

Proiectarea cu Microprocesoare
Anul universitar 2024-2025
Microsistem cu microprocesorul 8086

Burada Andrei-Alexandru

grupa 3C.1.1

Prof. Coordonator: Sebastian Petruc

Tema proiectului:

Să se proiecteze un microsistem cu următoarea structură:

- unitate centrală cu microprocesorul 8086
- 256 KB memorie EPROM, utilizând circuite 27C2048
- 64 KB memorie SRAM, utilizând circuite 62512
- interfață serială, cu circuitul 8251, plasată în zona 0AF0H – 0AF2H sau 0BF0H - 0BF2H, în funcție de poziția microcomutatorului S1
- interfață paralelă, cu circuitul 8255, plasată în zona 0D70H – 0D76H sau 0C70H – 0C76H, în funcție de poziția microcomutatorului S2
- o minitastatură cu 12 contacte
- 10 led-uri
- un modul de afișare cu 7 segmente, cu 6 ranguri

Toate programele în limbaj de asamblare vor fi concepute sub formă de subrutine. Programele necesare sunt:

1. rutinele de programare ale circuitelor 8251 și 8255;
2. rutinele de emisie/ recepție caracter pe interfața serială;
3. rutina de emisie caracter pe interfață paralelă;
4. rutina de scanare a minitastaturii;
5. rutina de aprindere/ stingere a unui led;
6. rutina de afișare a unui caracter hexa pe un rang cu segmente.

Structura rutinelor (intrări, secvențe, ieșiri) va fi stabilită de fiecare student.

Descrierea componentelor hardware utilizate

Pentru design-ul microsistemului am impartit proiectul in 6 module:

1. Unitatea Centrala

- 1 X procesor intel8086
- 1 X clock_generator 8284A
- 2 X data_bus_amplif 74X245
- 3 X amplificator 74X373
- 1 X crystal

2. Memoria

- 1 X decodor 74X138
- 1 X memorie_EEPROM 27C2048
- 2 X memorie_SRAM 62512

3. Interfetele

- 1 X decodor 74X138
- 2 X switch-uri
- 1 X translator nivel MAX232
- 1 X timer 8253A
- 1 X interfata_seriale 8251A
- 1 X interfata_paralela 8255A

4. Tastatura

- 1 X amplificator 74X373
- 1 X data_bus_amplif 74X244
- 12 X push_buttons
- 4 X rezistenta 10K Ω

