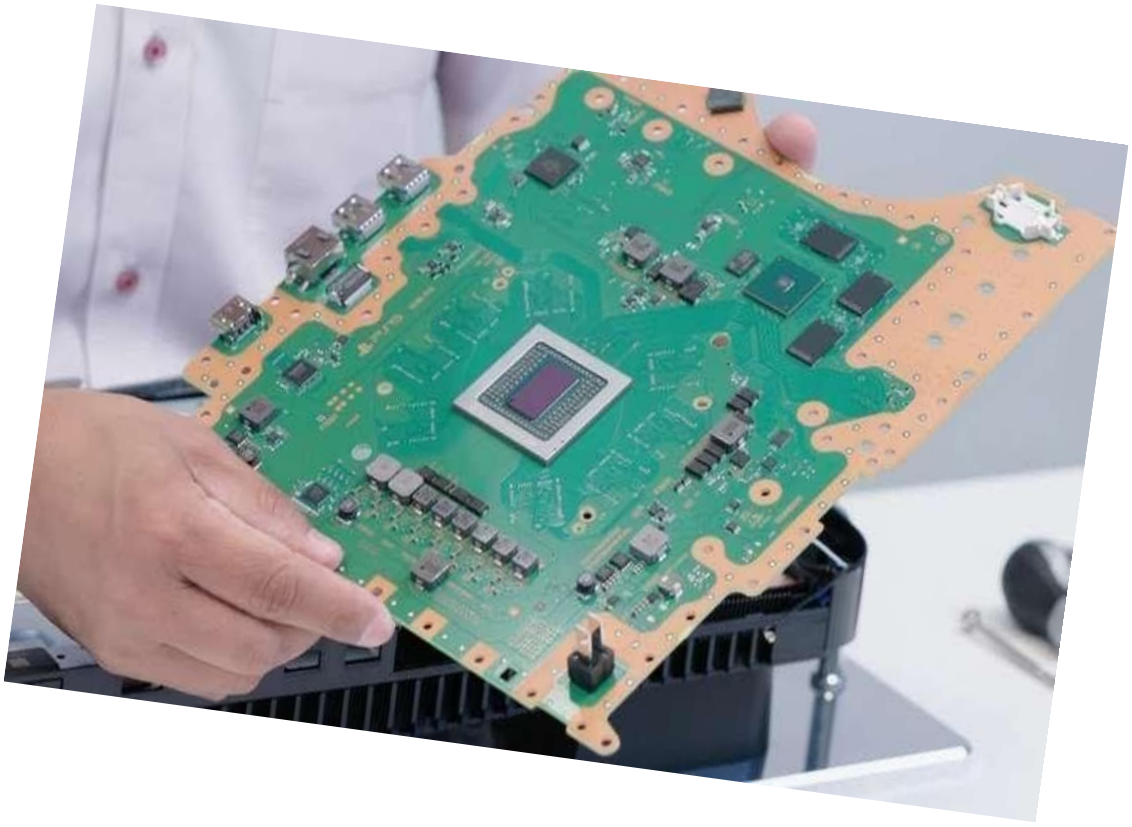


Playstation

Project 2 - 315 CA.

Dinu Ion-Irinel

Bianca Serban

Butilcă Rareș Dumitru

Teodoroiu Vlad-Mihail

Polytechnic University of Bucharest

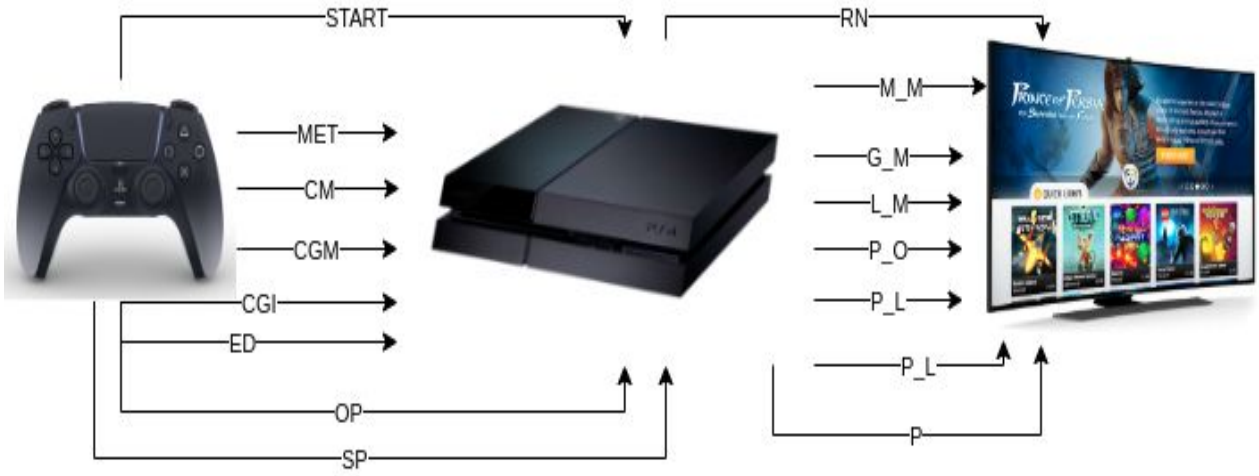
Faculty of Automation and Computers

Content

THEME OF THE PROJECT

The theme of the project is to make a PLAYSTATION. The console has two main modes of operation: a music mode and a game mode. Through the controller the user can choose his music module, and then he can start and listen to any song he wants. If he chooses to play, the user must decide whether to play online or locally via DVDs. It also has access to options such as switching the device to pause mode or turning off the power.

