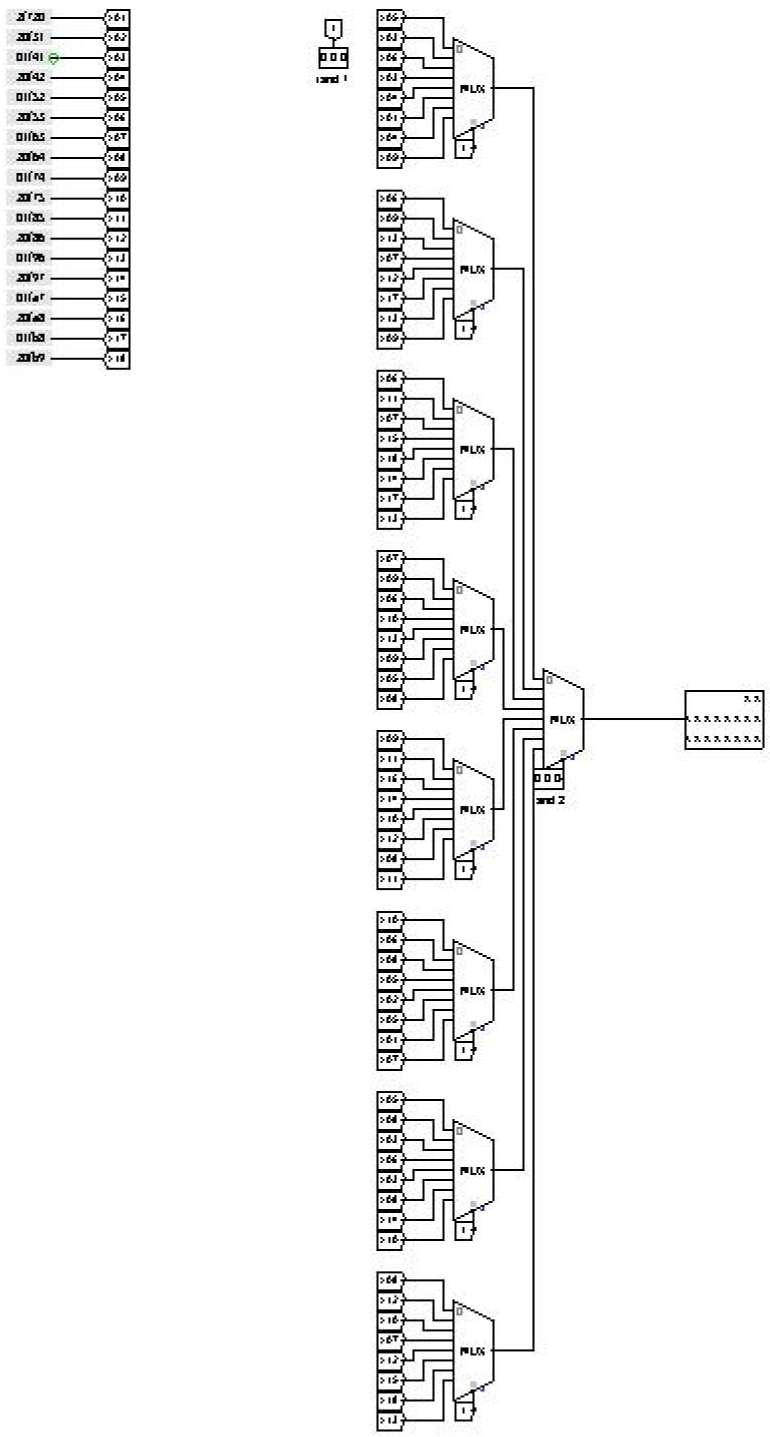
- Clocks of jumping or falling.

The bird circuit is responsible for accelerating the fall of the bird's jump. This circuit divides the clock by means of Counters and gradually switches between frequency dividers by means of 4 bit Multiplexers.

When the "jump" button is pressed, there is a temporary switch from the "Drop Multiplexer" to the "Jump Multiplexer" and a momentary reset of the value of the "Jump Multiplexer". After a certain time has elapsed, the Multiplexers are switched back and the value of the "Drop Multiplexer" is momentarily reset.

When the "start/restrart" button is pressed, the values of the Multiplexers and the clock dividers are reset.

6.5 rand_pipe



Screenshot №6 «rand_pipe circuit»

Input rand_pipe circuit:

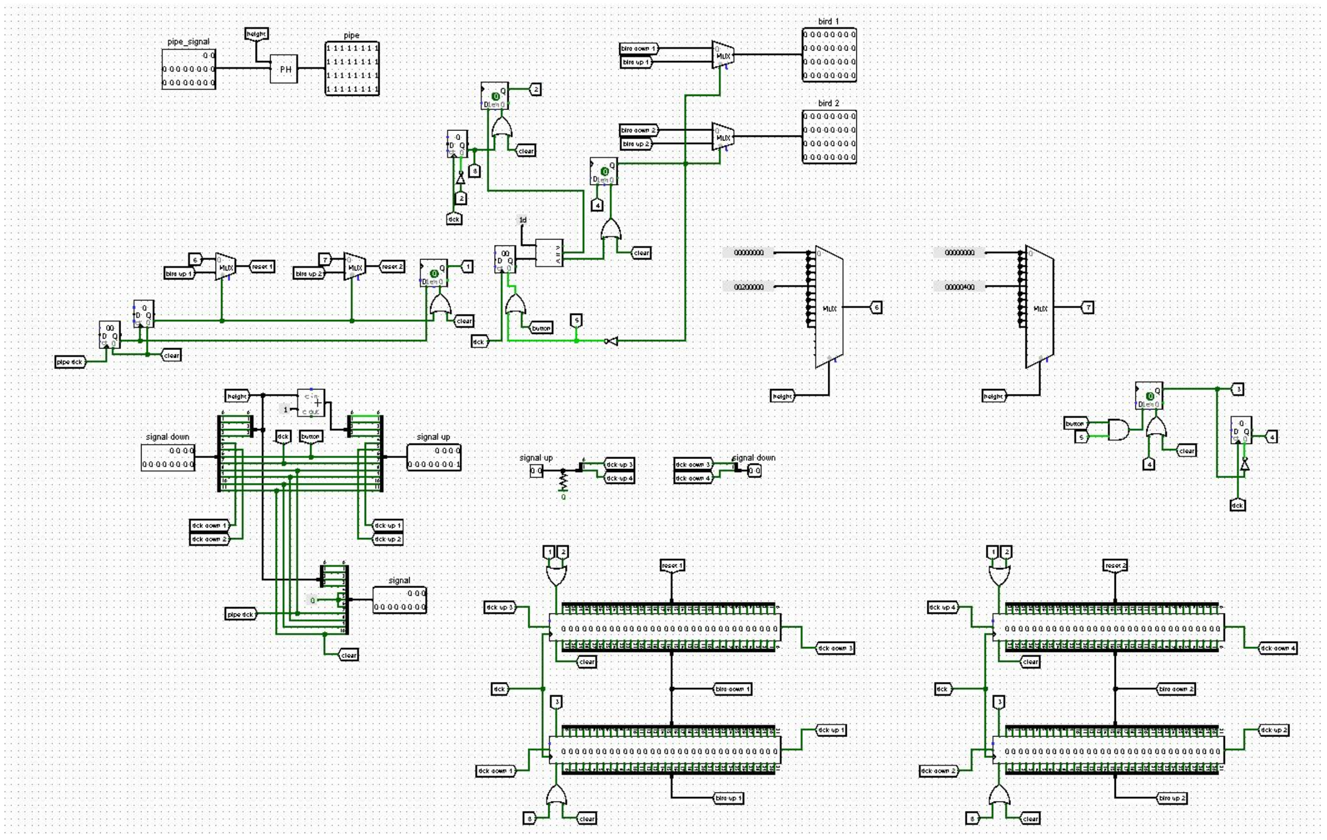
- 3 bit number (rand1), which is responsible for randomly selecting one of the 8 pipe combination patterns;
- 3-bit number (rand2), which is responsible for the serial number of the pipe from the selected pipe combination template;

Output bird circuit:

– 18-bit number. This number will be used to draw a specific pipe from the selected template in the pipe_height circuit;

The rand2 in a three-bit Multiplexer selects one of the eight pipe combination patterns. In turn, rand1 in another three-bit multiplexer alternately selects one of the eighteen constants (from the selected pattern), which is responsible for the desired pipe. The resulting number is passed to the output of the circuit.

