CS/MR Digital platforms 2022/23

Co-design Group Project

"Doodle Jump"

Notes Gubin Svyatoslav, Kirill Borodin, German Ritter

Table of contents

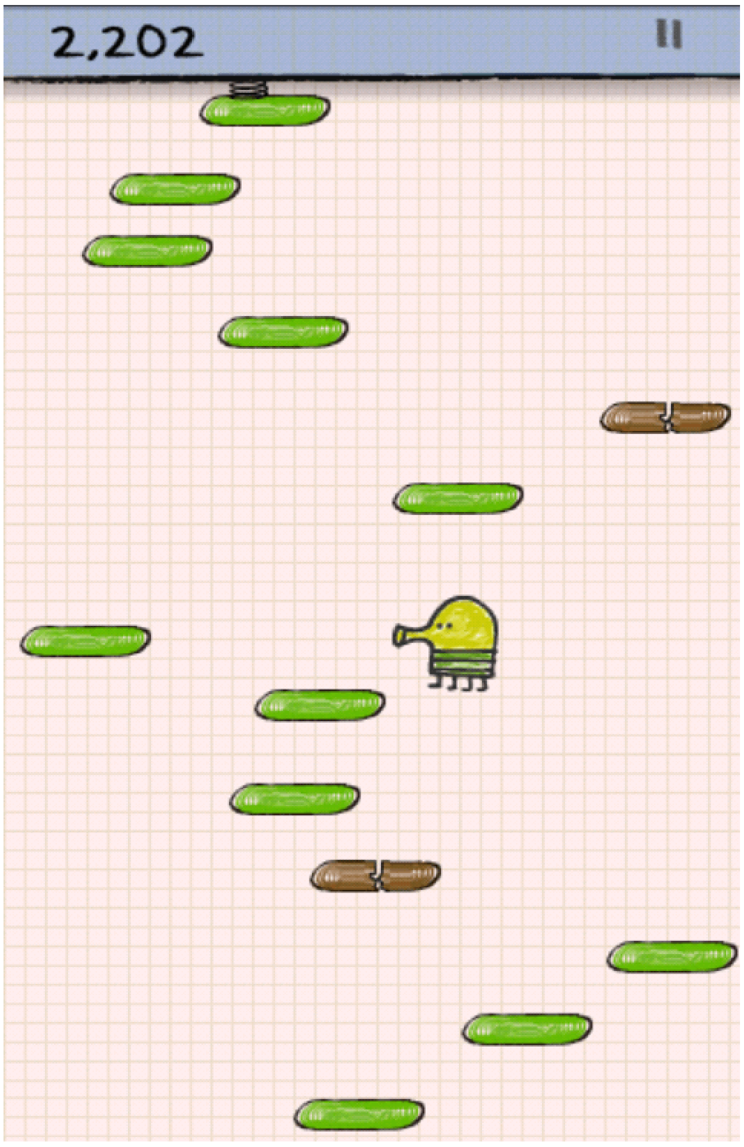
Overview

Our goal was to recreate the side-scrolling arcade platformer "Doodle Jump".
The main challenge was creating a map to be infinitely generated at random and adding a character to have one-way collision with platforms.

The hardware includes 13 chips:

1) loadField creates the map and its subsequent generation;
2) firstStageOfJump moves the character during the first stage of the jump;
3) CharSprite stores a character's sprite and changes it as you move around the map;
4) checkCollision checks the collision between the character and the platforms;
5) charPosition moves a character around the map;
6) scoreOutput calculates the score and displays it;
7) checkDeath checks if a character is dead or alive;
8) checkFall verify a character for a fall condition;
9) checkJump verify a character for a jump condition;
10) calcJumpStages Compute how high the character should jump in different jump states;
11) secondStageofJump determines the changes to the character and map during the second stage of the jump;
12) outputField outputs the current condition on the display;
13) clock slows down the tacts with a counter - it’s unclear.

