MINISTRY OF EDUCATION AND SCIENCE OF THE RUSSIAN FEDERATION

NOVOSIBIRSK STATE UNIVERSITY
(NSU)

Institute of Intelligent Robotics

Specialty: Mechatronics and robotics

Kadilenko Ivan

Kil Dmitriy

Semenov Daniil

First-year study, group 22933

“The Game of TV-Tennis”

CONTENTS

INTRODUCTION

This file describes the game “The Game of TV-Tennis” developed by us. The first version of this game, developed in 1969, was one of the first computer games in the world.

The goals of our project: to recreate “The Game of TV-Tennis” with the help of logisim using the Cdm-8 processor and assembler code.

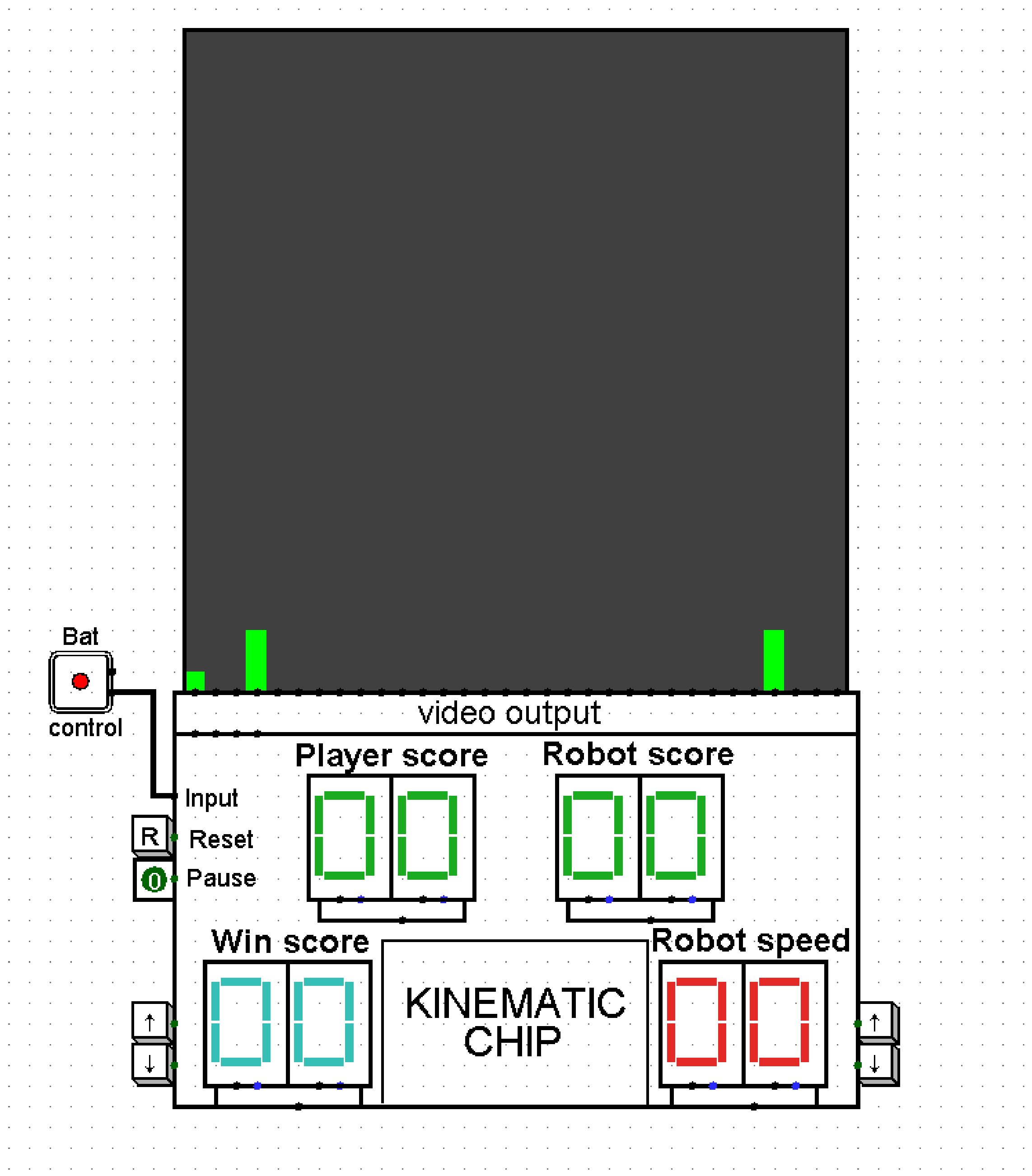
During the execution of the project, we want to gain new knowledge about building circuits, writing code and working with the processor.

Part 1 – Main

1.1 Main

"main" (Pic. 1) is the main circuit of the project. It contains a matrix, a chip for image output, a "kinematic chip" and auxiliary chips for value output.

Using the joystick, the player controls the bat, using the buttons located to the left of the "kinematic", he can set the desired number of points to win, using the buttons located to the right of the "kinematic", he can adjust the complexity of the robot, 1 - very easy, 10 - impossible. Using the "Reset" button, the player can reset all the values and start the game again. Using the "Pause" contact, the player can pause the game, while the pause is active, player can change number of points for winning and robot speed.