6.6 height



Screenshot №7 «height circuit»

Input height circuit:

– 12 bit number (signal) (described in detail in 6.1 rand);

– 2 bit number , for top-down movement of the bird;

– 18 bit number :

a) 0-3 bits - set the number of the matrix layer in the main circuit display , where the lower boundary of the pipe will be;

b) 4-7 bits - set the number of the matrix layer on the main circuit display, which will be the upper boundary of the pipe;

c) 8-17 bits - form the pipe opening on the corresponding matrix layers of the "main" scheme display.

Output height circuit:

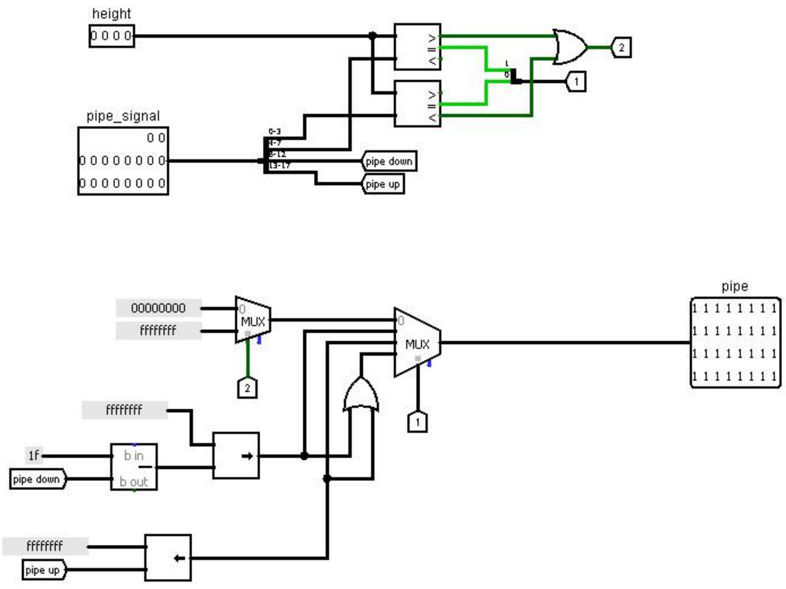
- 12 bit number (described in detail in 6.1 rand);
- 11 bit number
  a) 0-3 bits - denote the layer number of matrices in the “main” circuit display;
  b) 4-6 bits - denote the number of matrix in the layer;
  c) 7 bits - clock cycles that specify the speed of pipe movement;
  d) 8 bits - indicates the frequency and thickness of the pipes;
  e) 9 bit - death flag
  f) 10 bit - game restart flag
- 2 bit number, for top-down movement of the bird;
- 32 bit number - responsible for rendering of a pipe on a particular matrix layer;
- Two 32 bit values - responsible for moving the bird's model on the “main” scheme display column

The 18 bit number received at the input of this circuit is sent to pipe_height together with 0-3 bits of signal. Then the result is sent to the pipes circuit.

When the signal from the "start/restart " button is received, all counters, d-triggers and shift registers are zeroed. Then the values of the upper and lower boundary of the bird model are loaded into the shift registers. In the shift registers the values are moved at the rate given by the "bird" circuit.

When the "jump" button is pressed, values are loaded from the shift registers into other shift registers in the opposite direction. Due to the reverse direction of the shift registers, the direction of movement of the bird is reversed. After a certain time, the values of the upper and lower boundaries of the bird model are loaded into the initial shift registers and the direction of movement becomes the original direction.

6.7 pipe_height



Screenshot №8 «pipe_height circuit»

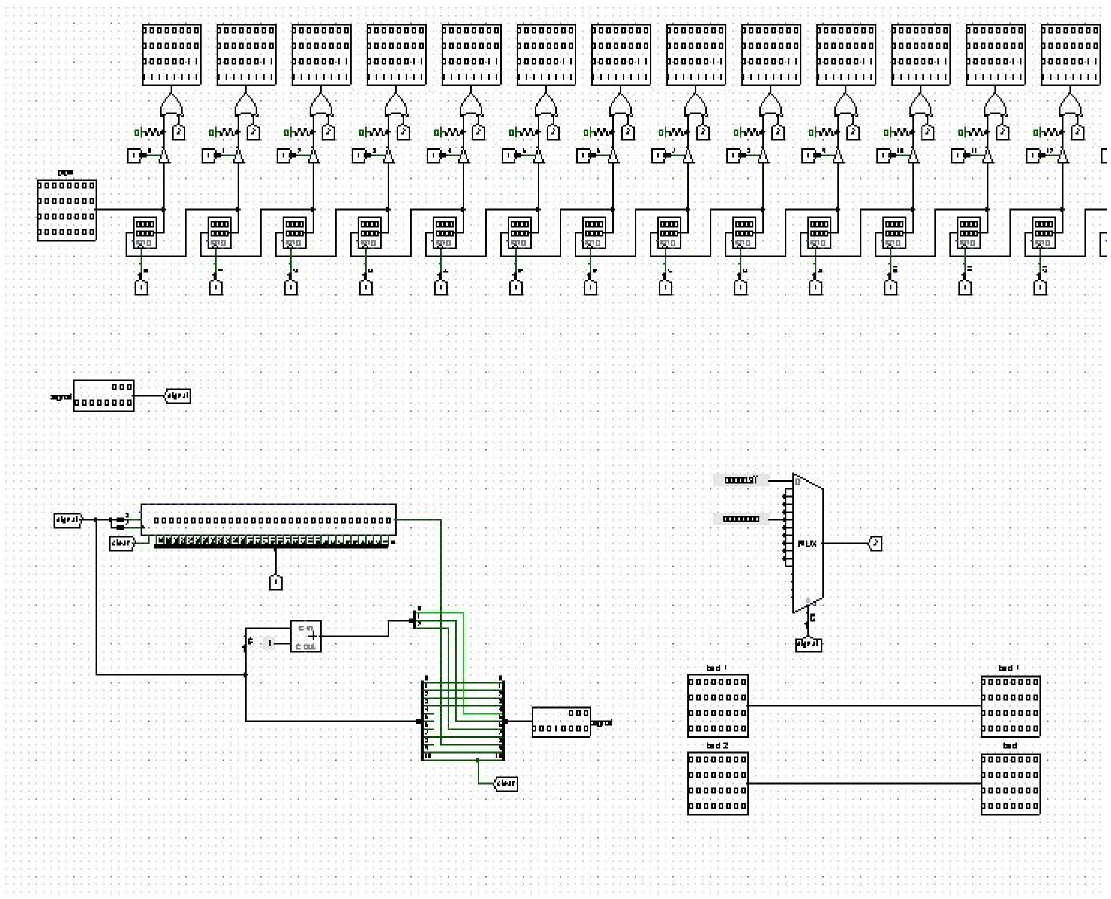
Input pipe_height circuit:

- 18 bit number:
  a) 0-3 bits - indicate the number of the matrix layer on the “main” circuit display, where the lower boundary of the pipe will be;
  b) 4-7 bits - designate the number of the matrix layer on the “main” circuit display, on which will be the upper boundary of the pipe;
  c) 8-17 bits - form the pipe opening on the corresponding matrix layers of the “main” scheme display.
- 4 bit number , defines the number of the matrix layer on the display of the “main” scheme on which the pipe_height scheme is located.