Preliminaries

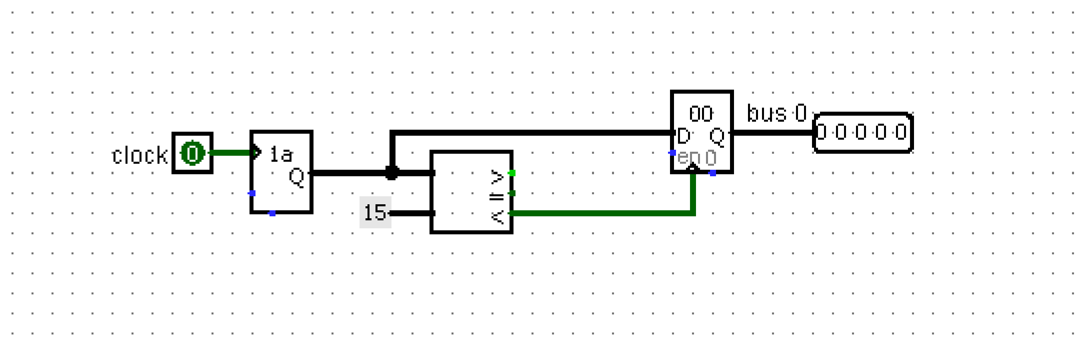


There is the character and platforms on which he is able to jump in the screenshot above.

The game was created for iOS on 6 April 2009 by Igor and Mark Puseniaki and was ported to Android and Blackberry devices on 2 March 2010.
The game ends when the character touches the bottom edge of the screen.