Pic. 1 - main

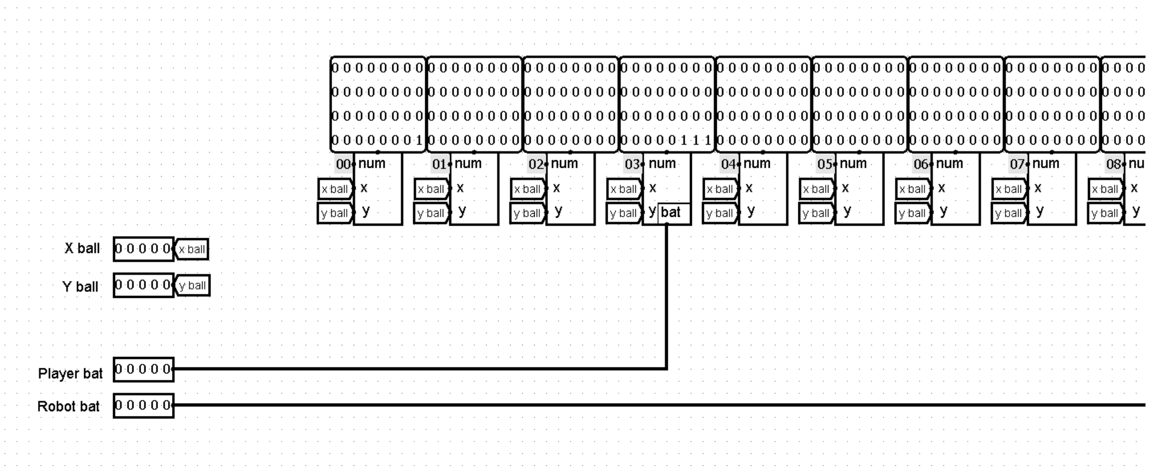
Part 2 – Video chip

2.1 full video chip

The "full video chip" (Pic. 2) circuit is responsible for the image output on the matrix.

Circuit receives 4 5-bit numbers as input: “X ball” –current position of the ball by X, “Y ball” - the current position of the ball by Y, “Player bat” - the current position of the Player bat, “Robot bat” - the current position of Robot bat.

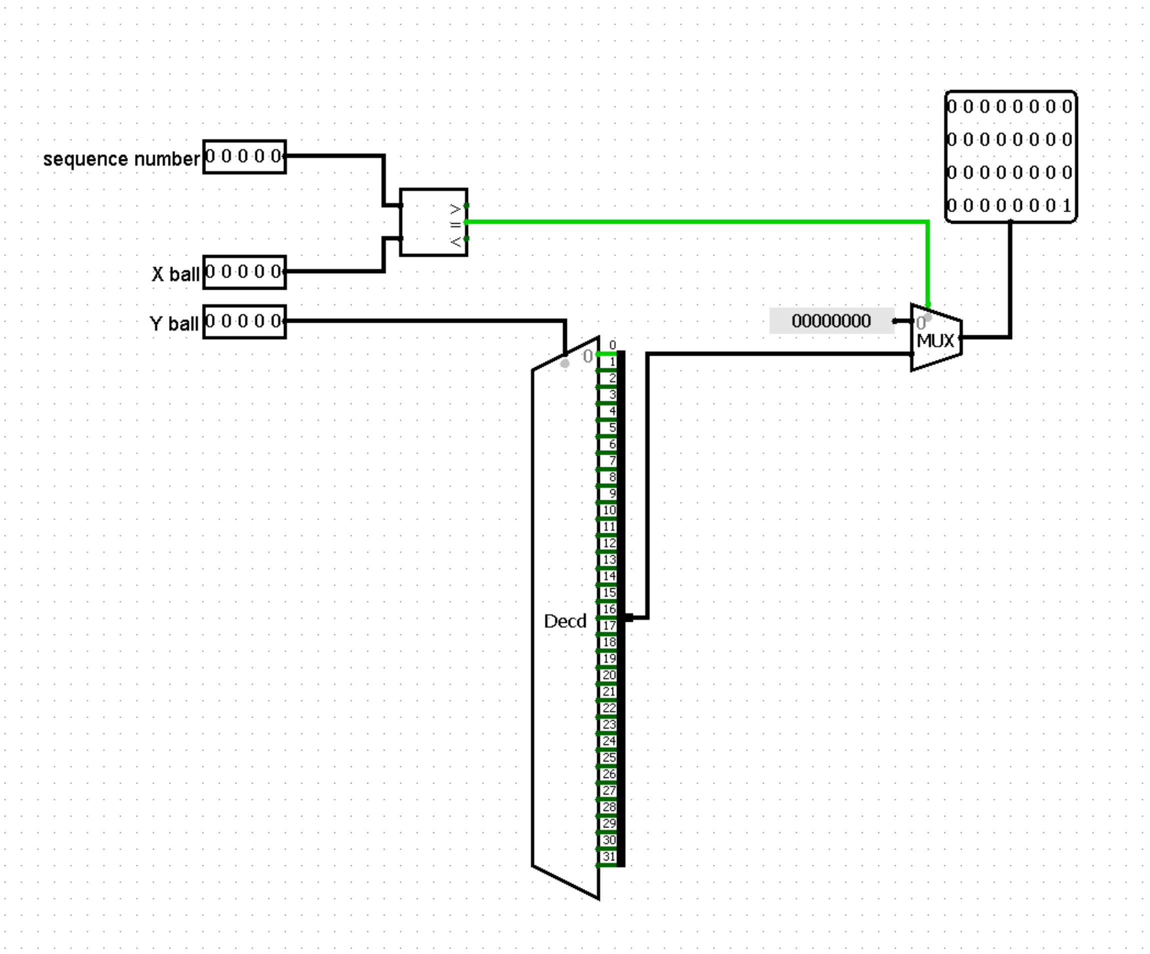
This circuit has 32 32-bit outputs, each of which, except for 3 and 28, has a "ball video" circuit connected to it. The outputs 3 and 28 are connected to “ball video with bat” circuit.