Depending on the input parameters, the output will be one of five options:

- fully off matrix;
- fully switched on matrix;
- the lower part of the matrix is switched on;
- the upper part of the matrix is switched on;
- The upper and lower parts of the matrix are switched on, and there is a switched off part of the matrix.

6.8 pipes



Screenshot №9 part of the «pipes circuit»

Input pipes circuit:

- 11 bit number (signal) (obtained as a result of the "hight" circuit) ;
- 32 bit number (pipe), which is responsible for the pipe value on a particular matrix;
- Two 32 bit numbers, which are responsible for transmitting the bird motion signal by pipes and pipes_&_bird schemes.

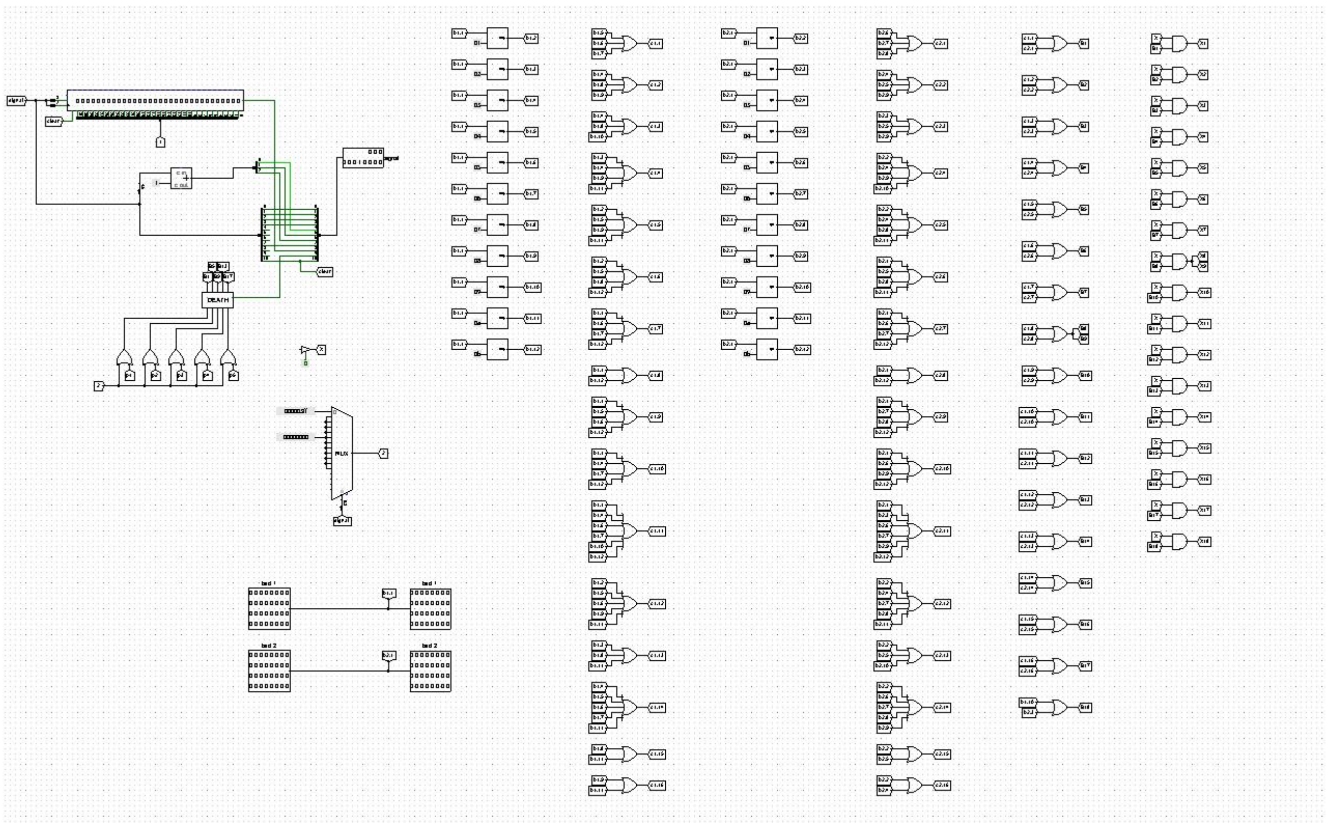
Output pipes circuit:

- 11 bit number (signal) (obtained as a result of the "hight" circuit) ;
- 32 bit number (pipe), which is responsible for the pipe value on a particular matrix;
- Two 32 bit numbers, which are responsible for transmitting the bird motion signal by pipes and pipes_&_bird schemes;
- Thirty-two 32-bit numbers that specify the image on the “main” circuit display matrix.

The seventh bit "signal" determines the speed of movement of the value set by bit 8 "signal" through the shift register. Bit 8 is responsible for moving the "pipe" through the thirty-two registers, as well as the activation of signals from these registers. The signal from each register is combined with a multiplexer signal, which determines the lower boundary of the available flight area by the number of the matrix layer.

When the signal from the "start/restart" button is received, the shift register is cleared, thus zeroing all output values on a particular matrix.

6.9 pipes_&_bird

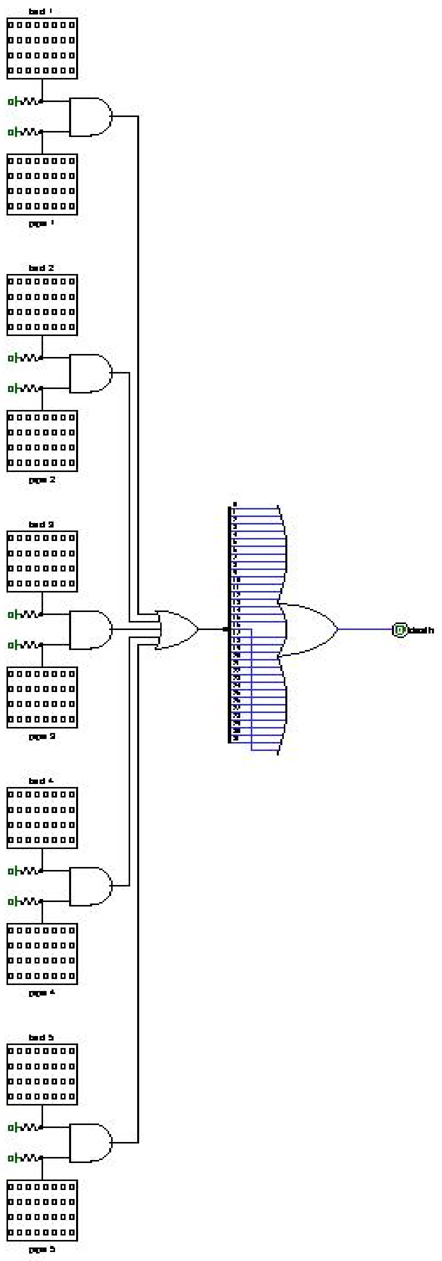


Screenshot №10 part of the «pipes_&_bird circuit»

This scheme is an improved version of the "pipes" scheme. The following is a description of the differences of this scheme.

Using special combinations of right shifts "bird1" and left shifts "bird2", identical columns are obtained, which are then merged (overlapping of the two images is necessary for correct rendering of the bird model at the junction of matrices). Some of these columns are used in the "death" scheme together with their corresponding outputs of 32-bit registers that specify the movement of the pipes (more details about them are given in 6.7 pipes). All columns combined with the 32 bit undefined value (to get red color on the matrix, "error color") are combined with the outputs of eighteen registers to output the image on the matrix.