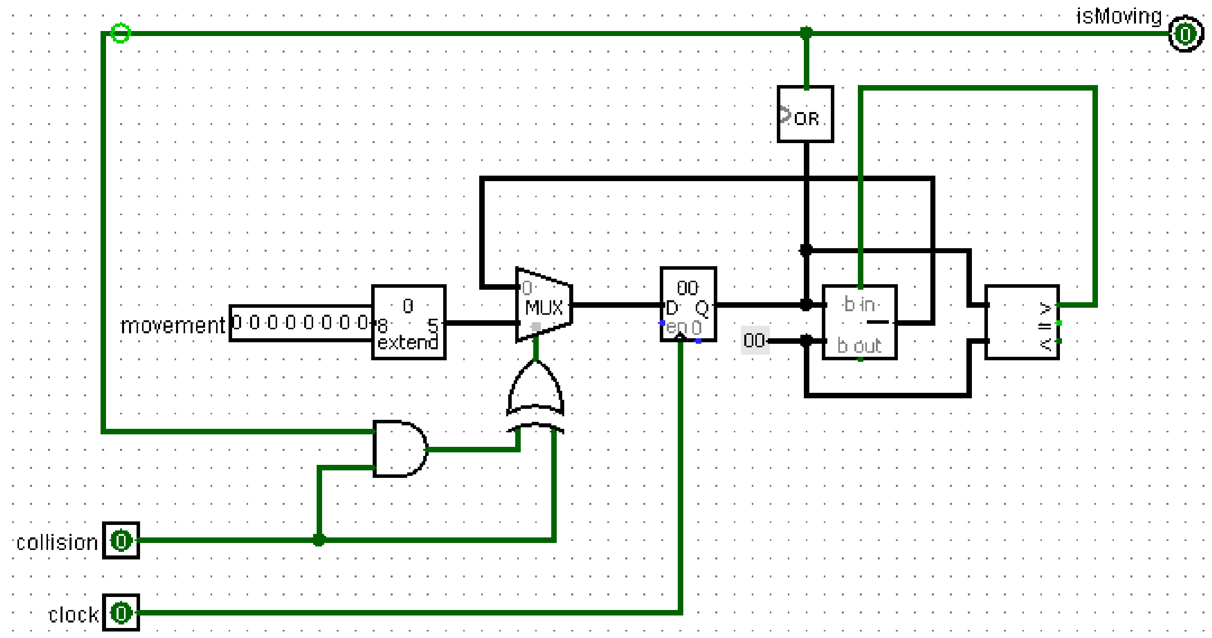
Hardware

getRandom



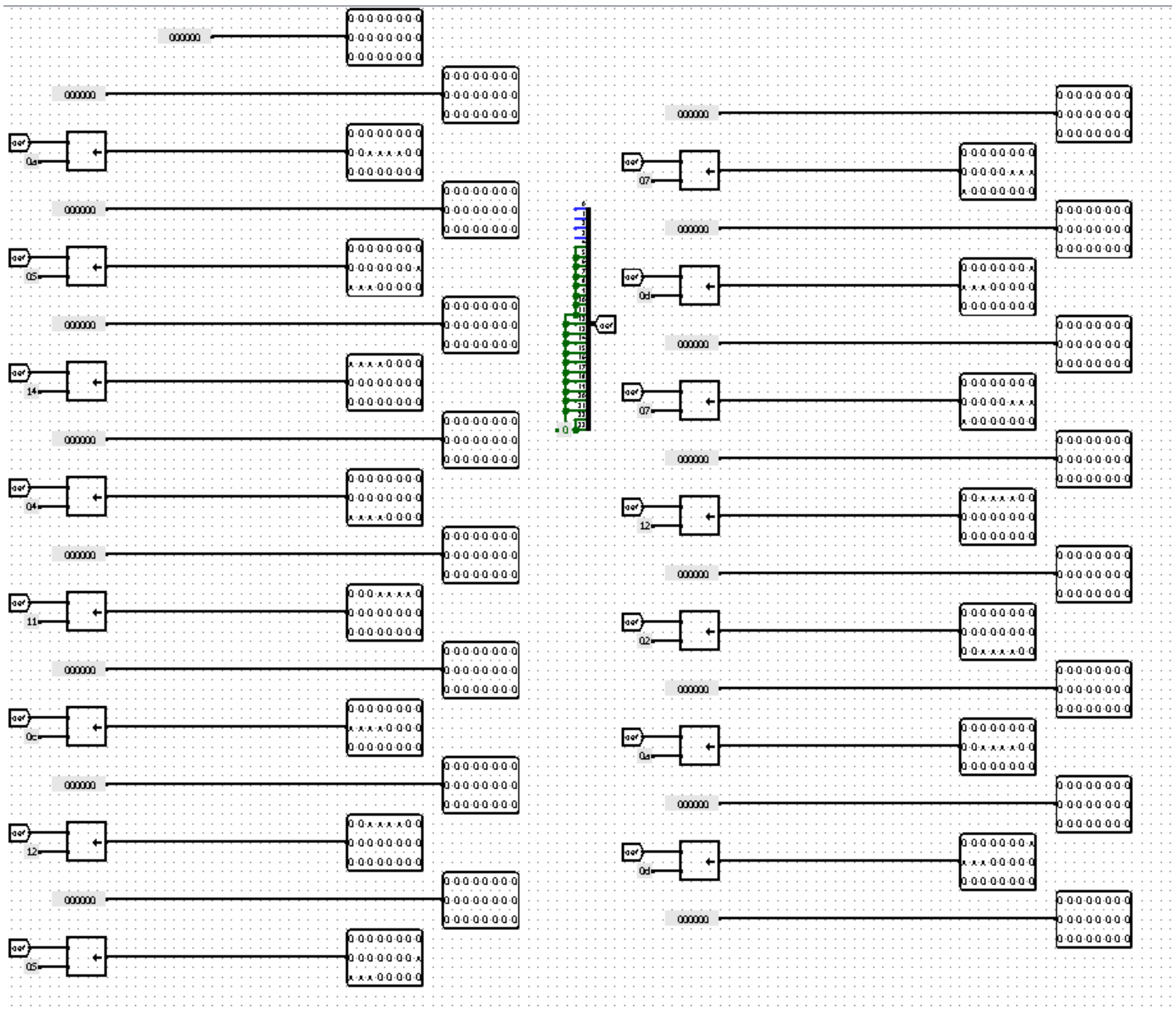
Creates a random shift within a set range to prevent the platform from generating beyond the game field using Logisim's built-in random number generator and comparator to control the maximum shift. Transmits as output the 5 bit string responsible for the platform position in the field generation chip.