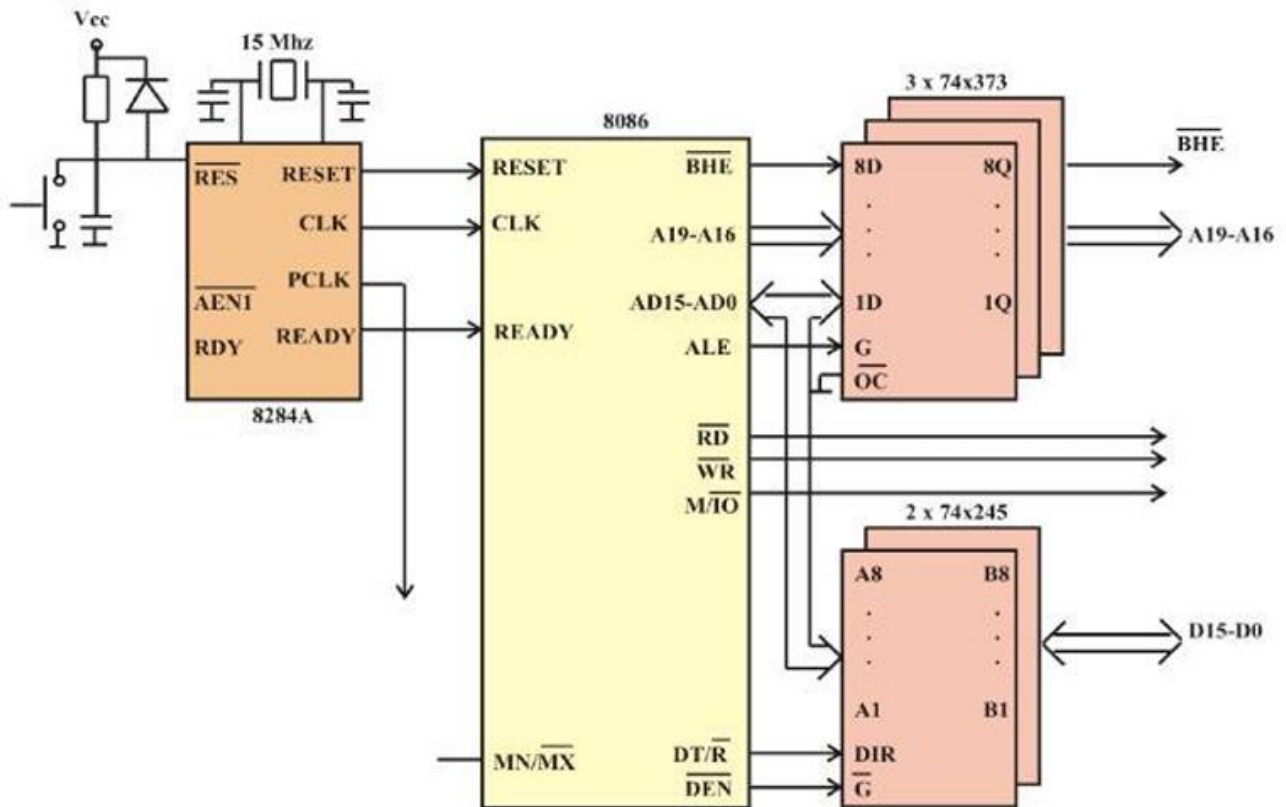
5. Led-uri

- 2 X amplificator 74X373
- 10 X led-uri
- 10 X rezistente 1K Ω

6. Modul de afisare cu segmente

- 6 X ranguri
- 6 X amplificator 74X373
- 48 X rezistenta 330 Ω

1. Unitatea Centrala (schema generalizata)



a) Microprocesorul 8086

Caracteristici:

- registrele interne și magistrala de date externă sunt pe 16 biți
- posibilitatea de a adresa direct 1 MB de memorie
- viteză mărită de lucru datorită atât frecvenței tactului cât și unei structuri interne bazată pe conceptul de suprapunere care permite aducerea din memorie, în avans, a instrucțiunilor în timpul unor cicluri fără acces la magistrale
- poate acoperi o gamă largă de aplicații datorită celor două moduri de lucru ale sale: minim și maxim
- magistralele de date și adrese sunt multiplexate iar o parte dintre terminalele de comandă au rol dublu; aceasta a permis încapsularea circuitului într-o capsulă cu doar 40 terminale

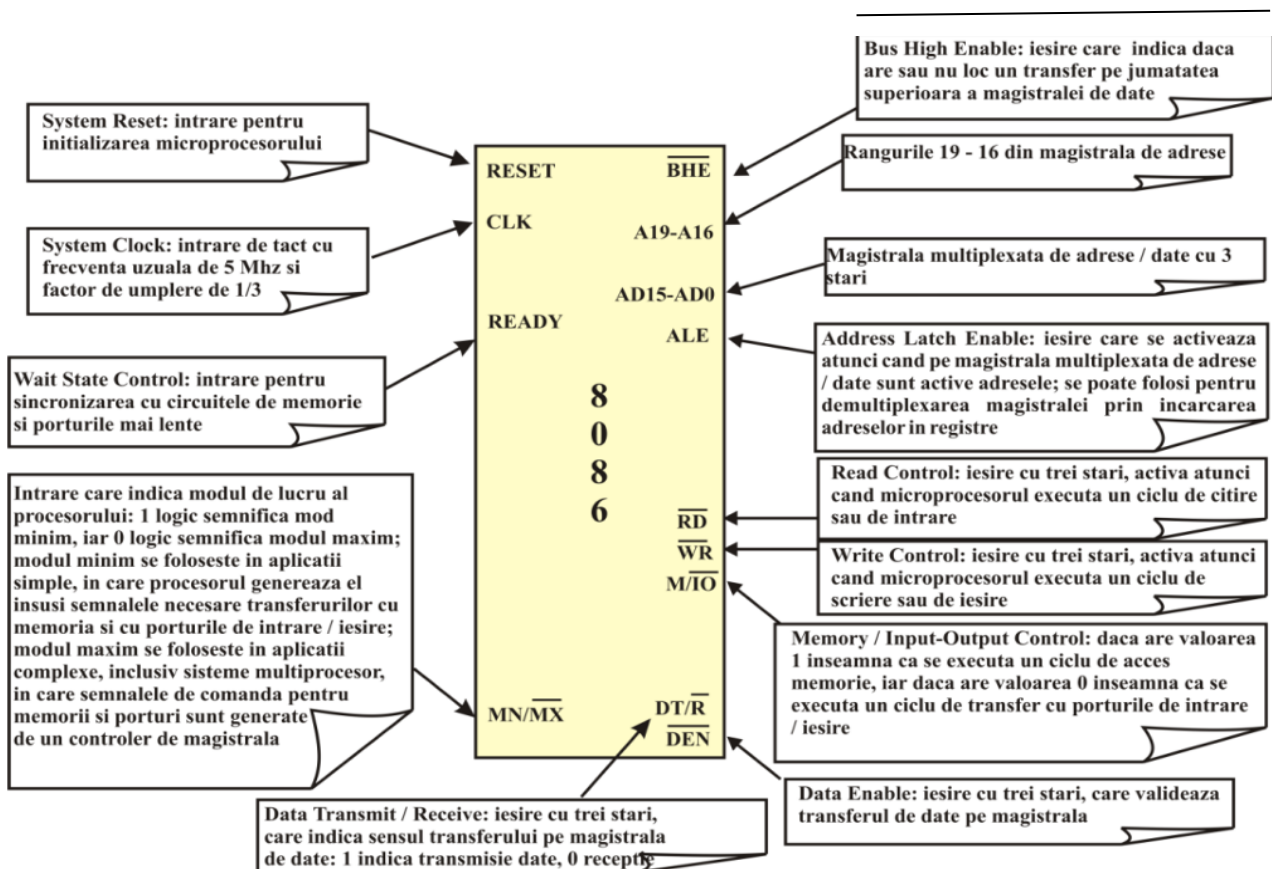
Moduri de lucru:

- **minim:** aplicații relativ simple, în care microprocesorul generează el însuși semnalele necesare transferurilor cu memoria și cu porturile de intrare/ieșire
- **maxim:** aplicații complexe, inclusiv sisteme multiprocesor, în care semnalele de comandă pentru memorii și porturi sunt generate de un controler de magistrală, 8288
- nu oferă privilegii diferite ci ele se recomandă în anumite configurații hardware, pentru tipuri de aplicații diferite,
- trecerea dintr-un mod în altul se face prin hardware: există terminalul MN/ /MX la care, prin 1 logic se cere modul minim iar prin 0 logic se cere modul maxim.

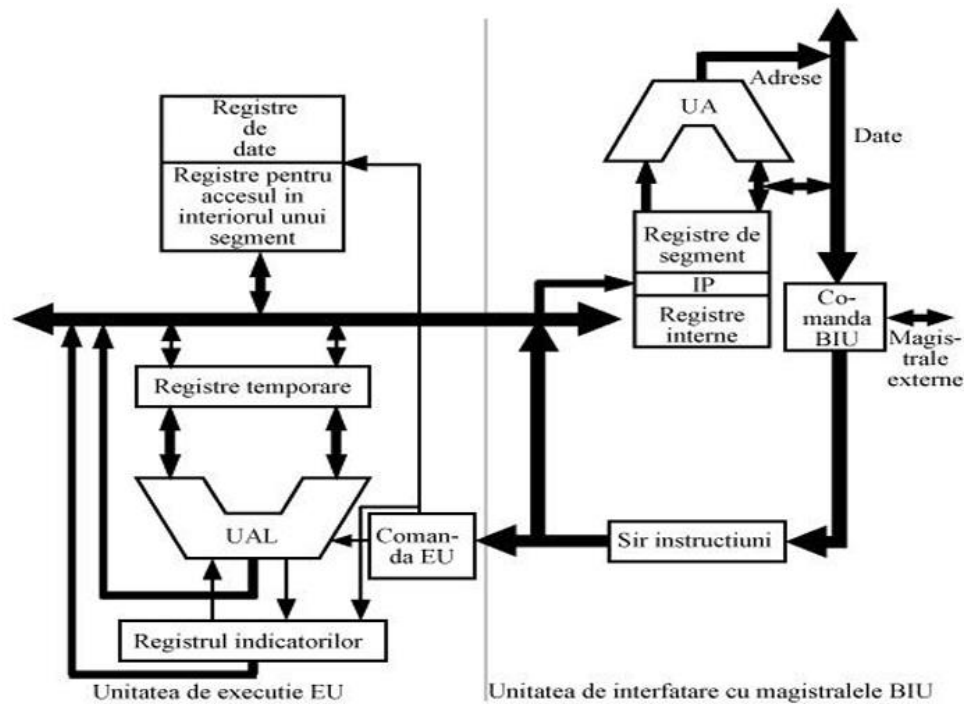
Terminale:

- magistrale multilexate de adrese/date
- linii de comanda si control cu 2 semnificatii

Configuratia terminalelor:



Structura interna:



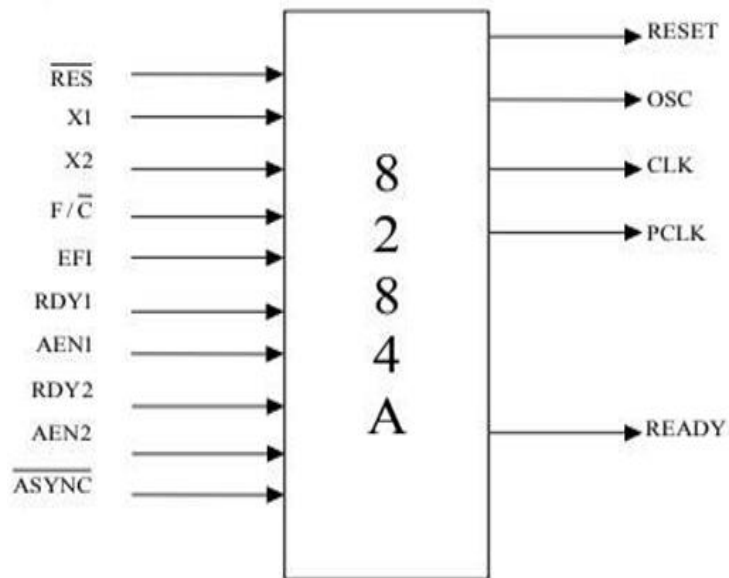
b) Generatorul de tact 8284A

- generează tactul către microprocesor și pentru circuitele specializate pentru interfețe
- generează semnalul READY către microprocesor, sincronizându-l cu tactul și
- generează semnalul de inițializare, RESET, către microprocesor, sincronizându-l cu tactul

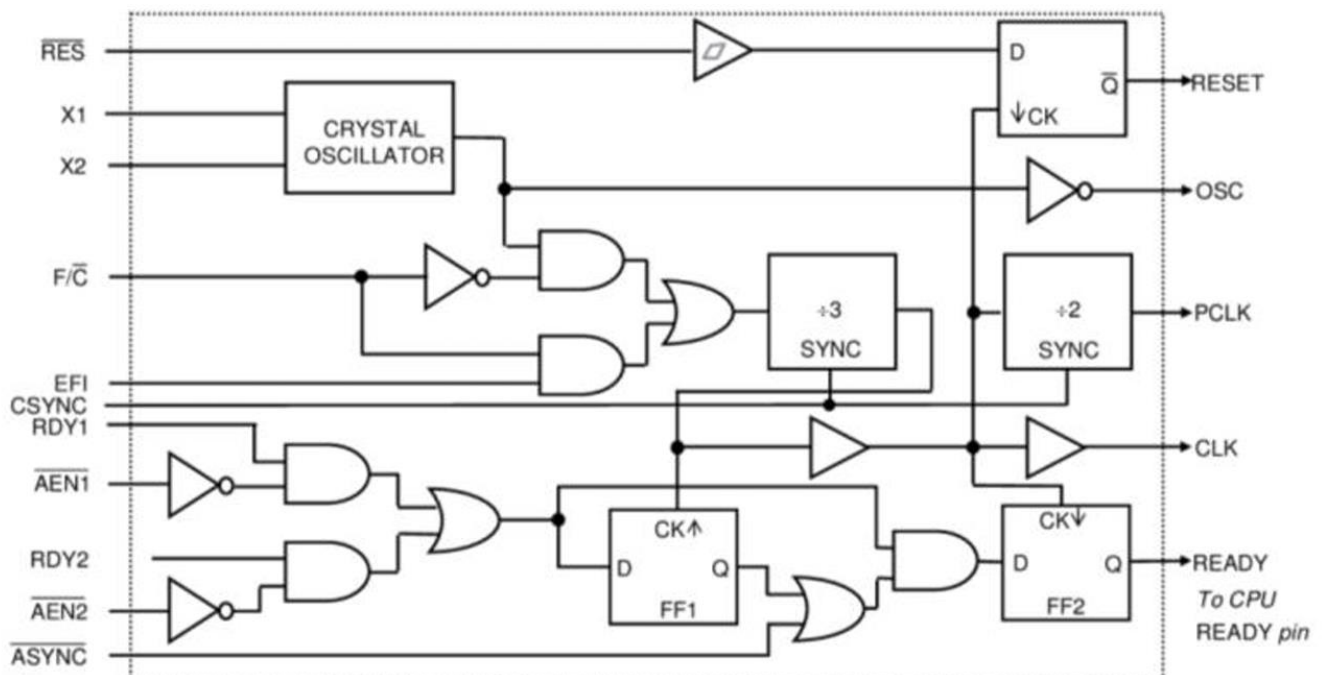
ieșirea READY are rolul de a genera cererea pentru inserarea de stări de așteptare și se poate activa prin intrările RDY1 sau RDY2 care reprezintă intrări pentru cereri externe de stări, active la 1 logic care la rândul lor sunt validate de intrările /AEN1 și /AEN2.

Semnalul RESET care reprezintă ieșirea pentru inițializarea microprocesorului este rezultatul sincronizării intrării de inițializare /RES și frontului căzător al tactului. La intrarea /RES vom conecta un grup RC și un comutator.