6.10 death



Screenshot №11 «death circuit»

Input death circuit:

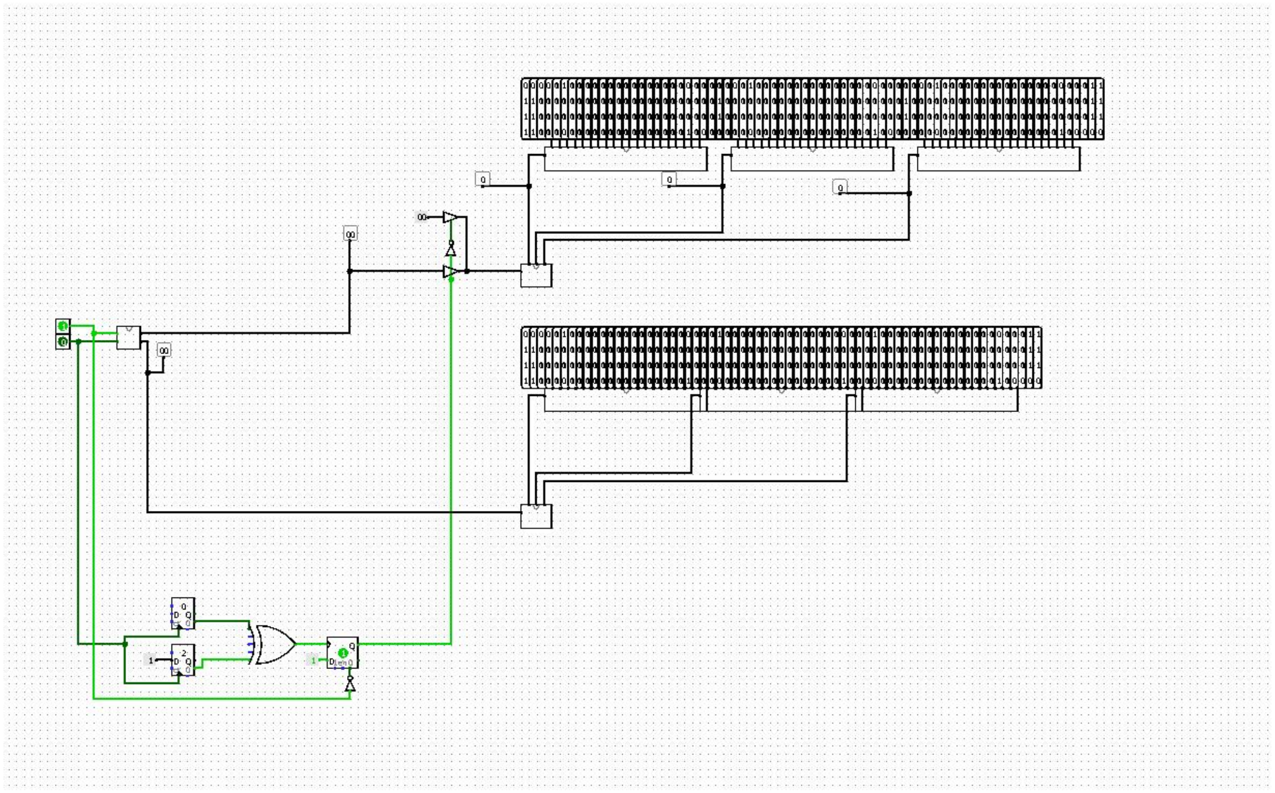
- Five 32 bit numbers, each number is responsible for a column of bird on the matrix (the bird moves only one column of the "main" circuit display);
- Five 32 bit numbers, each number is responsible for a pipe column on the matrix.

* numbers are taken from a single pipes_&_bird component.

Output death circuit:

- 1 bit value (Death Flag), 0 - the bird is alive, 1 - the bird is dead.

6.11 indecators



Screenshot №12 «death indecators»

Input indecators circuit:

- 1 bit number - game state flag (whether the bird is alive or not);
- 1 bit number - reports a +1 to the score;

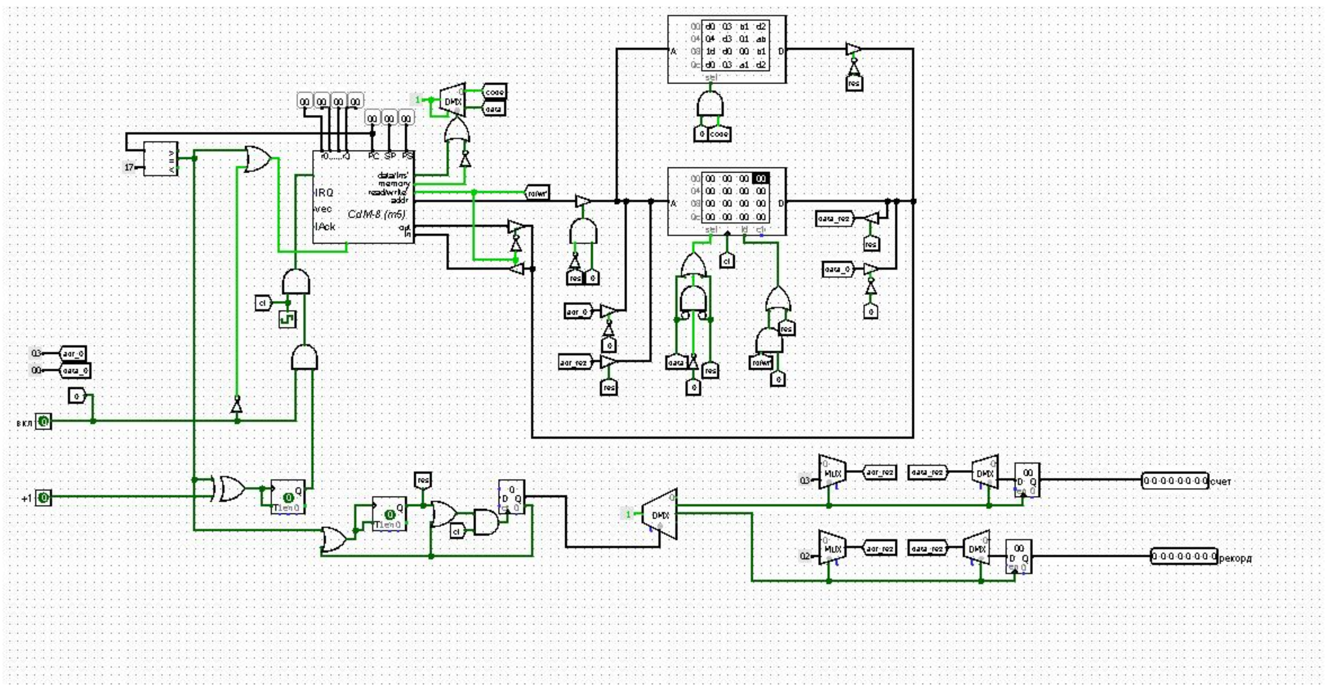
Output indecators circuit:

- One hundred twenty 32 bit values are required to display the currrent score and best score states on the display matrices of the "main" circuit (derived from the “numbers” circuit)

This circuit passes two one-bit values to the "cdm" circuit (the operation of the "cdm" circuit is described in section 6.12 "cdm"). The values obtained after the operation of the "cdm" circuit are passed to the "numbers" circuit. The scheme "numbers" outputs the numerical values necessary for displaying on the display matrices of the scheme "main" the current score and the best score in the game in decimal notation system.

The "indecators" circuitry links the CdM-8 processor to the main circuits of the game.

6.12 cdm



Screenshot №13 «cdm circuit»

Input indecators circuit:

- 1 bit number - game state flag (whether the bird is alive or not);
- 1 bit number - reports a +1 to the score;

Output indecators circuit:

- two 8-bit numbers, which are responsible for the current score and the best score in the game.