isMoving



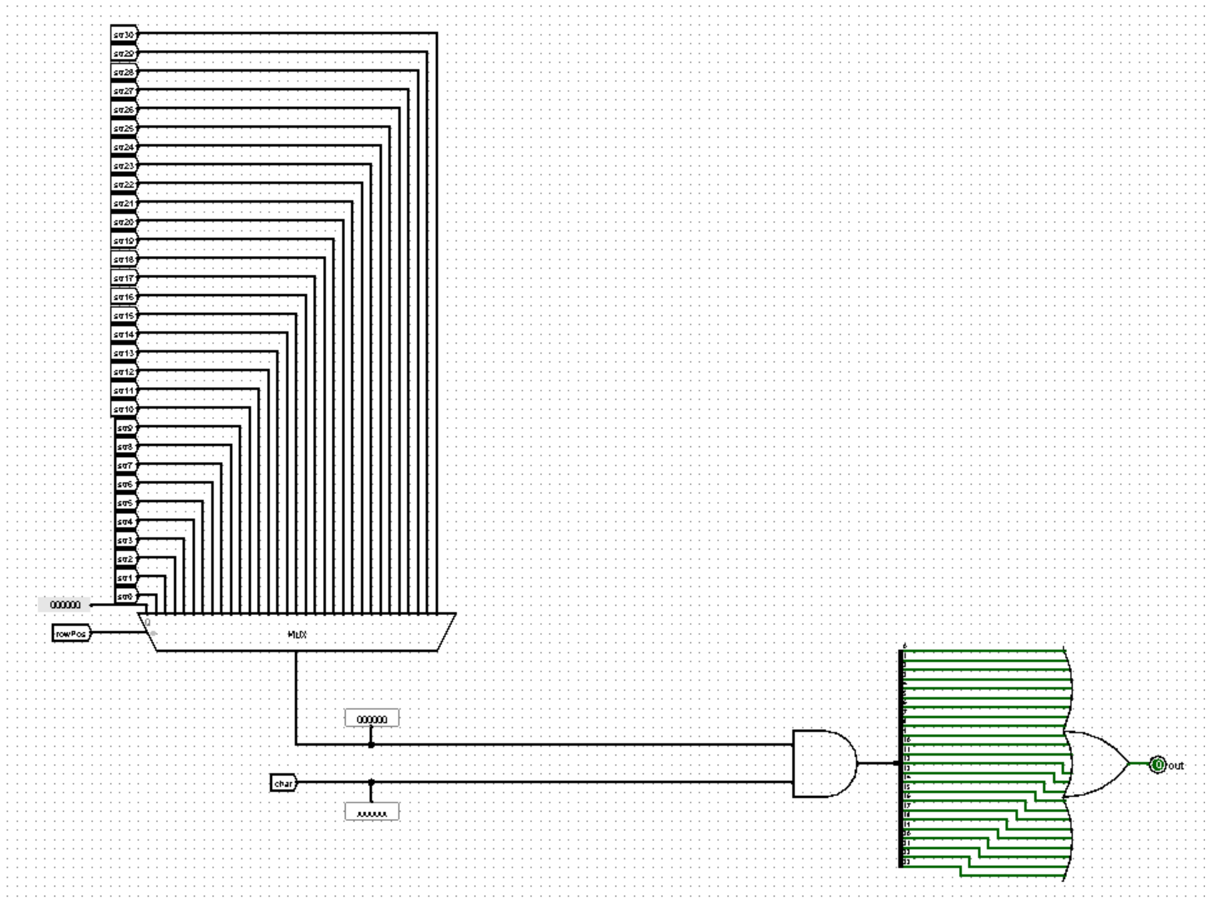
The chip gets several input values: Displacement vector as an 8-bit number, jump state, and a clock generator. The chip outputs the state of the character's movement by a certain vector.

startGameField



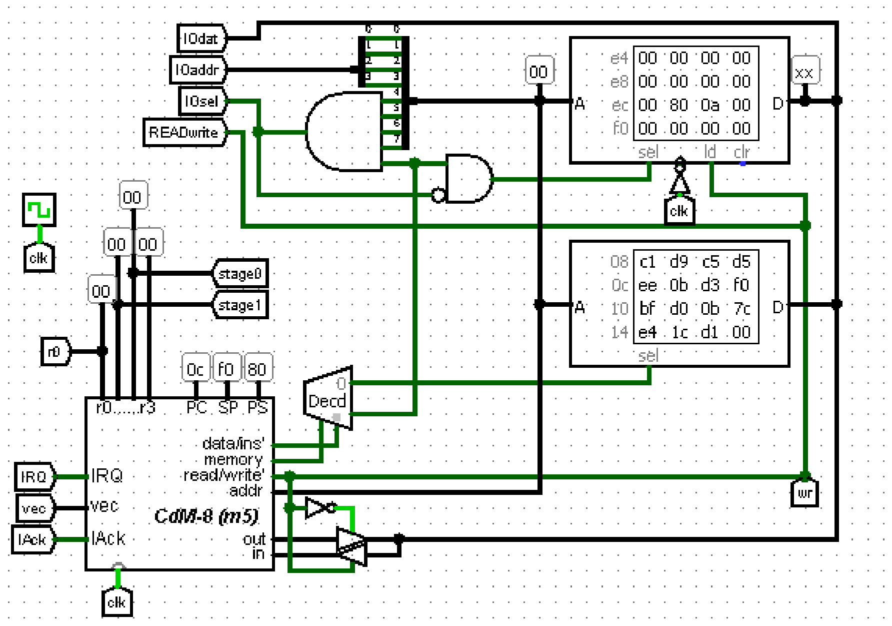
This microcircuit stores the 32 24-bit constants setting the initial location of the platforms on the map via red error which is achieved by using a preset pattern with a splitter with 4 floating bits and a shifter.