Pic. 2 Part of “full video chip”

1.2 ball video

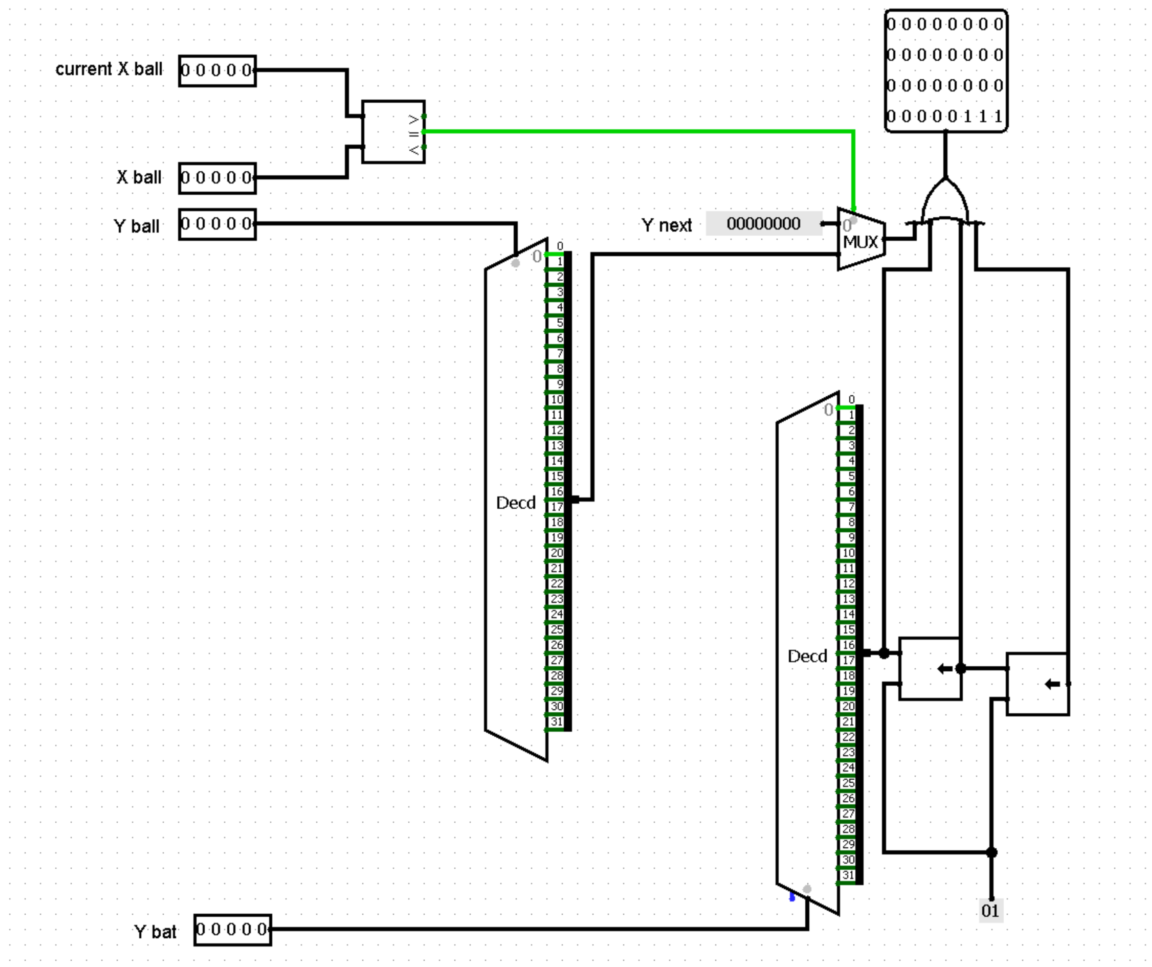
The "ball video" (Pic. 3) circuit receives 3 5-bit numbers as input: "sequence number" – it’s number of current segment, “X ball” – current position of the ball by X, “Y ball” - the current position of the ball by Y. When "sequence number" and "X ball" are equal, this means that at the moment ball is in the column under the number "sequence number", and the 5-bit “Y ball” value is converted into a 32-bit value by a decoder and sent to the output, and we see dot in current ball position. If they are not equal, this means that at the moment ball isn’t in the desired column, and 0 is sent to the output.



Pic. 3 “ball video”

2.3 ball video with bat

The “ball video with bat” (Pic. 4) works the same way as "ball video", but also, with the help of a decoder and two left bit shifts of one, and operation “or”, displays the current position of the bat. And, if the ball is in the same column as the bat, the position of the ball is also passed.



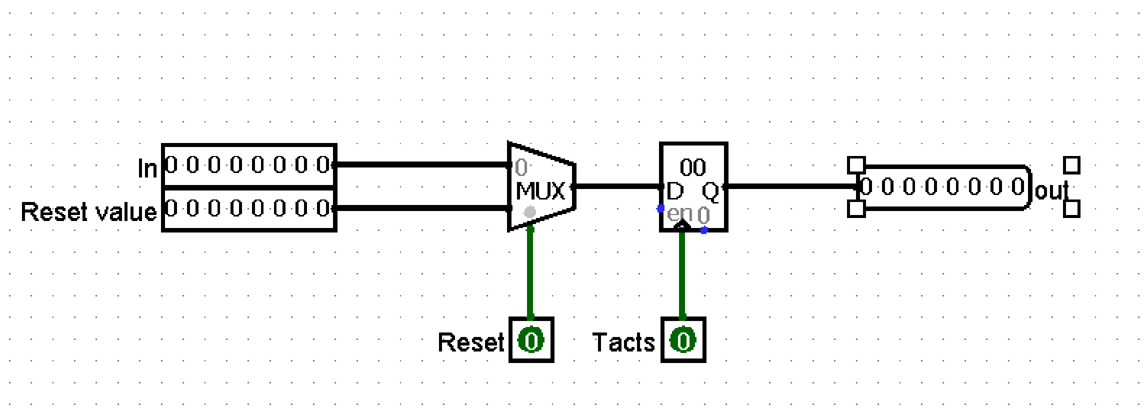
Pic. 4 ball video with bat

Part 3 – Registers

3.1 registers

The registers store the value, update it according to the clock purity applied and the value applied to “in” input, reset it to the original value applied to the “Reset value” if "reset" input is 1 and send stored value to the output "out".