BLOCK SCHEME



NOMENCLATURE OF INPUT / OUTPUT VARIABLES

Entries:

Notation	Explanation Notation
START	A successful button must be pressed to turn on the appliance. 0 = no, 1 = yes
MET METHOD	Any user must choose an activity they want to perform. For 0 you access music_mode and for 1 you access game_mode.
CM CHOOSE_MUSIC	The user must choose a song. For 1 the music starts, and for 0 it remains in the current state.
CGM CHOSE_GAME_MODE	After choosing to play, the user must choose the way he wants to play. For 0 choose to play online with friends, and for 1 choose local mode.
CGO CHOOSE_GAME_ONLINE	To start the online game the user must choose a type of game. 1 = game start, and for 0 it remains in the current state.
ED ENTER_DVD	Local games can only be accessed via DVDs. 1 = insert dvd with game, 0 = stay in the current state.
OP OPTION_POWER	Device operation options. 0 = pause_mode and 1 = switch off the device
SP STOP_PAUSE	For 1 the device exits the pause mode, and for 0 it remains in the current state.
N NULL	Variable set to 0 (grounded) that was used as an auxiliary variable where a decision with a zero constant was needed.

Exits:

Notation	Explanation Notation
R_N RUNNIG	Console turned on. Wait! You have to choose an activity.
M_M MUSIC_MODE	You have chosen to listen to music. Now you can choose a song!
G_M GAME_MODE	You have chosen to play! Now you can choose a game mode!
L_M LISTEN_MUSIC	The song you chose to listen to has started.
P_O PLAY_ONLINE	You have selected the option to play online, now you have to choose a type of game.
P_L PLAY_LOCAL	You have chosen the option to play locally, so you need to insert a DVD with the desired game.
R_G RUNNIG_GAME	The game you chose to play has started.
P PAUSE	You have selected the console pause mode!

Available parts:

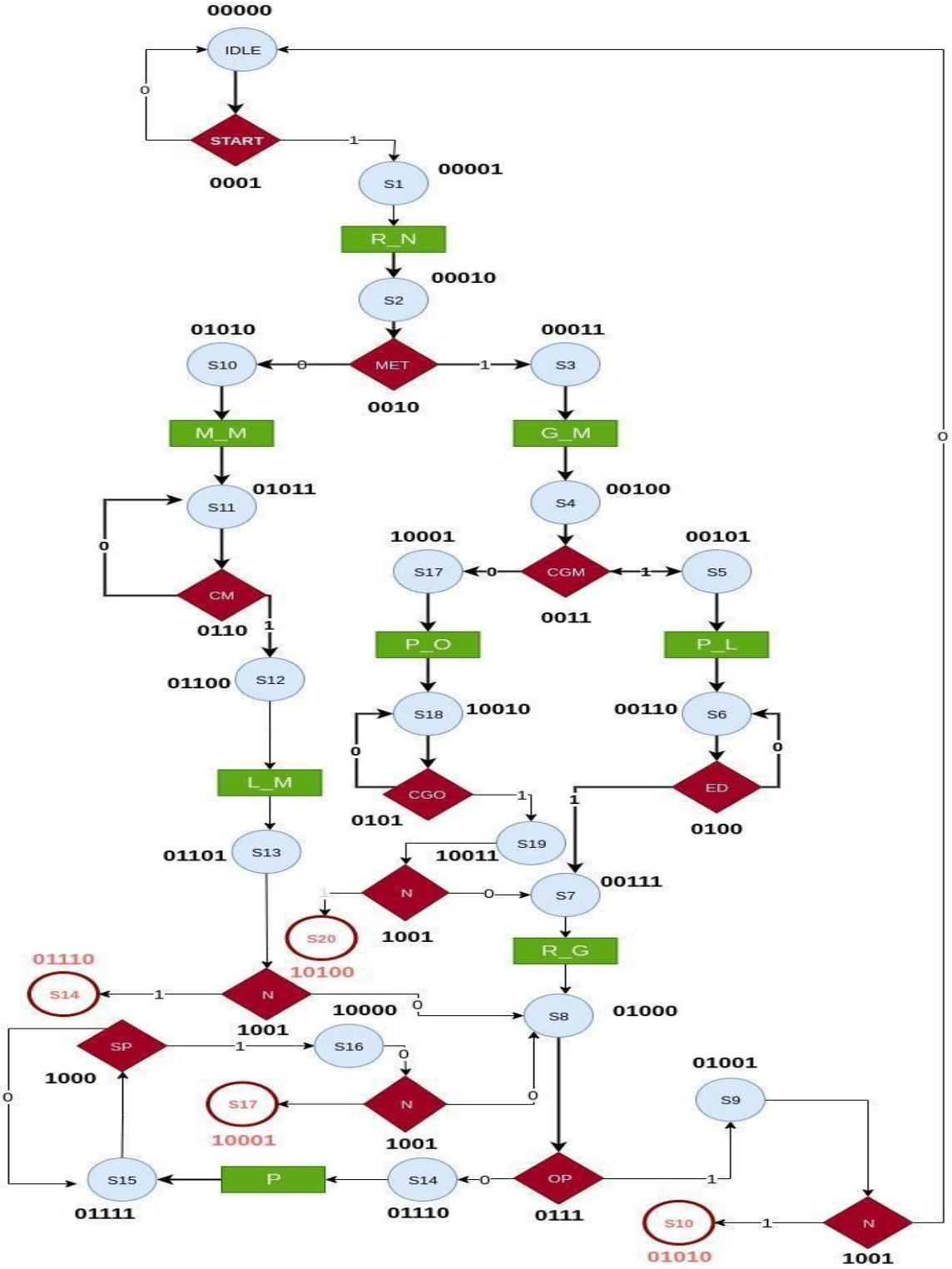
- 3 memory circuits (64 * 4 CAT 22C10)
- two counted on 4 bits (SN74193)
- a multiplexer (16: 1 54150).
- 3 AND gate faces
- 1 inverter chip