checkCollision

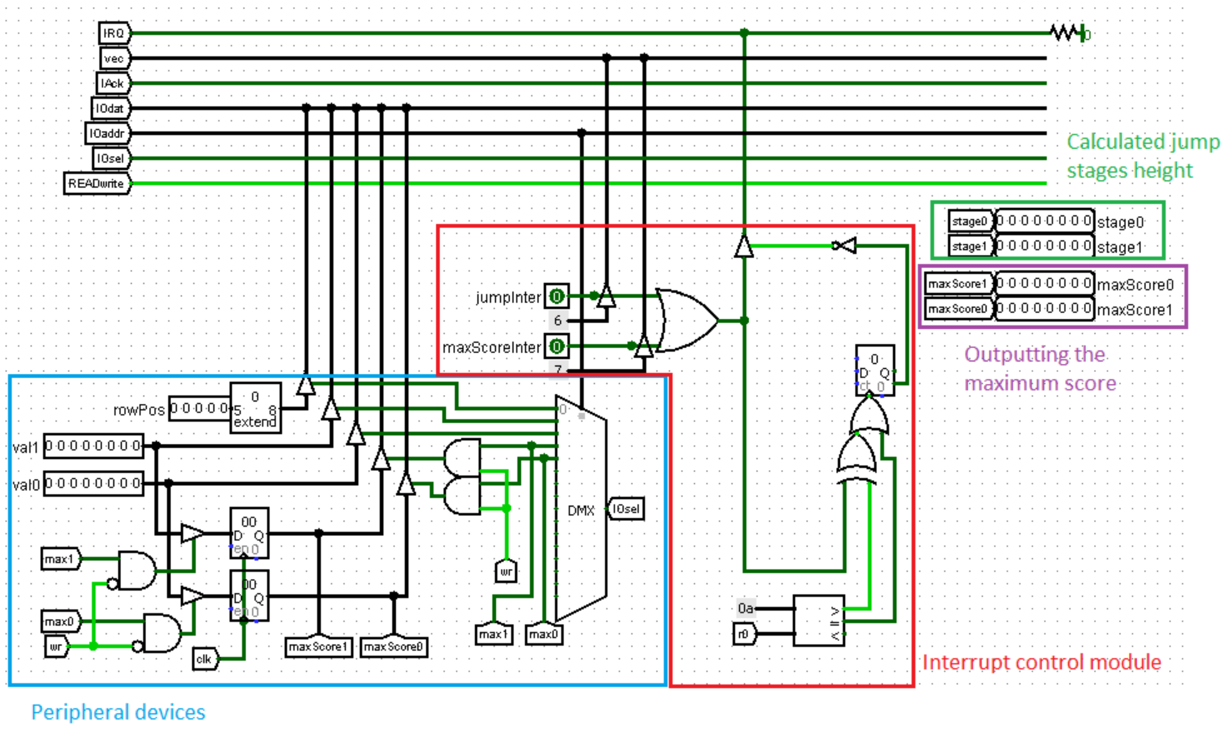


It inputs the platform’s location on the map data and checks the collision of the character with the bottom by the multiplexer to get string data, while uses the AND 32BitOr operator to determine if the platform collides with the character.