The project has registers for storing 3-bit, 5-bit and 8-bit values (Pic. 5)



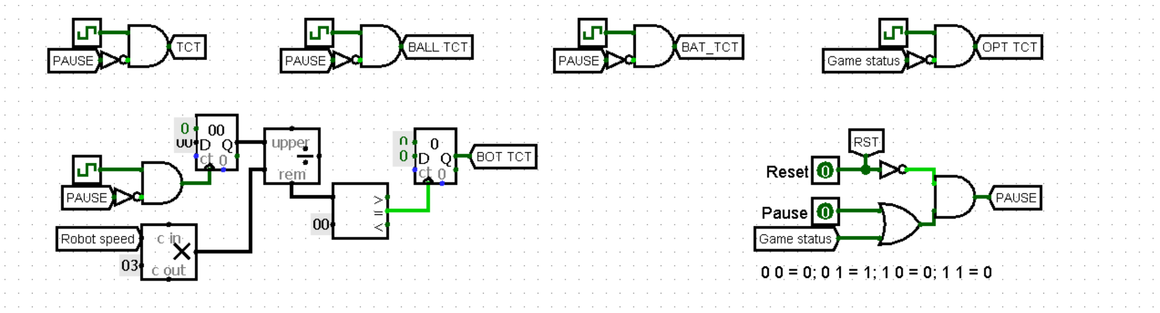
Pic 5 8-bit reg

Part 4 – Kinematic chip

The Kinematic chip has many circuits that together describe the logic of the game.

4.1 Clock generators

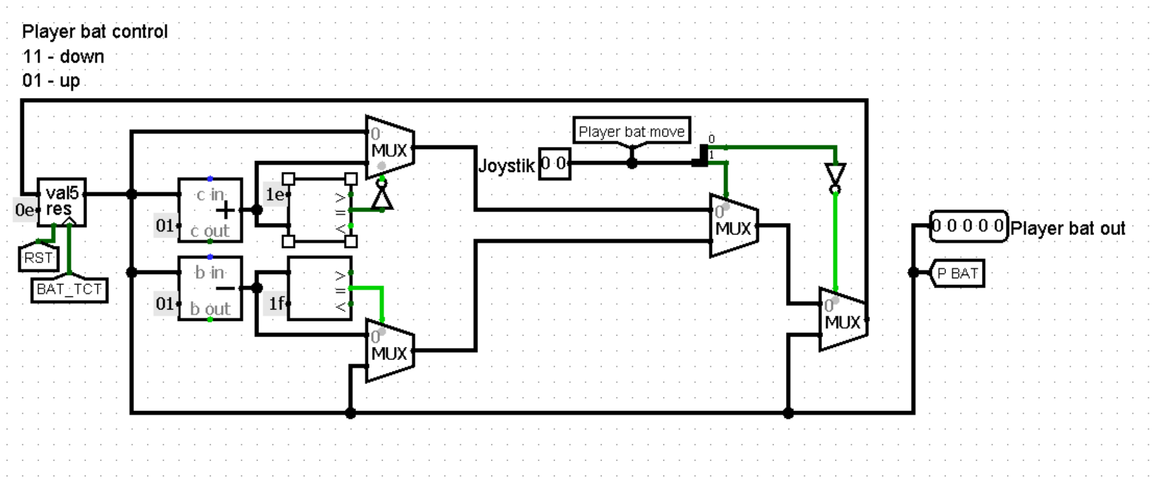
Different clock cycles are needed to adjust speed of individual chip components. With the "and" element we can specify at what point the cycles should stop refreshing. Most of them stop updating if you press pause or if the "game status" value is 1. Cycles for setting the robot's speed and "win score" only stop updating if "game status" is set to 1, so you can change the settings while the game is paused. It is also possible to update all values to their original values during a pause with the help of the circuit at the bottom right. All cycles on Pic. 6.



Pic. 6

4.2 Player bat control

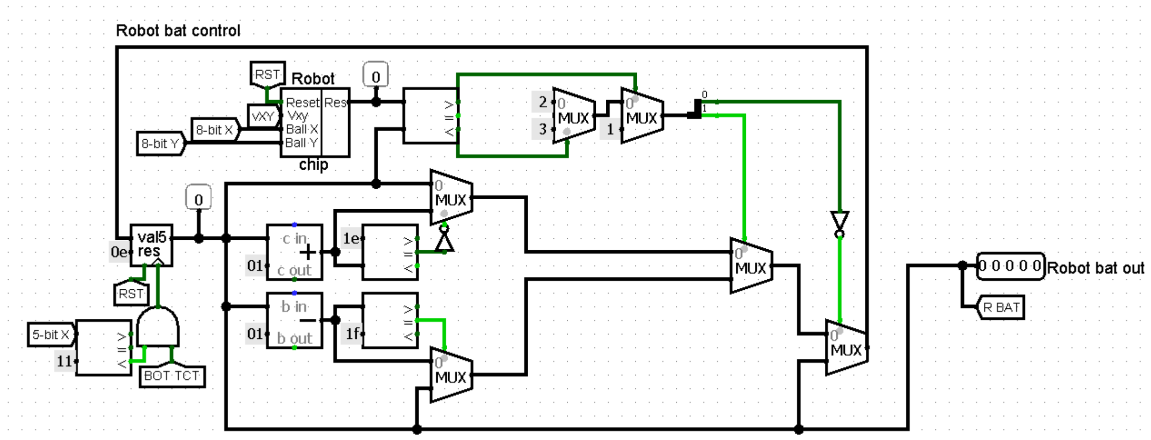
The "Player bat control" (Pic. 7) circuit is responsible for controlling the player's bat. The 5 - bit register stores the current value of the bottom pixel of the bat. The circuit calculates the next bit position, which can remain the same, increase or decrease by 1. Then, with the help of comparator and multiplexers, it checks if the calculated values are outside the screen. Then, depending on the joystick position, it sends the updated value to the "Player bat out" output and to the input of its register.