EXPLANATION OF THE FUNCTIONALITY OF THE DEVICE

The PLAYSTATION console starts from the state **IDLE** in which it remains as long as the power button is not pressed (variable **START** remains 0) .When **START** becomes 1 console starts, goes in the state **S1** , and then the user is expected to choose a boot mode. If you choose the music mode (**MET = 0**) it goes in the state **S10** where the user is expected to choose a song. Then if he successfully chooses a song, it starts and reaches the state **R_A (RUNNING_ACTIVITY)** . Also if the user chooses to play at startup, (**MET = 1**), the console reaches an intermediate state **S3** in which the user must choose whether to play online or locally. If you choose to play online (**CGM = 0**), the device reaches the state **S17** intended for online play, then in the state **S18** where the user will have to decide on a game to start. If you choose to play locally (**CGM = 1)** , the console reaches the state **S5,** then **S6** and will wait for a DVD with a game to be inserted. After one of these options has been chosen, the game starts and the console reaches the same state **R_A** .

While the user is listening to music or playing (the console is in **R_A**) it has access to several options. If he wants to rest, the user can choose the pause mode, and the default console enters the pause_mode (**OP = 0**), in which it remains as long as the user does not return and deactivates the option (**SP = 1**). If he wants to turn off the console the user can turn off the power (**OP = 1**) and the console returns to its original state **IDLE.**

ORGANIZATIONAL CHART



Calculation of the length of the microinstruction

Console design involves:

- 20 starts: $2^5 > 20 > 2^4 - 1 \Rightarrow$ 5 bits required
- 8 input variables: $2^4 > 8 > 2^3 - 1 \Rightarrow$ 4 bits required (n_ci = 4)
- 8 output variables: 8 bits required (n_out = 8)

1) L Calculi calculation if we do not take into account the chosen memory:

Light = max (Light Type 1, Light Type

0) Light Type 1 = 1 + n_out = 1 + 8 = 9;