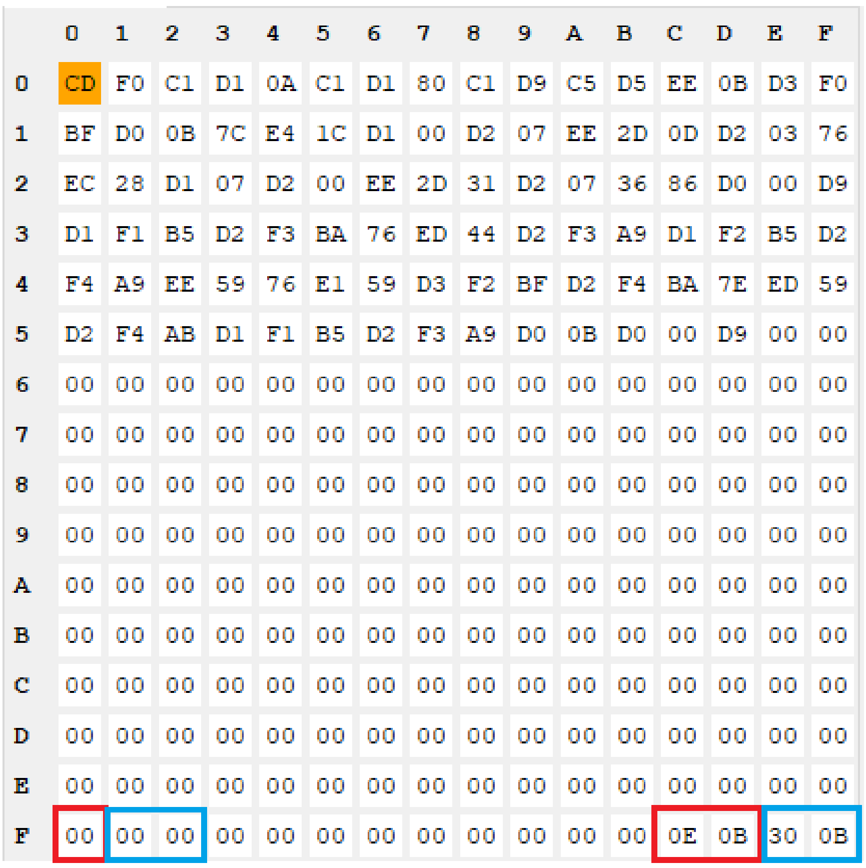
Processor



Based on the Harvard Processor Architecture model, external devices were connected to obtain character position information and store maximum score.Also was created the module to detect interrupts in the processor.


IRQ
vec
IAck
IOdat
IOaddr
IOsel
READwrite
Calculated jump stages height
stage0 0 0 0 0 0 0 0 0 stage0
stage1 0 0 0 0 0 0 0 0 stage1
maxScore1 0 0 0 0 0 0 0 0 maxScore0
maxScore0 0 0 0 0 0 0 0 0 maxScore1
Outputting the maximum score
jumpInter
maxScoreInter
rowPos
extend
val1
val0
max1
max0
wr
clk
maxScore1
maxScore0
DMX
IOsel
0a
r0
Interrupt control module
Peripheral devices

Software



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	CD	F0	C1	D1	0A	C1	D1	80	C1	D9	C5	D5	EE	0B	D3	F0
1	BF	D0	0B	7C	E4	1C	D1	00	D2	07	EE	2D	0D	D2	03	76
2	EC	28	D1	07	D2	00	EE	2D	31	D2	07	36	86	D0	00	D9
3	D1	F1	B5	D2	F3	BA	76	ED	44	D2	F3	A9	D1	F2	B5	D2
4	F4	A9	EE	59	76	E1	59	D3	F2	BF	D2	F4	BA	7E	ED	59
5	D2	F4	AB	D1	F1	B5	D2	F3	A9	D0	0B	D0	00	D9	00	00
6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
7	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
F	00	00	00	00	00	00	00	00	00	00	00	00	0E	0B	30	0B

GameLogic

The software divides the logical part of the jump, which consists of 2 stages to keep the character within a certain area of the screen to prevent him from constantly shifting to the bottom of the display. The first stage of the jump is to move the platforms without moving the character. The second stage of the jump, in its turn, moves the character without moving the platforms.

Programm

The program constantly waits for interrupts. Depending on the location of the character on the playing field (which is read from the external device located at address 0xf0), the program calculates the movement in each stage of the jump and maintains the location of the lower edge of the character no higher than the 12th line of the display storing the first jump stage value in the register r1 and the second stage in the register r2.

When a character dies, an interrupt is triggered on the processor which updates the value in the max score field if the player has reached a higher position than in previous attempts