Pic. 7 - Player bat control

4.3 Robot bat control

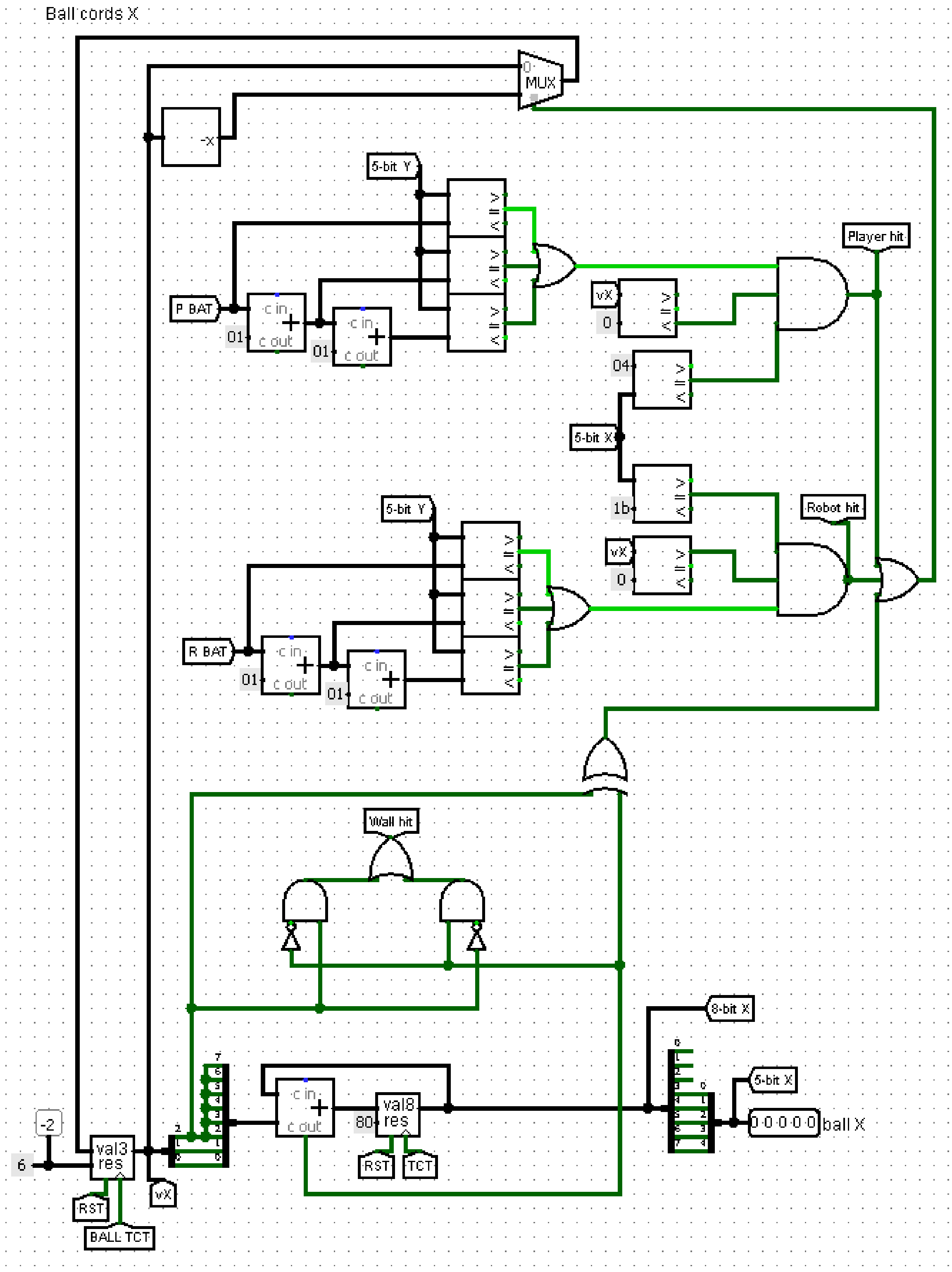
Works the same way as "Player bat control", but selects the direction of the racket depending on the value calculated by the CPU in “Robot chip”. Also restricts the movement of bat if ball crosses the middle of the field. Pic. 8



Pic. 8 – Robot bat control

4.4 Ball cords X

“Ball cords X” (Pic. 9) circuit responsible for the movement of the ball on the X, its speed, reflection from rackets and walls. X coordinate of the ball is stored in 8-bit register, its speed in 3-bit register. Each tact set for the ball, current X position of the ball increases by its speed. If ball reflect from any bats or walls, its speed becomes negative, and the tunnels transmit information about what the ball bounced off, 8-bit X position and 5-bit X position.