L Tipi Type 0 = 1 + n_ci + n_adr = 1 + 4 + 5 = 10

Lμi = max (9, 10) = 10

2) **Calculation Lμi taking into account the type of memory used:**

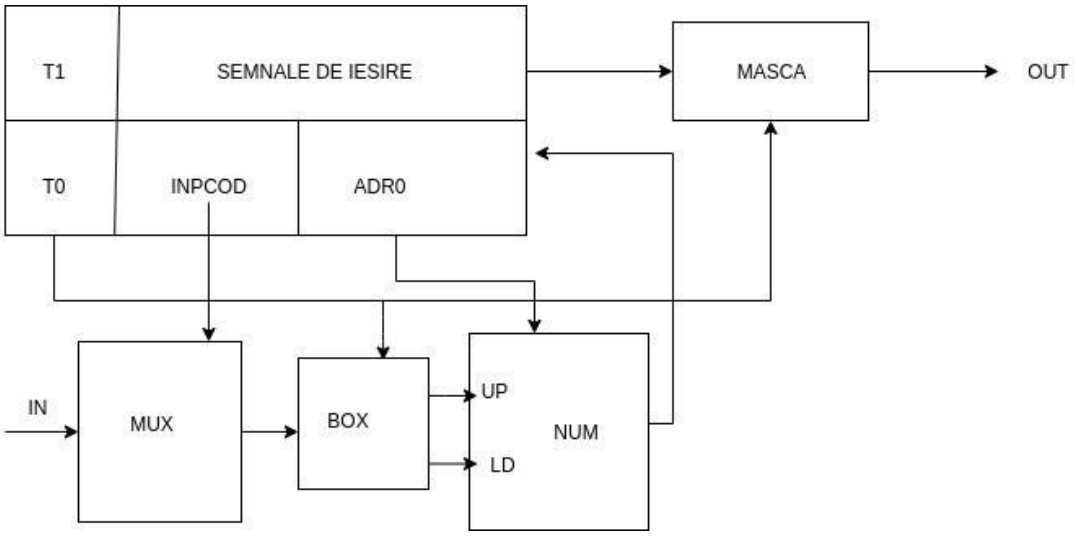
I chose 3 memory chips 64 * 4 CAT (CAT 22C10) Lμi

= 3 * 4 = 12

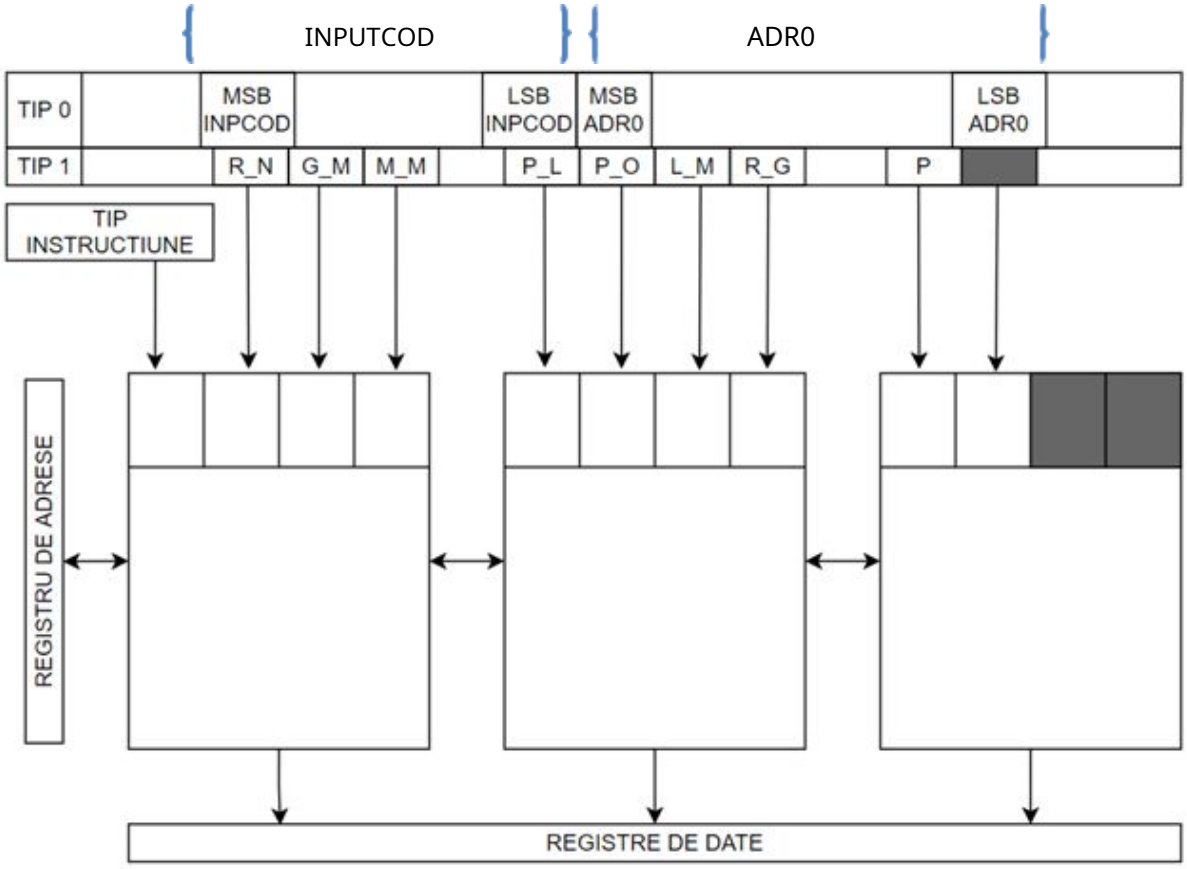
Filling in the contents of the firmware memory