Configuratia terminalelor:



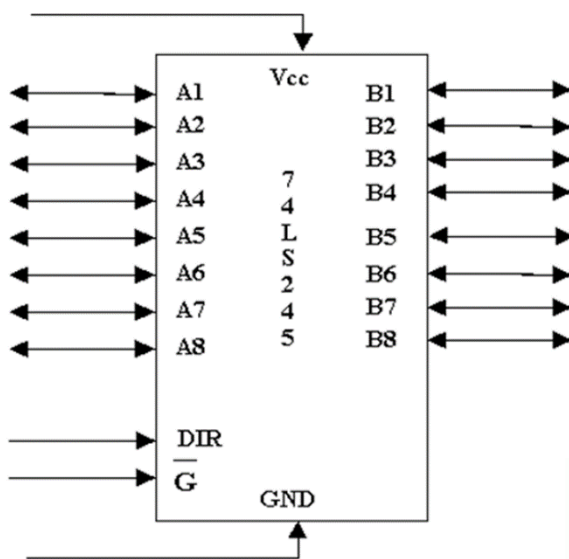
Structura interna:



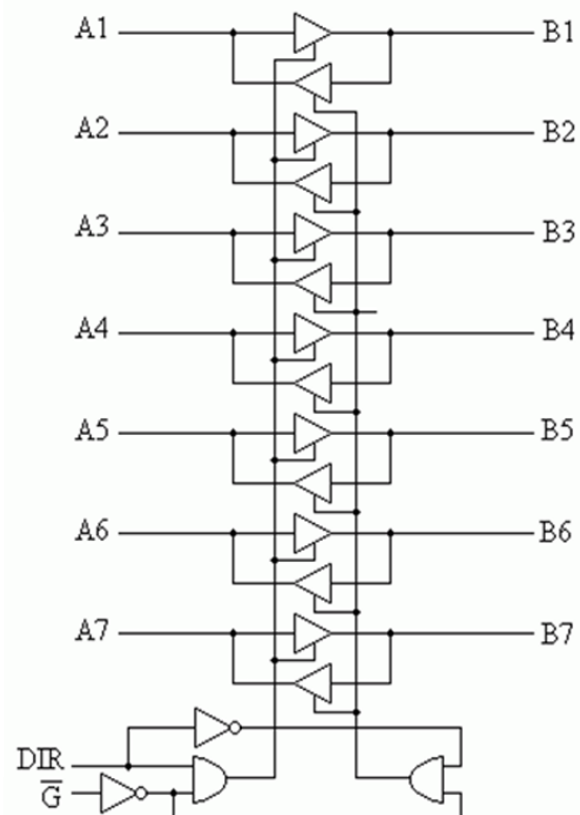
c) Circuitul 74X245

- este un circuit folosit pentru amplificarea/separarea magistralelor bidimensionale ale microprocesoarelor
- este alcătuit din 8 perechi de porți cu 3 stări bidirecționale
- are o singură intrare de validare pentru toate porțile “~CE” la care conectăm semnalul “~DEN” și o intrare pentru stabilirea direcției de transfer

Configuratia terminalelor:



Schema interna:



Proiectarea Microsistemelor Digitale

- Funcționarea:

/G	DIR	A8 – A1	B8 – B1
0	0	B8 – B1	Intrări
0	1	Intrări	A8 – A1
1	X	A 3 – a stare	A 3 – a stare

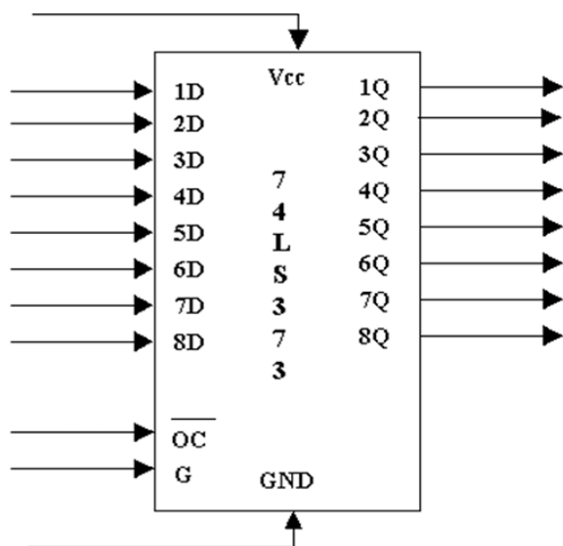
d) Circuitul registru 74X373

- este un circuit registru cu 3 stări folosit pentru demultiplexarea magistralelor
- are două intrări(cu excepția celor pentru date):
 - una pentru validarea tuturor ieșirilor($\sim OE$)
 - una pentru încărcarea bistabilelor (LE)
- la intrarea LE conectăm semnalul ALE al microprocesorului, iar la intrarea $\sim OE$ conectăm la GND pentru ca ieșirile să fie validate încontinuu.