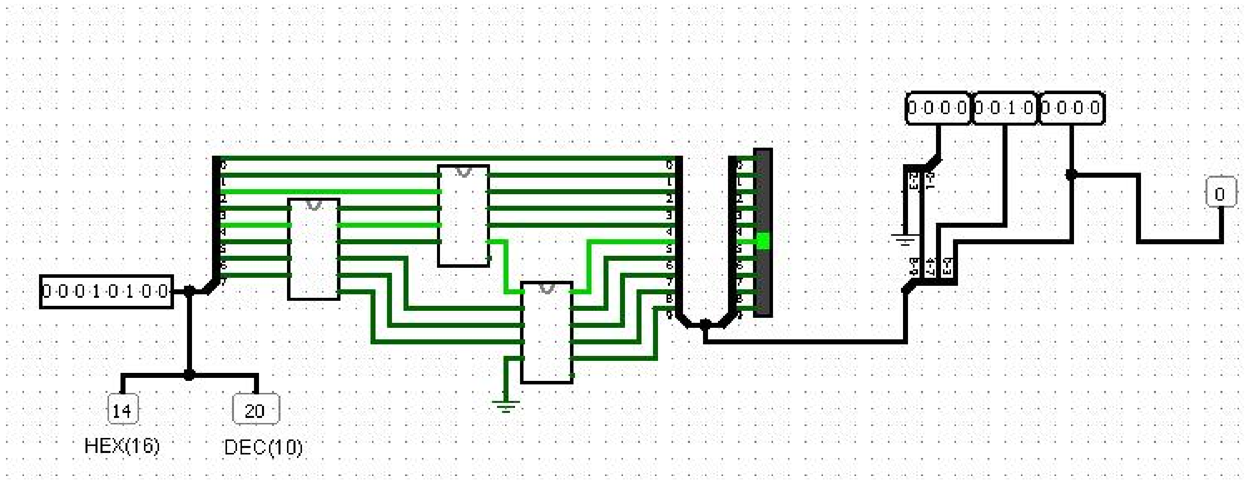
This circuit represents the CdM-8 processor and the elements necessary for its correct operation in our game. CdM-8 is connected according to the Harvard system. The processor performs the scoring function in the game, namely it controls the current score and keeps track of the best score.
CdM-8 works when the game is active and the bird has flown over an obstacle (two incoming signals to the "cdm" circuit are active). In other cases, the clock is silenced and the processor is not powered.
the cdm circuit has two input signals:

The cdm circuit has two input signals. In the case when both contacts are equal to one, the program is launched in ROM, the current score and the best score are updated.

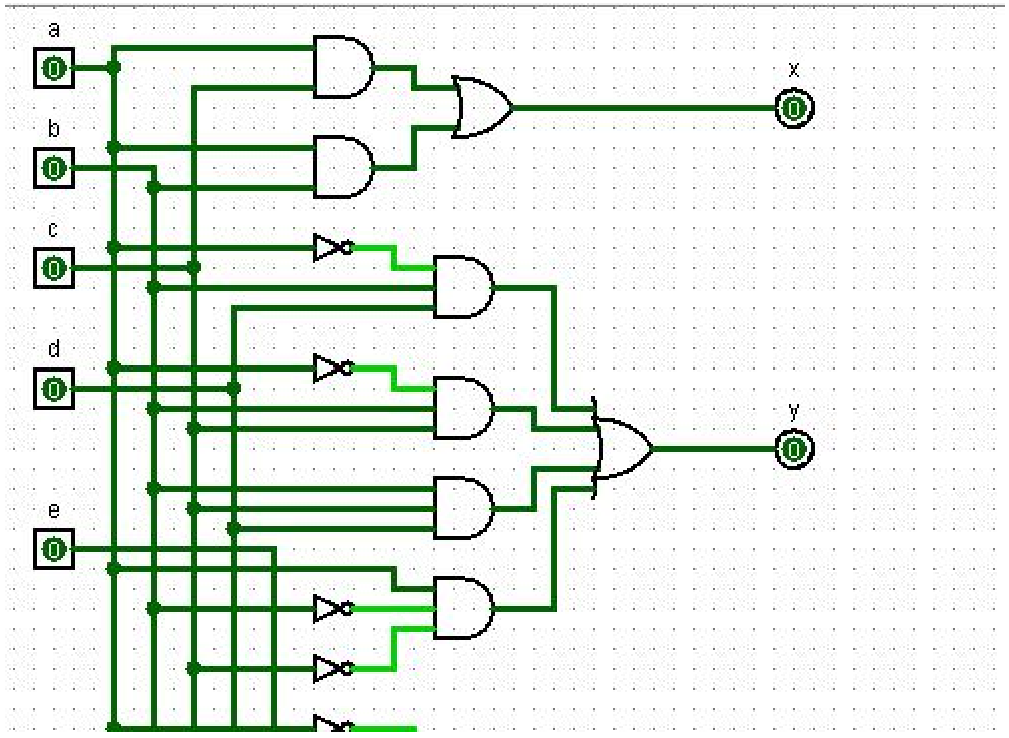
After executing the program, the values in RAM are output to the input, which are responsible for the current score and the best score.

The software part of the processor operation was described in paragraph 5 FUNCTIONAL CHARACTERISTICS.

6.13 from_16_to_10 and help_from_16_to_10



Screenshot №14 «from_16_to_10 circuit»



Screenshot №15 part of «help_from_16_to_10 circuit»

These schemes will be considered together because help_from_16_to_10 is auxiliary to the from_16_to_10 scheme

Input from_16_to_10 circuit:

– single 8-bit number.

Output from_16_to_10 circuit:

– three 4 bit values, each value is responsible for a digit of the incoming number in 10 number system.

Input help_from_16_to_10 circuit:

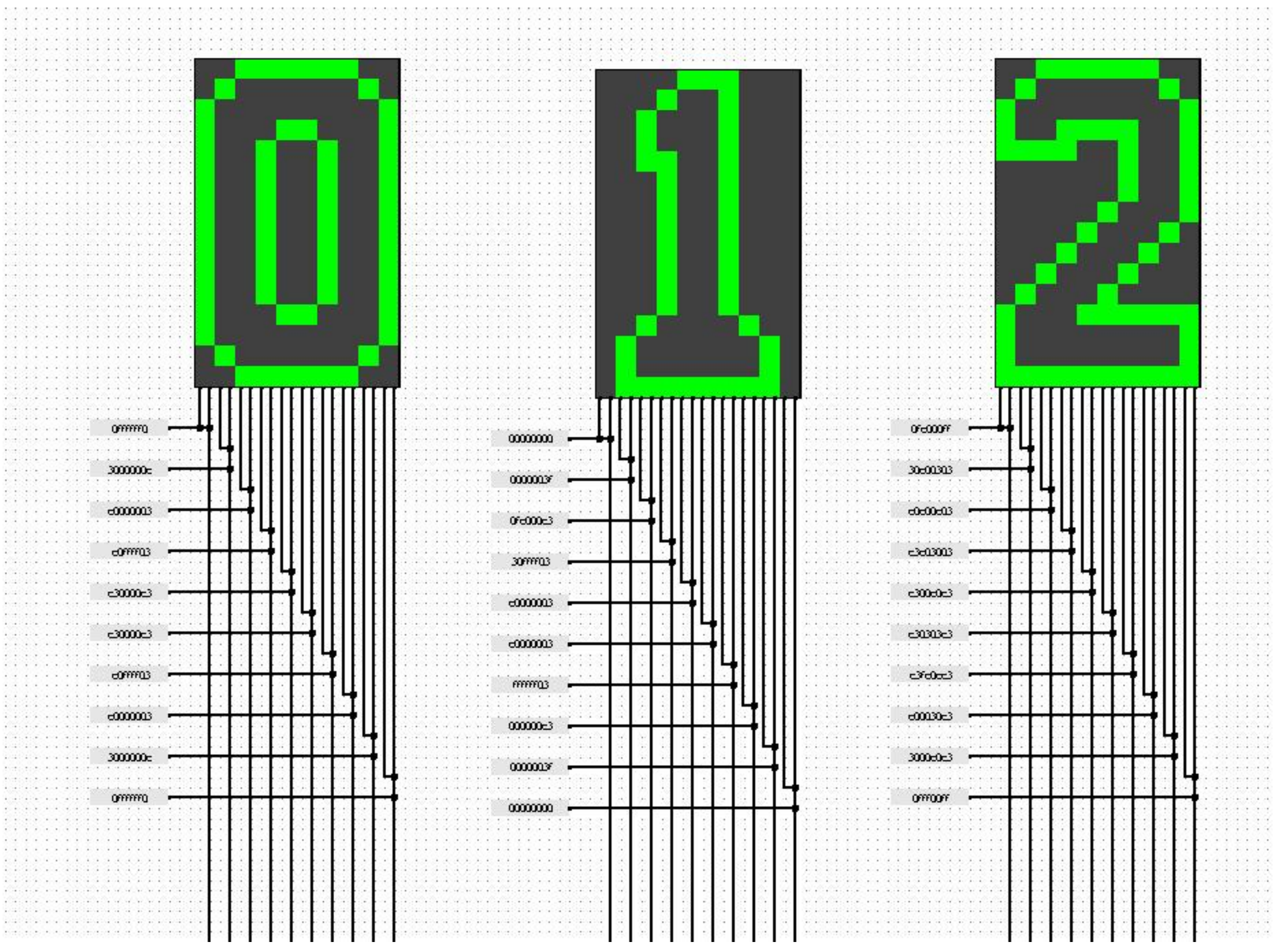
– five one bit values.