	T	R_N	G_M	M_M	P_L	P_O	L_M	R_G	P	
		INPCOD				ADR 0				
00000	0	0	0	0	1	0	0	0	0	0
00001	1	1	0	0	0	0	0	0	0	*
00010	0	0	0	1	0	0	1	0	1	0
00011	1	0	1	0	0	0	0	0	0	*
00100	0	0	0	1	1	1	0	0	0	1
00101	1	0	0	0	1	0	0	0	0	*
00110	0	0	1	0	0	0	0	1	1	0
00111	1	0	0	0	0	0	0	1	0	*
01000	0	0	1	1	1	0	1	1	1	0
01001	0	1	0	0	1	0	0	0	0	0
01010	1	0	0	1	0	0	0	0	0	*
01011	0	0	1	1	0	0	1	0	1	1
01100	1	0	0	0	0	0	1	0	0	*
01101	0	1	0	0	1	0	1	0	0	0
01110	1	0	0	0	0	0	0	0	1	*
01111	0	1	0	0	0	0	1	1	1	1
10000	0	1	0	0	1	0	1	0	0	0
10001	1	0	0	0	0	1	0	0	0	*
10010	0	0	1	0	1	1	0	0	1	0
10011	0	1	0	0	1	0	0	1	1	1
10100	*	*	*	*	*	*	*	*	*	*