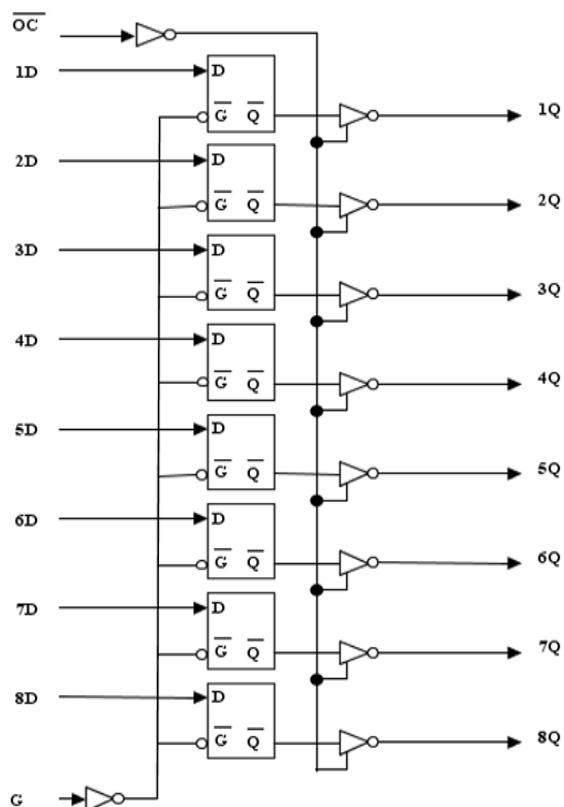
Magistrala conectată la ieșirea $A19-A16$ a microprocesorului și magistrala conectată la ieșirea circuitelor 74LS373 sunt unidirecționale și pe ele circula doar biți de adrese(magistale de adrese).

Dacă microprocesor dorește să transmită o adresă, va activa circuitul 74LS373 prin semnalul ALE și atunci încarcă biții de adrese în regiștrii. Cum ieșirea acestor circuite este întotdeauna validată, registrele vor avea la ieșire biții respectivi.

Configuratia terminalelor:



Schema interna:



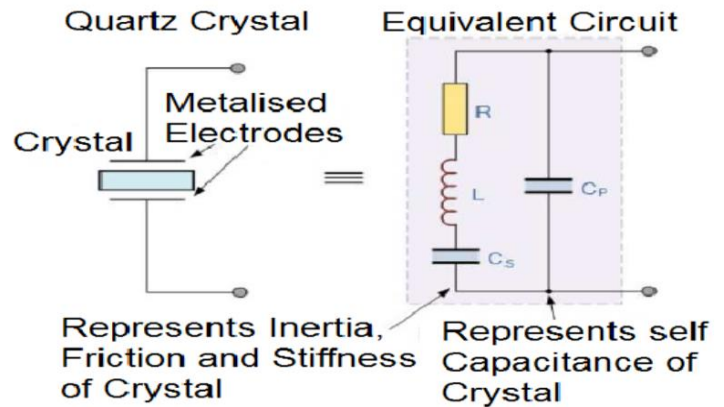
Proiectarea Microsistemelor Digitale

- Funcționarea:

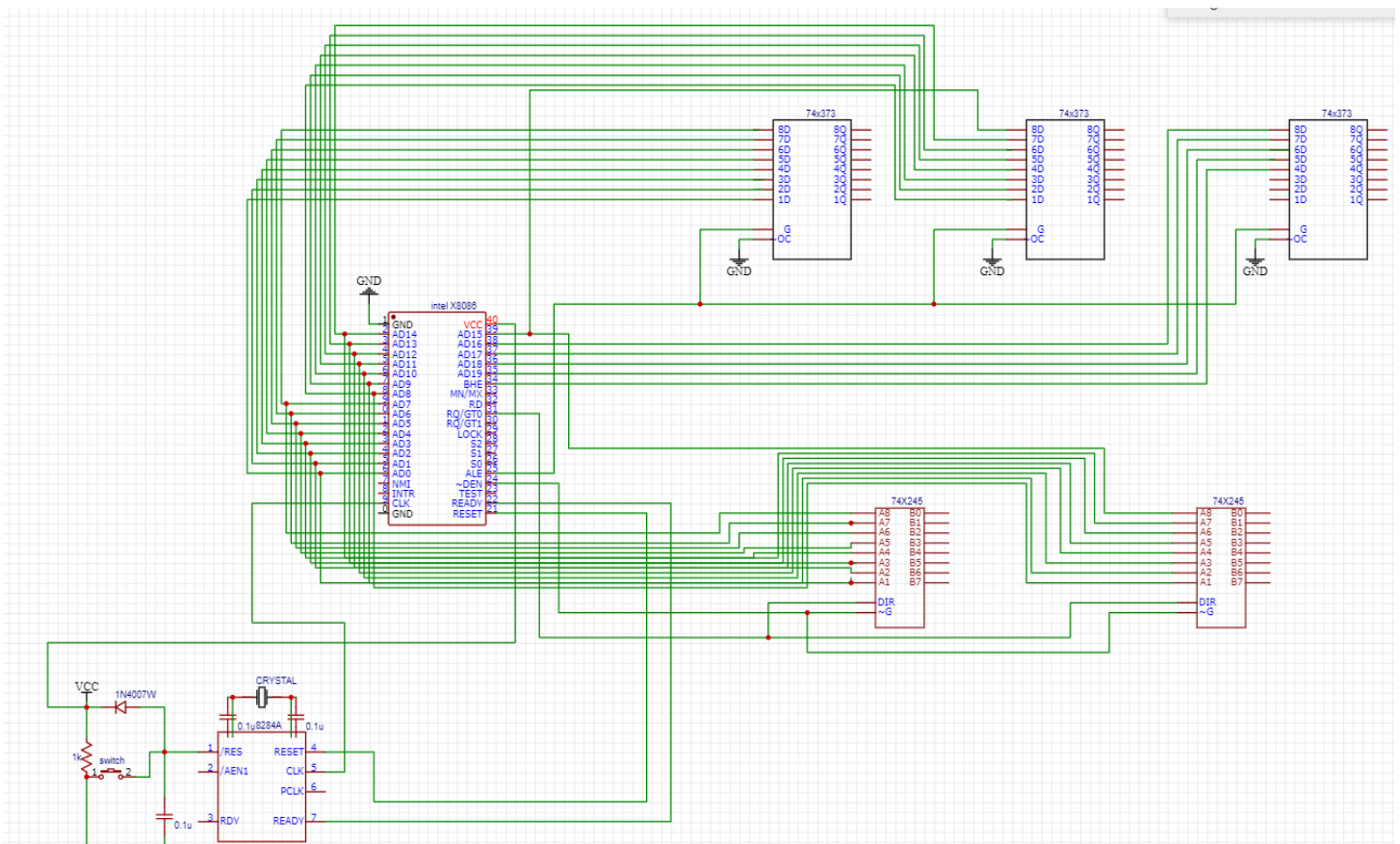
/OC	G	8Q – 1Q
0	0	Vechiul conținut
0	1	8D – 1D
1	X	A 3 – a stare