Output help_from_16_to_10 circuit:

– six one bit values.

In order for the user to see the current and best count in decimal notation, it is necessary to split one eight-bit number into three four-bit numbers. To do this, the 8-bit number must be converted to BCD form. These circuits do this job.

6.14 numbers



Screenshot №16 part of the «numbers circuit»

Input indecators circuit:

– The 4-bit value indicates the required digit to form an 8-bit number on the display matrices of the "main" circuit in the decimal notation system

Output indecators circuit:

– twenty 32-bit values are required to display the current score and the state of the best score on the same display matrix of the "main" circuit (derived from the "numbers" circuit).

This scheme acts as a library of pre-prepared numerical values to map information onto the matrices of the "main" scheme.

7 CONCLUSION

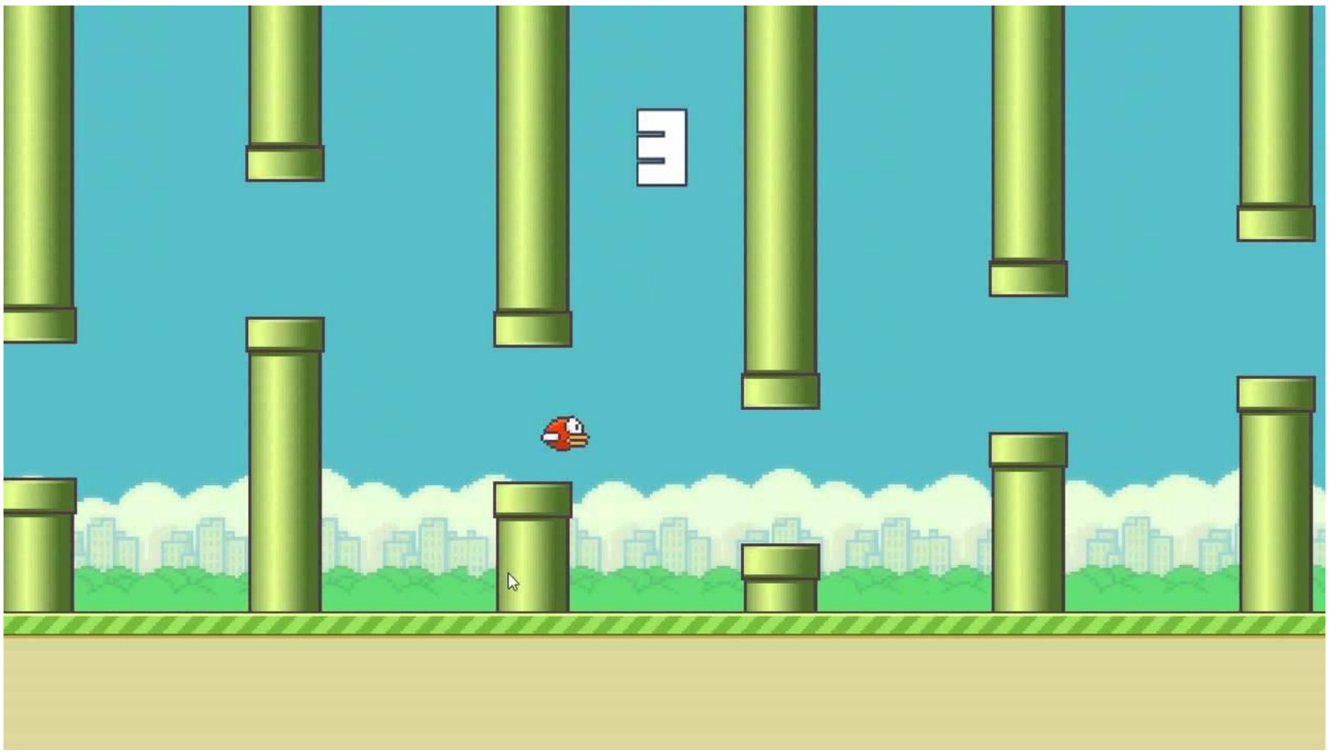
During our collaborative work, we successfully developed a game called "Digital Flappy Bird".
The game is implemented using electronic circuits in which we utilized the CdM-8 processor to execute our own software code. All the functional requirements were fully implemented as outlined in the "DEVELOPMENT REQUIREMENTS" section. We believe that our project achieved a high degree of similarity with the original version of FB.
Working on this project allowed us to expand our knowledge in creating electronic circuits, working with a processor built on the Harvard architecture, writing explanatory note and improving our skills in effective teamwork.

8 SOURCES USED IN DEVELOPING

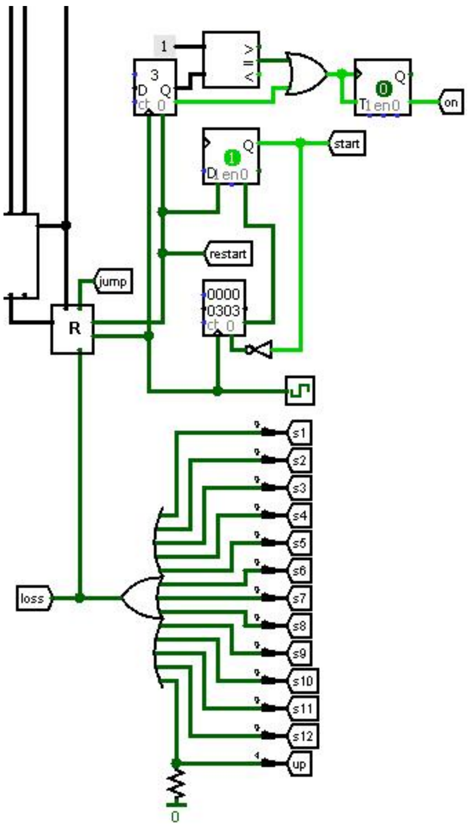
COMPUTING PLATFORMS / A.SHAFARENKO, S.P.HUNT. – 2015.