Diagram of the microprogrammed control unit

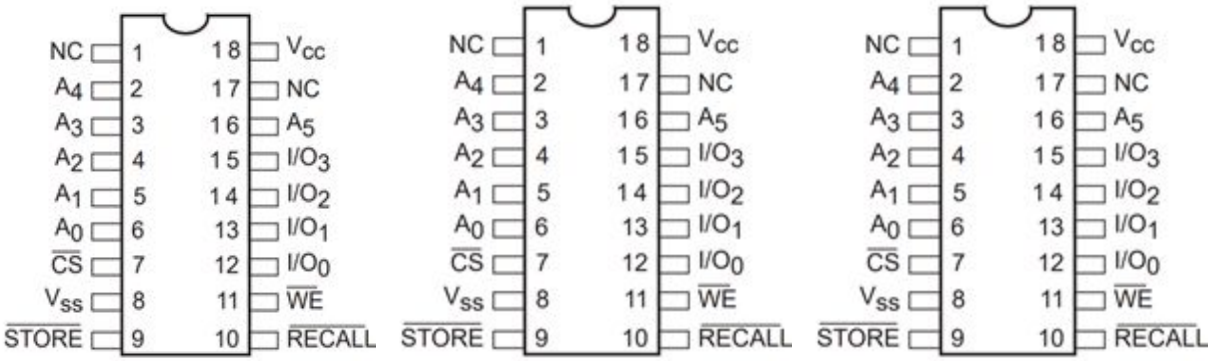


Memory faces - theoretical form



Memory faces - physical form

- memory 64x4 CAT22C10-

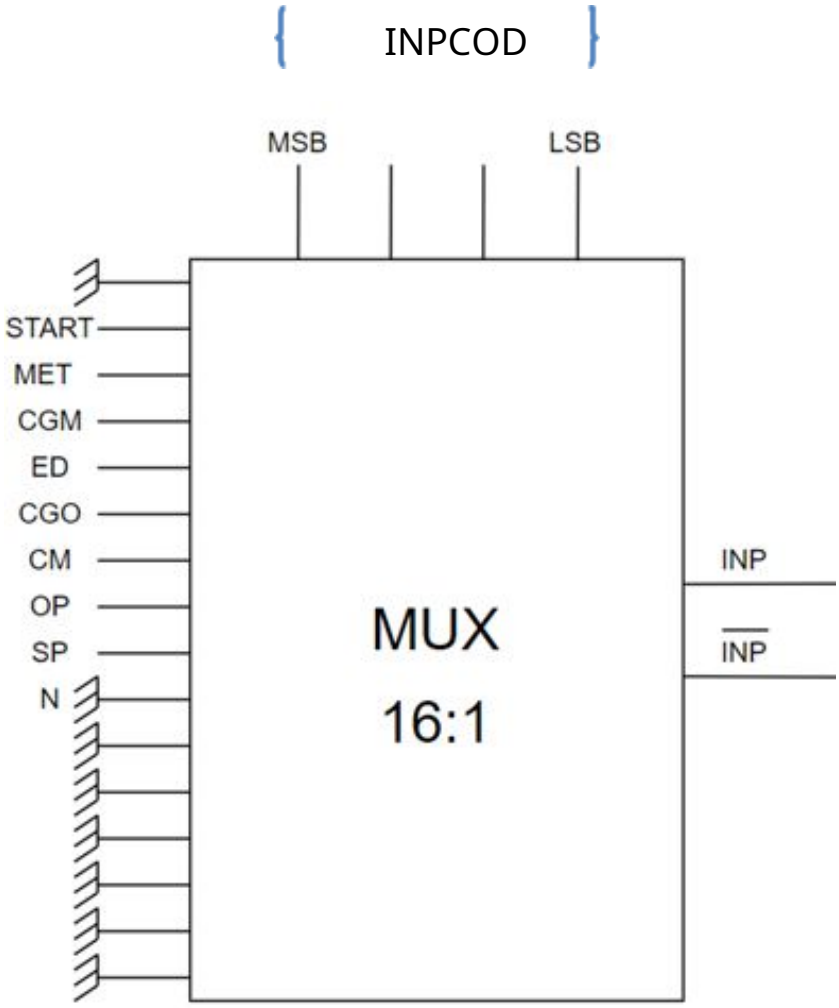


Mode	Input				I/O
	CS	WE	RECALL	STORE	
RAM Read	L	H	H	H	Output Data

- A0 - A5 -> addressing pins (MSB - set to 0, and the rest will be taken from the ADR0 code)
- I / O (0-3) -> input / output pins (are common and the output mode is selected by! CS = 0,! WE = 1,! RECALL = 1,! STORE = 1)

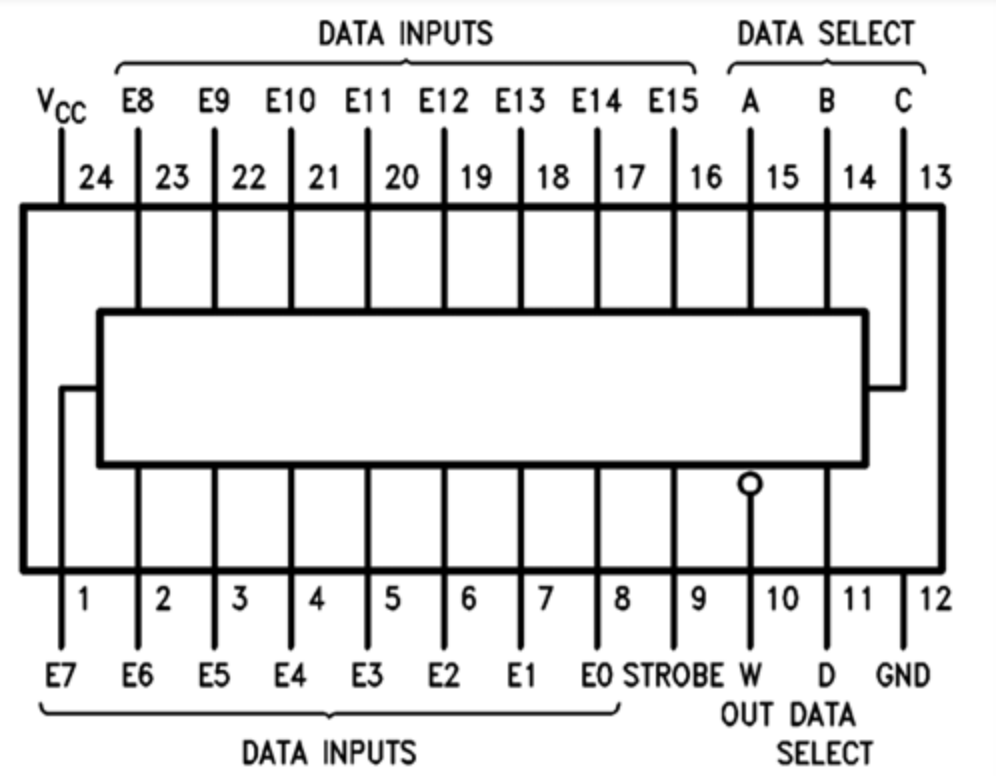
Multiplexer - theoretical form

9 inputs - 1 set to 0 (N) -> one selected input (INP)