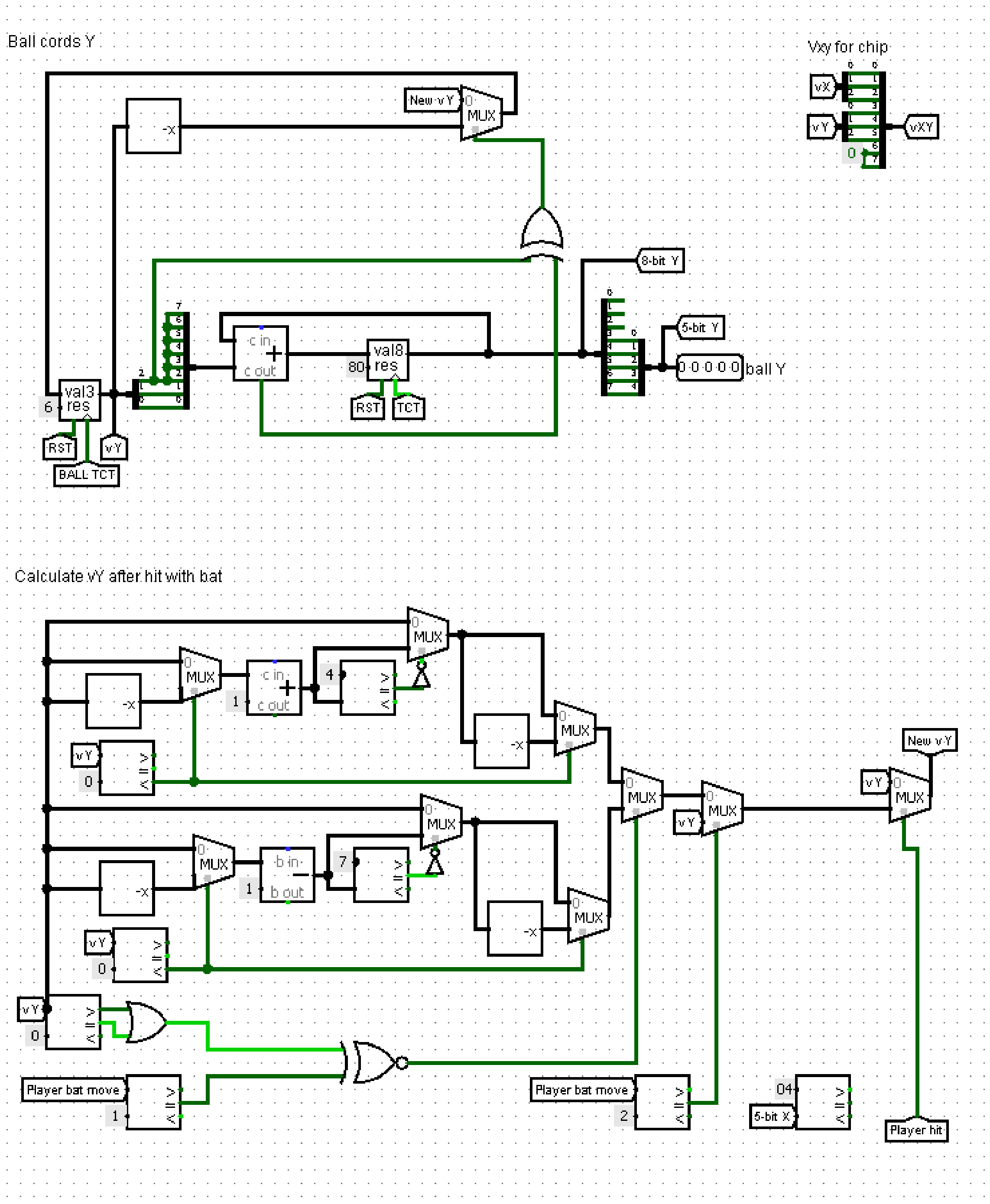


Pic. 8 – Ball cords X

4.5 Ball cords Y

“Ball cords Y" (Pic. 9) works in the same way as "Ball cords X", but instead of reflections from bats and walls, handles reflections from the top and bottom of the screen, and calculates the new speed in Y if the ball bounces off the player’s bat.

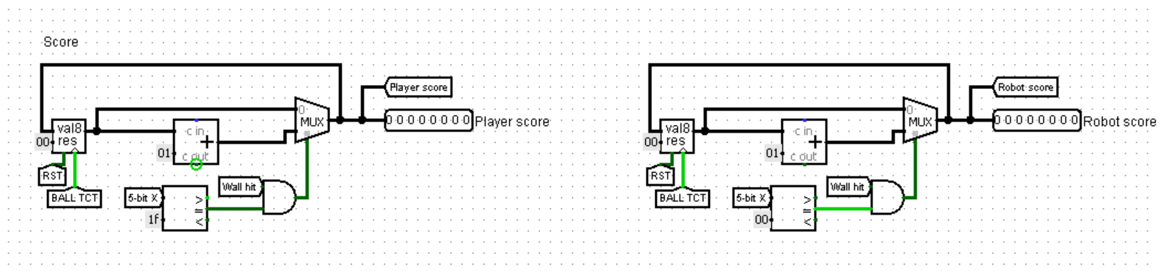
The new speed is calculated using the "XNOR" operation. If the bat moves upwards, the "Player bat move" tunnel gives a value of 1, otherwise 0. If the Y speed of the ball is greater than or equal to 0, it means it moves upwards, then 1 is send. Further, if "XNOR" gives out 1, then they are moving in the same direction and the Y speed of the ball is reduced by, if it gives 0, Y speed is increased by 1. Next comes a check to see if the racket has moved and reflected the ball, if not, the speed remains the same.



Pic. 9 – Ball cords Y and Calculate vY after hit with bat

4.6 Score

“Score” (Pic. 10) schemes store the number of points a player and a robot have. If the "wall hit" tunnel is 1 and the 5-bit X coordinate is 31, player's score is increased by 1, if the 5-bit X coordinate is 0, robot's score is increased by 1.