e) Crystal

- are rolul de a genera un semnal de ceas stabil și precis pentru sincronizarea procesorului și a componentelor
- este fabricat din cuarț iar acesta vibrează la o frecvență specifică atunci când este supus unui câmp electric
- acesta este conectat la generatorul de tact 8284A pentru a sincroniza întreaga funcționare a unității centrale



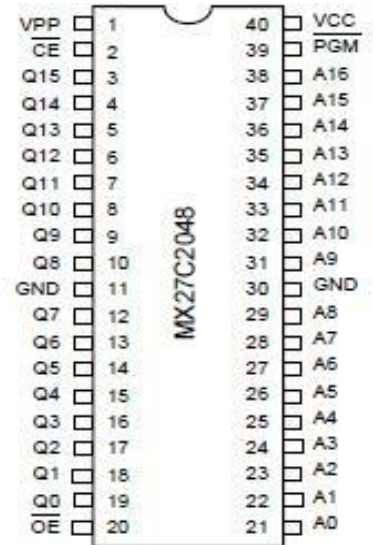
Unitatea Centrala in EasyEDA:



2. Memoria microsystemului

a) *Memoria EPROM* (Erasable Programmable Read Only Memory)

- este un tip de memorie **nevolatilă** (își păstrează datele chiar și când i se întrerupe alimentarea cu curent electric), având timpul de acces de 90-200ns
- ținem cont ca în ciclul de citire a datelor, la intrarea **Vpp** trebuie să fie o tensiune egală cu cea de la **Vcc**
- în cadrul proiectului nostru avem nevoie de o memorie EPROM de capacitatea 256 KB, prin urmare vom folosi doar o memorie 27C2048, deoarece aceasta are configurate 16b ieseire



b) *Memoria SRAM* (Static Random Access Memory)

- este un tip de memorie semiconductor RAM(Random-Access Memory) ce folosește un flip-flop ca să memoreze fiecare bit de date, spre deosebire de memoriile **DRAM** (Dynamic Random Access Memory), unde este necesar un ciclu periodic de refresh
- este o memorie **volatile** (pierde datele dacă este întreruptă de la sursa de current), având timpul de acces 45 – 84 ns
- are ca scop implementarea **memoriei cahe pentru CPU**
- în cadrul proiectului nostru avem nevoie de o memorie SRAM de capacitatea 64KB, ceea ce este susținută de o singură componentă 62512, **DAR** o componentă poate gestiona simultan numai **8b** la intrare, noi având implementat o magistrală cu **16b**, prin urmare vom folosi 2 componente 62512 conectate la un decodor 74X138 pentru a gestiona datele primite