Multiplexer - physical form

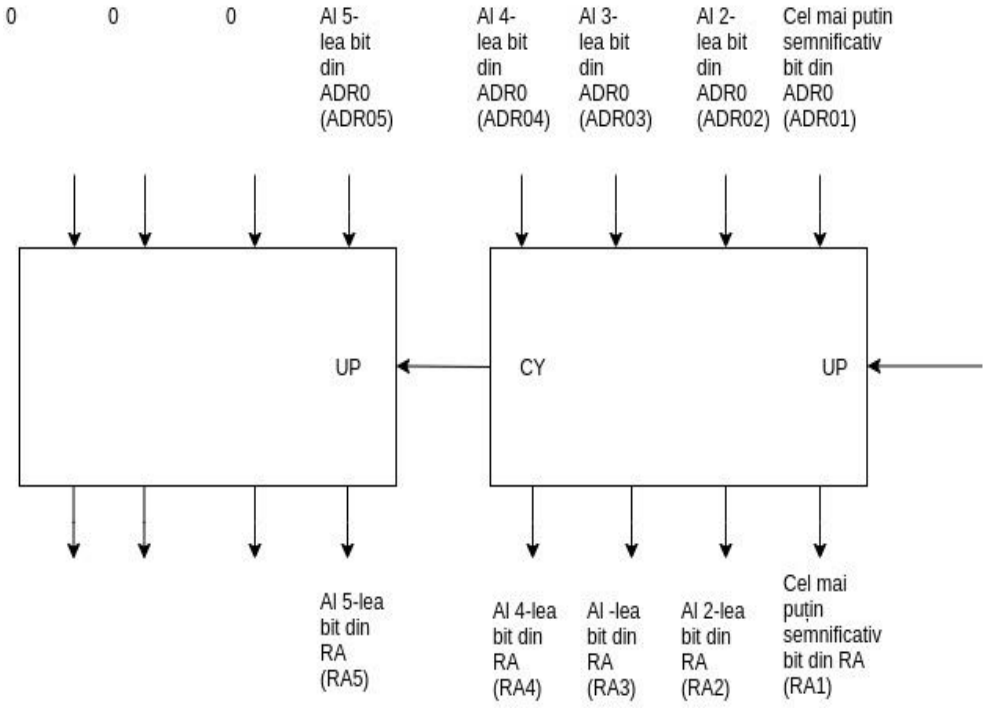
- multiplexer 54150-



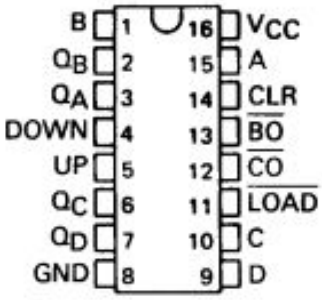
- E0-E16 -> pins for inputs, we will use only 9 of which one set to 0 => there will be 8 pins connected to ground and 8 connected to the memory register
- A, B, C, D -> pins for selection variables, we will use them all
- W OUT -> pin for negative output
- STROBE = 0 -> enable (in the circuit diagram I used a similar multiplexer where G = STROBE)

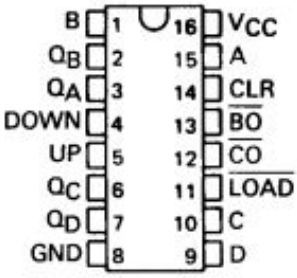
Address book - theoretical form

- RA-> 6-bit (5 variable bits and one set to 0)
- 2 bits of the counter are redundant
- We will use 2 counters on 4 bits



Address book - physical form





- D, C, B, A - input pins (if! LOAD = 0) D - represents the
- pin associated with the most significant bit
- We will use pins D, C, B, A in the first counter for the first 4 bits of the addresses, then pin A in the second for bit 5 of the addresses. The rest of the pins will be tied to 0.

- QD, QC, QB, QA from the first counter will be used for the first 4 bits of the RA, and QA from the second counter will be used for bit 5 of the RA. The rest of the pins will be left in the air.

- UP - clock signal pin for ascending counting
- ! CO - the pin that will send the clock signal to the second counter. The DOWN pin
- will be set to 1 (VCC) because, CLK =! DOWN +! UP and when DOWN = 1 -> CLK =! UP (CLK depends only on the UP input)
- CLR is set to 0 (MASS) because when it has the value 1 the values in the counter are reset

BOX site

- UP from the first counter receives the clock signal only when no LOAD is done, ie when it moves to the next state (no jump). So UP receives clock signal when type = 1 or when (type = 0 & normal MUX output is 1). Otherwise it receives the value 0 constant.

  ⇨ UP = CLK * (TYPE +! W)

- ! LOAD being on negative logic, the LOAD itself is done when the pin receives the signal 0. Given that the LOAD is done in cases where UP does not receive a clock signal, it means that! LOAD = 0 when UP = 0 - constant (does not depend on the clock).