APPENDIX

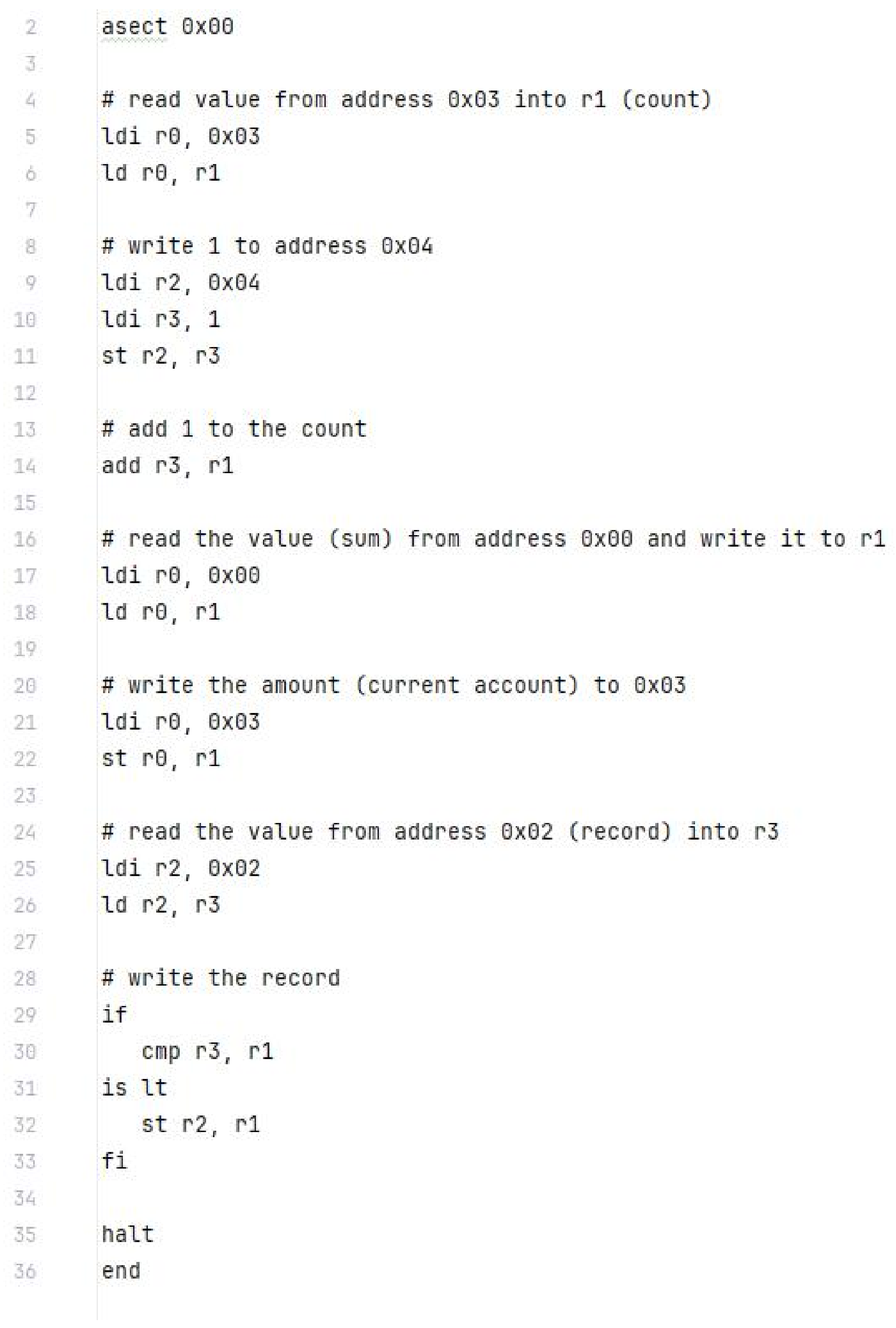
Screenshot №1 «Screenshot from the original game»:



Screenshot №2 «Game State Tracking Unit»:



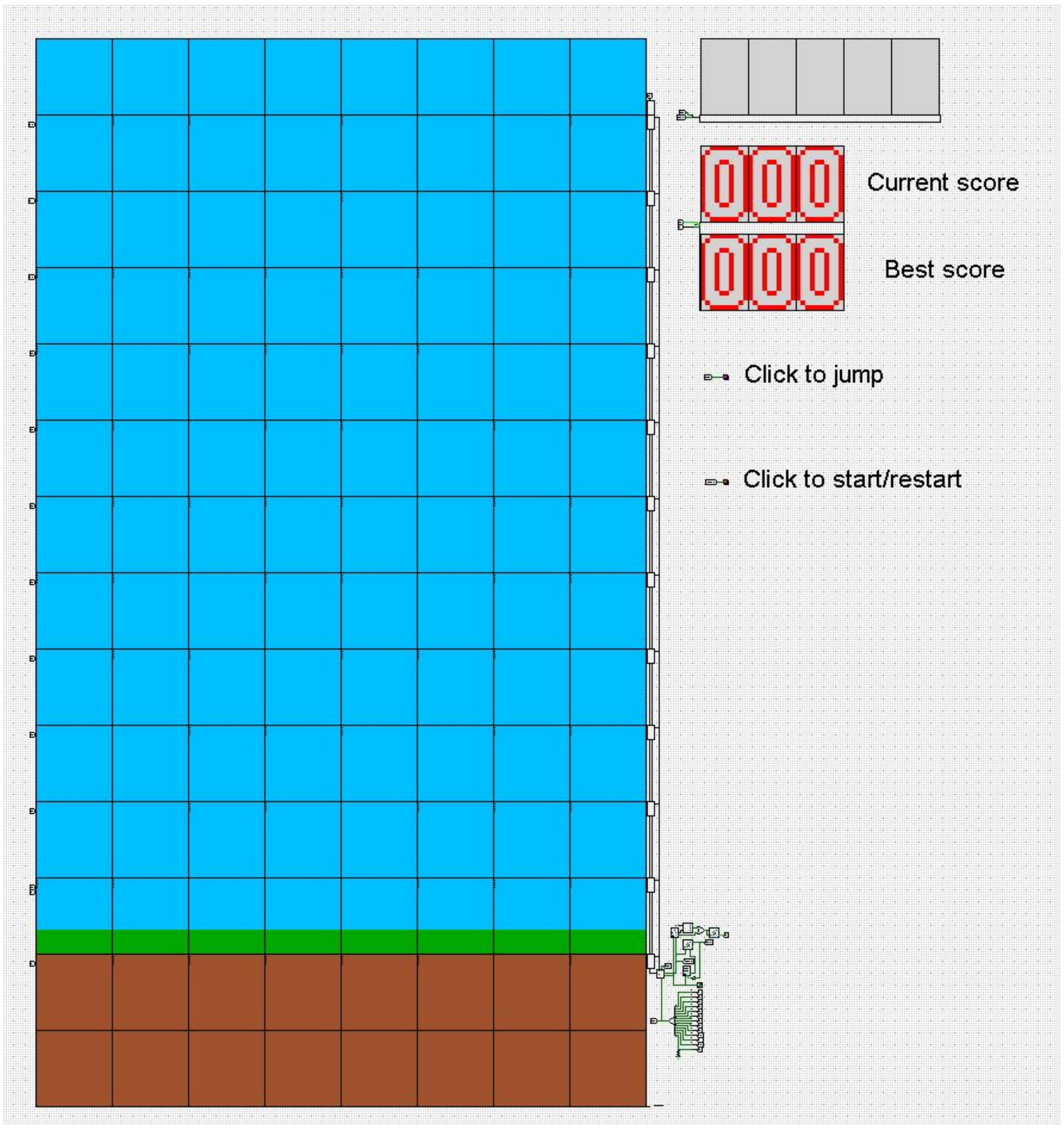
Screenshot №3 «assembly language code for CDM-8



```
asect 0x00

# read value from address 0x03 into r1 (count)
ldi r0, 0x03
ld r0, r1

# write 1 to address 0x04
ldi r2, 0x04
ldi r3, 1
st r2, r3

# add 1 to the count
add r3, r1

# read the value (sum) from address 0x00 and write it to r1
ldi r0, 0x00
ld r0, r1

# write the amount (current account) to 0x03
ldi r0, 0x03
st r0, r1

# read the value from address 0x02 (record) into r3
ldi r2, 0x02
ld r2, r3

# write the record
if
   cmp r3, r1
is lt
   st r2, r1
fi

halt
end
```

USER MANUAL

At the beginning, the user sees the circuit diagram. There is a large display similar to a cell phone screen, a display showing the game status, and displays showing the player's current and best score. There are two buttons on this circuit that the player can use to control the game. Screenshot 3 shows a snapshot of the initial screen of the game.