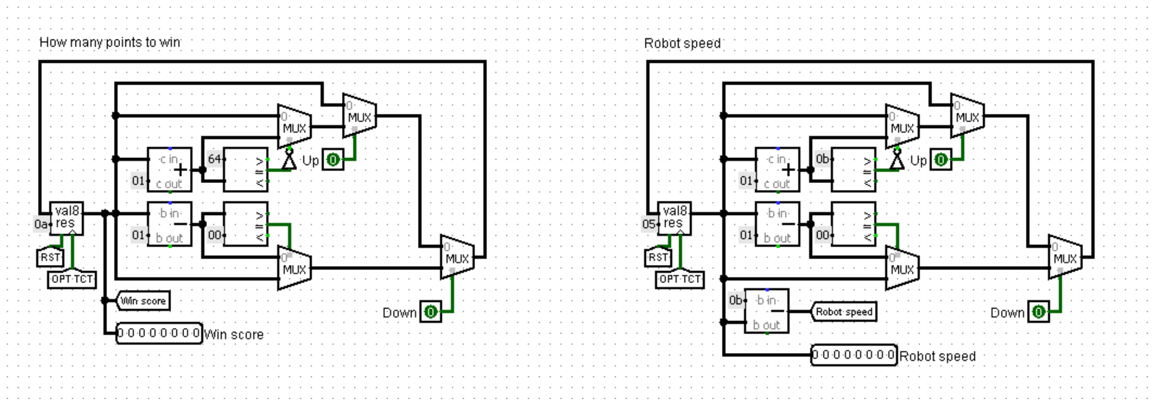


Pic. 10 – Score

4.7 How many points to win and Robot speed

These schemes (Pic. 11) store the number of points needed to win and the speed of the robot accordingly. These values can be changed by player using the buttons. The maximum value for points is 99, the maximum value for robot speed is 10 and the minimum value is 1 for both.

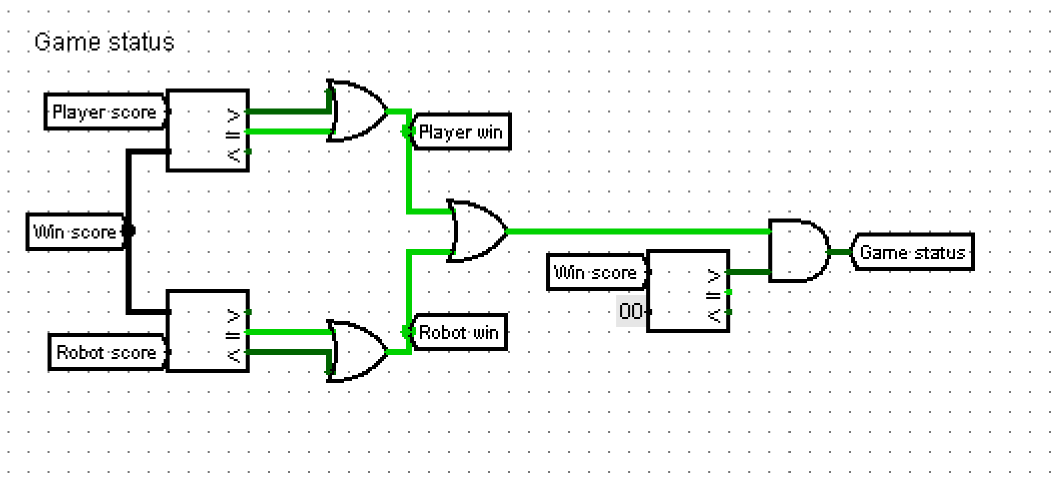
Also "How many points to win" passes the value to the "Game status" circuit. "Robot speed" passes (11 - the value set by the user) to the cycles.



Pic 11 - How many points to win and Robot speed

4.8 Game status

“Game status” (Pic. 12) Responsible for whether the game is active or not. The game is not active if the player has won or lost. This is calculated by comparing the "win score" and the points of the player and the robot.