  ⇨ ! LOAD = TIP +! W

! Remember that W is the negative output of the MUX

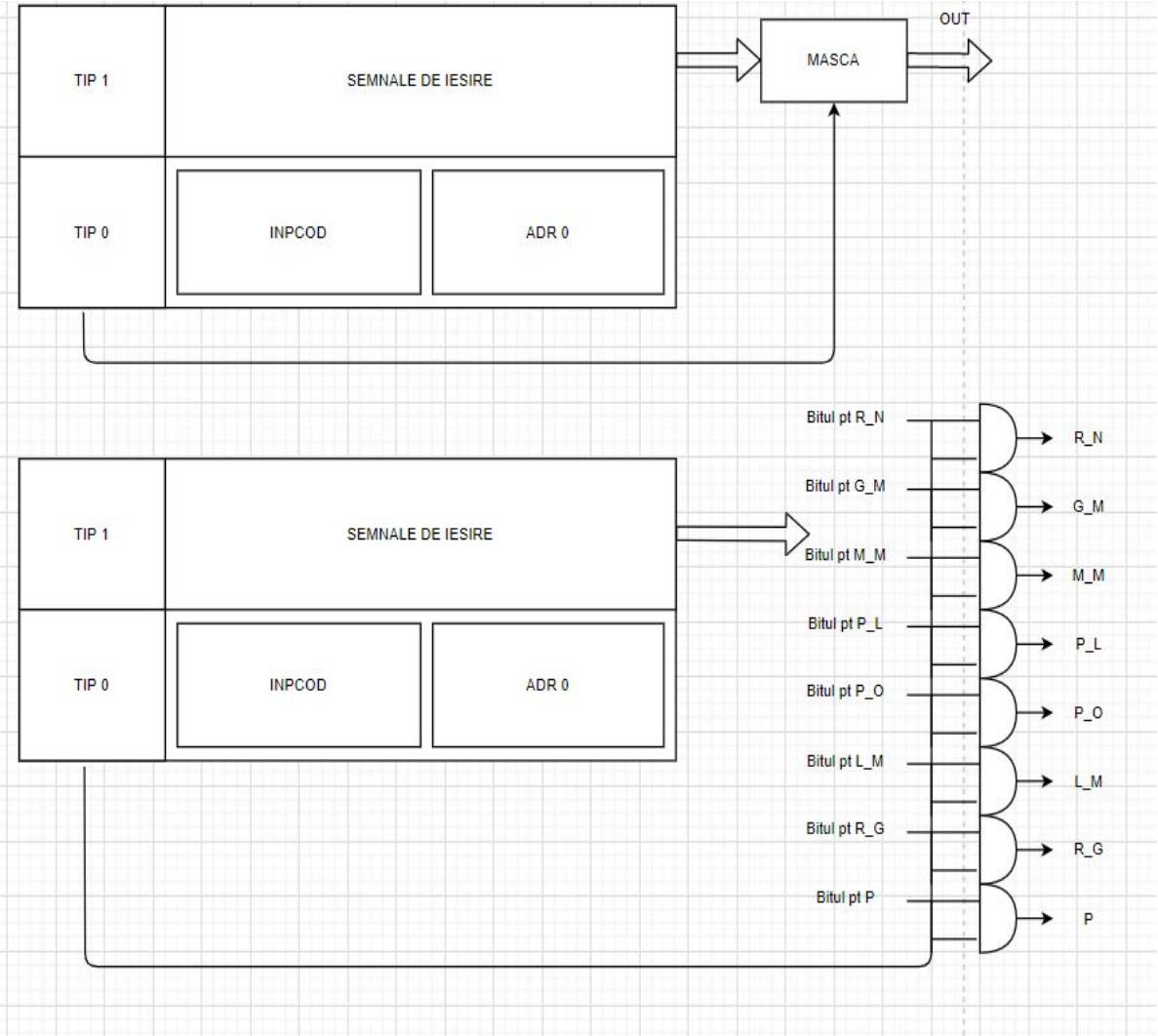
- We chose to use an AND and NOT chip to implement these pins, therefore:

  ⇨ UP =! (! TYPE * W) * CLK

  ⇨ ! LOAD =! (! TYPE * W)

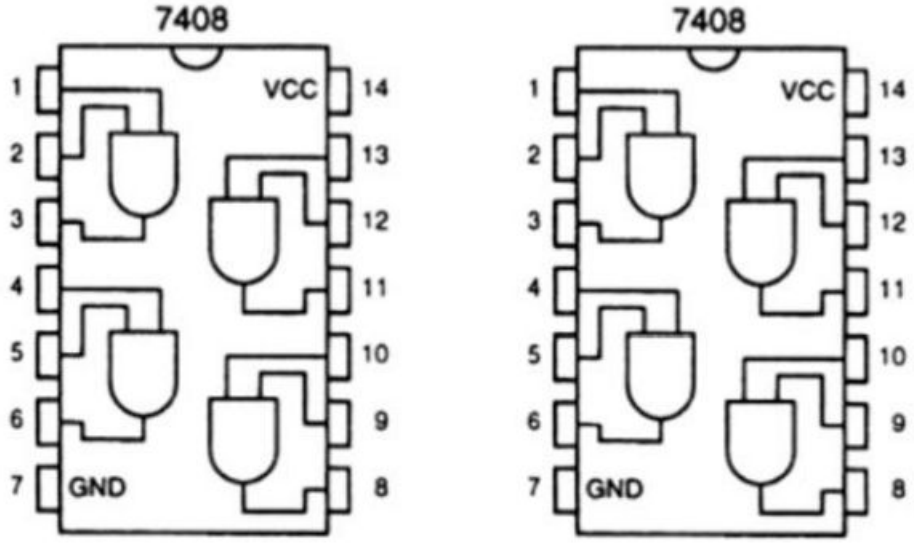
Output mask - theoretically

- Statement type = 1 => outputs are generated
- Statement type = 0 => no outputs are generated



Output mask - basically 7408

- 8 ports AND
- Chips with 4 ports AND => 2 chips 7408



- 1, 2 - input pins for the first gate
- 3 - output pin for the first gate

Drawing of the wiring of the designed circuit