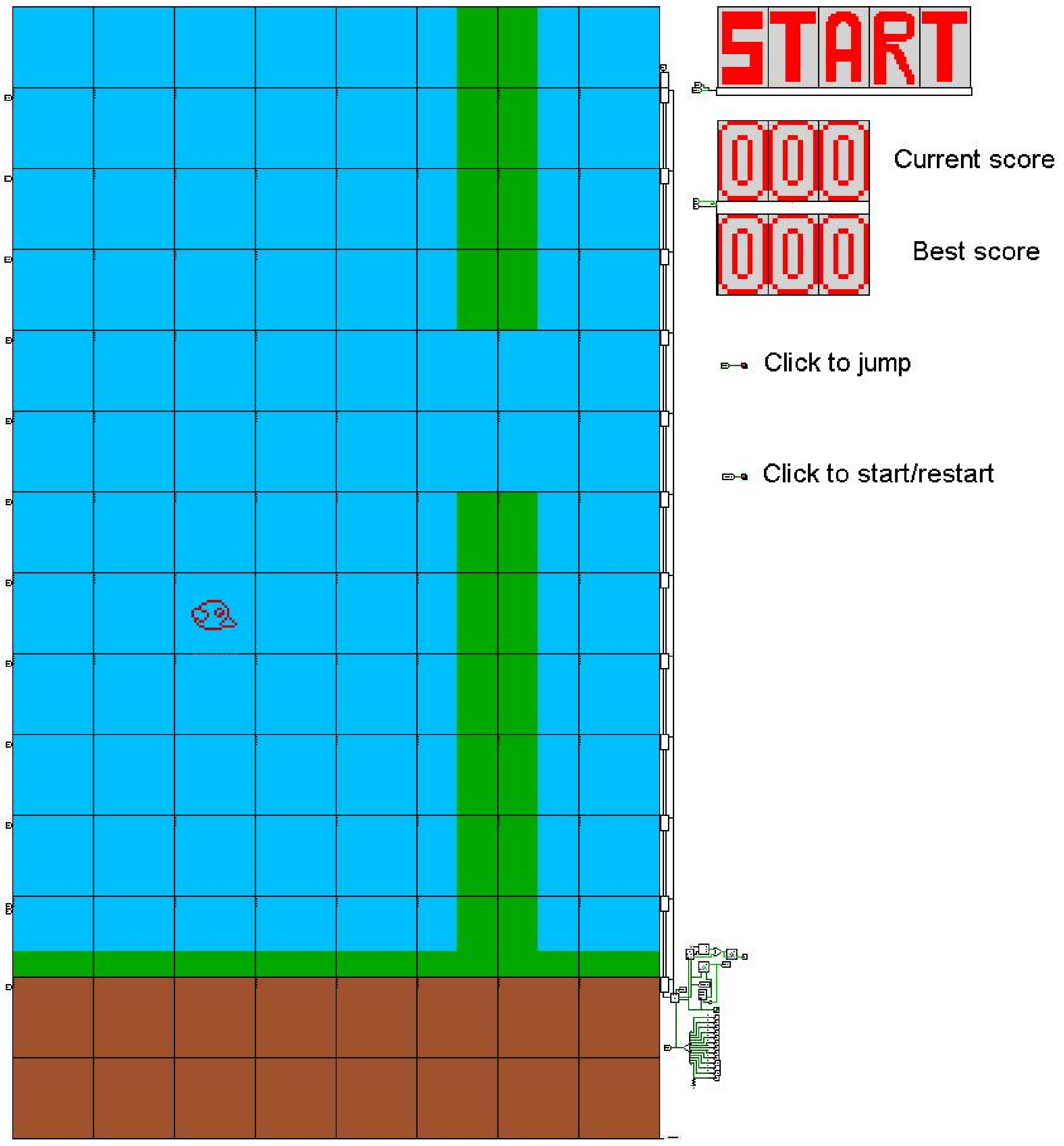
Screenshot №3 «Game start screen»

To run the game, you must enable modeling and clocking (the clock frequency required to run the game is 4.1 kHz).

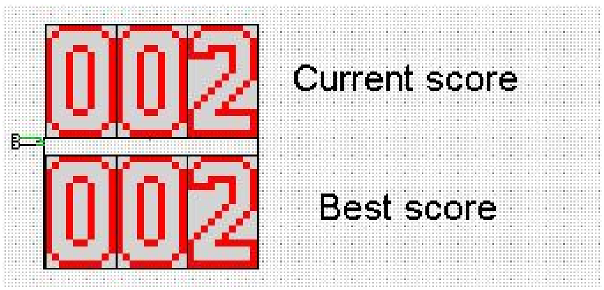
The next step is to press the "start/restart" button. The upper right display will show "START", which informs the user that the game is running. On the playing field will appear a model of a bird, which you need to control , and green pipes (Screenshot №4 «START»).

In order for the bird to take off it is necessary to press the "jump" button

Screenshot №4 «START»



In the upper right corner of the "main" circuit, you can see two screens that will display the current game score and the best score of the game session.

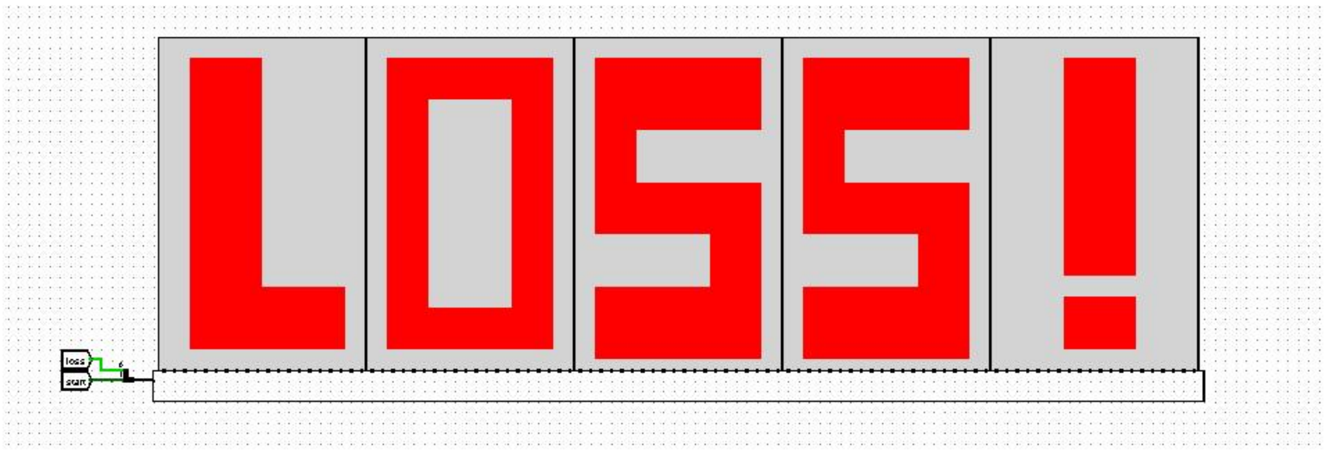


Screenshot №5 «score»

The player will lose and the attempt will end in the following cases:

– the bird crashes into the pipe;
– bird falls on the grass;
– bird hits the ceiling.

When the attempt is over, a "LOSS!" message will appear on the upper right screen of the "main" circuit.



Screenshot №6 «LOSS!»

To restart the game you need to click on the "start/restart" button.
In order for the bird to take off it is necessary to press the "jump" button