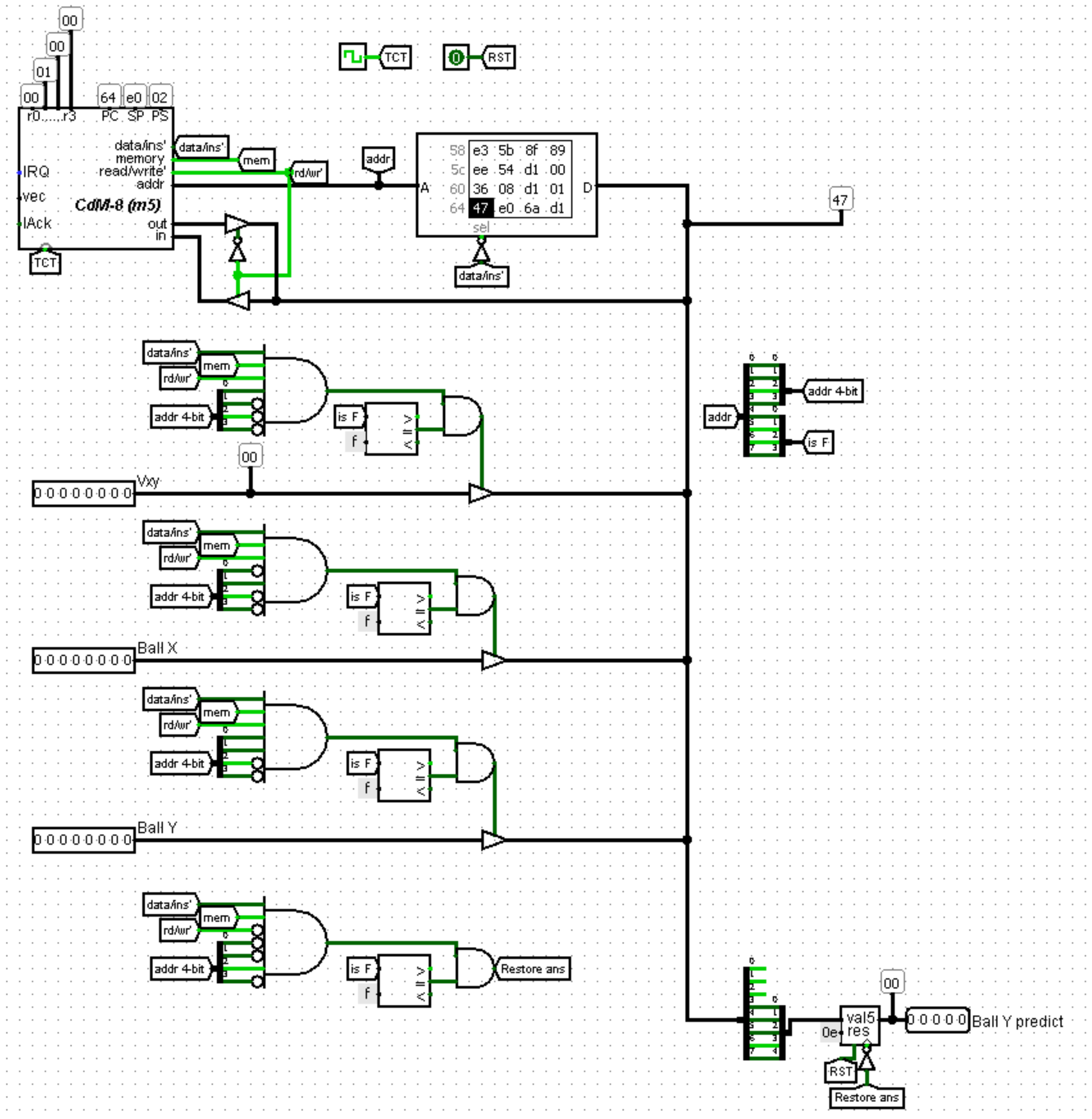


Pic. 12 – Game status

Part 5 – Robot chip

5.1 – Robot chip

“Robot chip” (Pic. 13) use CdM-8-mark5 and Von Neumann architecture. The values transmitted to the processor are stored in the cells 0xf1 - vxy, 0xf2 - ball X, 0xf3 - ball Y. The calculated value is stored in cell 0xf4. Using the "And" operation, we check which cell is executed in the processor, if it matches one of the inputs, we pass it to the processor using buffer gate, if it is equal to output, we overwrite the value in the register. The circuit itself gives the predicted 5-bit position of the ball.



Pic. 13 – Robot chip

Part 6 – CdM–8 code

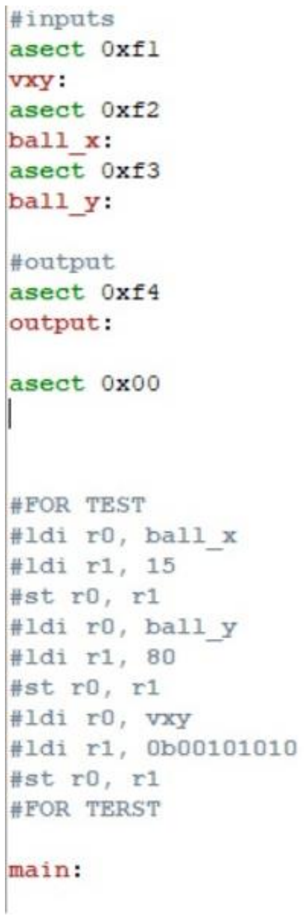
6.1 – Code

The code calculates the position of the ball when it reaches the robot's racket. This is done using the formula Y + (224 - X) / vX * vY.

In order for the calculation to be correct, you need to take into account that the ball will be reflected from the upper and lower boundaries. Code does this by counting overflows when multiplied by vY. If this count odd, then we have to subtract the result from 0.

If vY is negative, the code reflects the field vertically. Code does this by negating vY and subtracting Y from 0. In order for the position to be correct, the result is subtracted from 0.

Also code calculating predicts only if vX more than 0, this is done for optimization.



```
#inputs
asect 0xf1
vxy:
asect 0xf2
ball_x:
asect 0xf3
ball_y:

#output
asect 0xf4
output:

asect 0x00



#FOR TEST
#ldi r0, ball_x
#ldi r1, 15
#st r0, r1
#ldi r0, ball_y
#ldi r1, 80
#st r0, r1
#ldi r0, vxy
#ldi r1, 0b00101010
#st r0, r1
#FOR TERST

main:
```

```
ldi r1, vxy
ld r1, r1
ldi r0, 0b00000100
if
    and r0, r1
is z
    ldi r1, ball_x
    ld r1, r1
    ldi r0, 208
    sub r0, r1 # 208 - x

    shr r1
    ldi r0, 0b01111111
    and r0, r1 # (208 - x) / 2
    move r1, r0 # res in r0

    ldi r1, vxy
    ld r1, r1
    shr r1
    shr r1
    shr r1
    ldi r2, 0b00000111
    and r2, r1 #vY in r1

    ldi r2, 0b00000100 #check more or less that 0
    if
        and r1, r2
    is z #vY >= 0
        ldi r2, ball_y
        ld r2, r2

        ldi r3, 0 #correction
        sub r2, r3
        move r3, r2
        ldi r3, 0 #count of C
        while
            tst r1
        stays gt
            if
                add r0, r2
            is cs
                inc r3
            fi
            dec r1
        wend
        move r2, r0 #y + (208 - x) / 2 * vY (for possitive vX) IN r0
    else #vY < 0
        neg r1
        ldi r2, 0b00000111
        and r2, r1

        ldi r2, ball_y
        ld r2, r2
        ldi r3, 0 #correction
        sub r3, r2

        ldi r3, 0 #count of C
        while
            tst r1
        stays gt
            if
```

```
                    add r0, r2
                is cs
                    inc r3
                fi
                dec r1
            wend
            ldi r1, 0
            sub r1, r2
            move r2, r0 #y + (208 - x) / 2 * vY (for negative vY) IN r0
        fi

        #Carry check
        ldi r1, 1
        if
            and r1, r3
        is nz
            ldi r1, 0
            sub r1, r0
        fi
        #Carry check

        ldi r1, output
        st r1, r0 #res in output
    fi
br main

halt
end
```

CONCLUSION

As a result of the work done, we have created one of the first computer games in the world. The work was done by building circuit and writing code for the processor. During the project, we gained knowledge on how to build circuits, write assembly code and work with the processor.

REFERENCES

1. Digital Platform lectures
2. Computing platforms / A. Shafarenko and S. P. Hunt