www.romlab.prv.pl
siudym@epf.pl

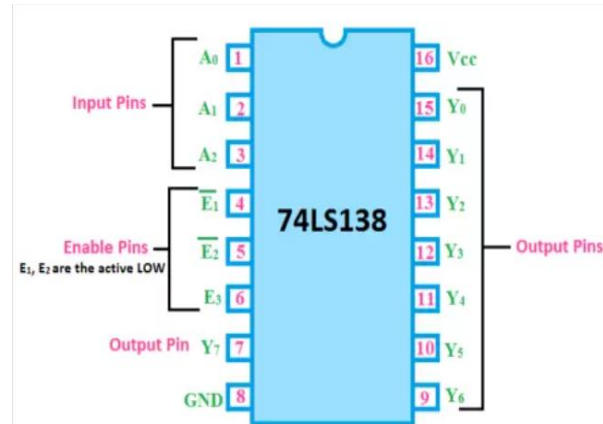
62512 RAM

2.1 Construirea decodicatorului de memorii

Pentru a reusi mapa zonele de meorie necesare pentru functionarea microsistemului avem nevoie de un decodor '3 la 8'. Pentru acest tip de decodor vom folosi componenta **74X138**:

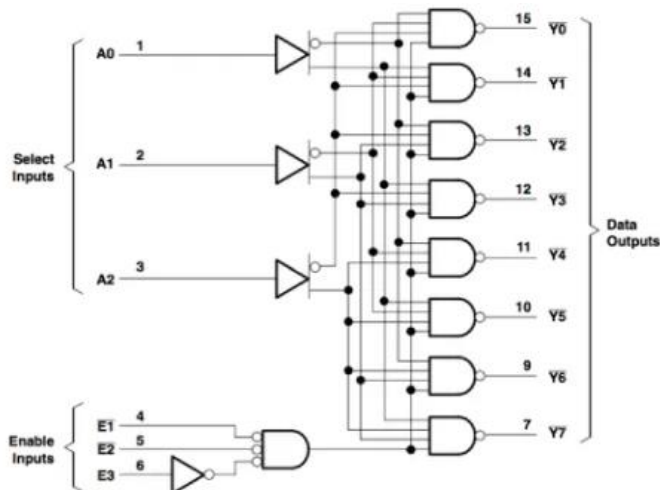
Caracterisitici:

- activează o singură linie la ieşire corespunzătoare codului de intrare, aceasta linie devine 0 si restul sunt setate pe 1
- are 3 intrari (**A0,A1,A2**) care prin transformarea din binar in zecimal se activeaza iesirea cu indexul respective**Y(0-7)**
- are 3 intrari pentru selectia circuitului active pe 1 logic **E3**, respectiv pe 0 logic **E1, E2**.



configuratia terminalelor

Schema interna:



Functionarea:

E3	/E2	/E1	C	B	A	/Y7	/Y6	/Y5	/Y4	/Y3	/Y2	/Y1	/Y0
1	0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	0	1	0	1	1	1	0	1	1	1	1	1
1	0	0	1	1	0	1	0	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1	1	1	1
0	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	1	X	X	X	1	1	1	1	1	1	1	1

2.2 Alegerea numarului componentelor de memorie utilizate

- 1 componenta EPROM_27C2048**, aceasta oferind memoria precizata in cerinta (256 KB) precum si numarul necesar de porturi de iesire(16) pentru a comunica in mod corect cu magistrala pe 16biti a microsistemului implementat
- 2 componente SRAM_62512**, datorita faptului ca avem nevoie de 16 porturi de iesire pentru SRAM, capacitatea unei componente fiind suficienta(64 KB), dar numarul de porturi de iesire(8) este insuficient pentru magistrala proiectului nostru

2.3 Determinarea adreselor de start & final / block

Pasul 1: calculam bitul in functie de marimea block-ului

SRAM_1 / SRAM_2: 64 KB = $2^6 * 2^{10} = 2^{16} \Rightarrow$ **A16** -> '1'

EPROM: 256 KB = $2^8 * 2^{10} = 2^{18} \Rightarrow$ **A18** -> '1'

Tipul de memorie	Adresa de start	Adresa de final
SRAM_1	0X00000	0X0FFFF
SRAM_2	0X10000	0X1FFFF
EPROM	0X20000	0X5FFFF

!IMPORTANT! Pentru a sporii eficienta accesarii memoriei microsistemului, vom incepe maparea cu cele 2 zone de adrese pentru 2 SRAMs, deoarece acestea sunt divizori ai zonei mai mari de memorie necesara maparii EPROM-ului. Un alt aspect este maparea tuturor memoriilor intr-o zona continua, deci cele 2 SRAM vor fi primele mapate, urmate imediat de EPROM.

Pasul 2: crearea hartii de memorie

zona mem.	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	adrese S/E
S1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	start
	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	end
S2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	start
	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	end
E	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	start
	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	end

Pasul 3: determinarea ecuatiilor semnalelor de selectie

$SEL(SRAM_1) = \sim A19 + \sim A18 + \sim A17 + \sim A16$

$SEL(SRAM_2) = \sim A19 + \sim A18 + \sim A17 + A16$

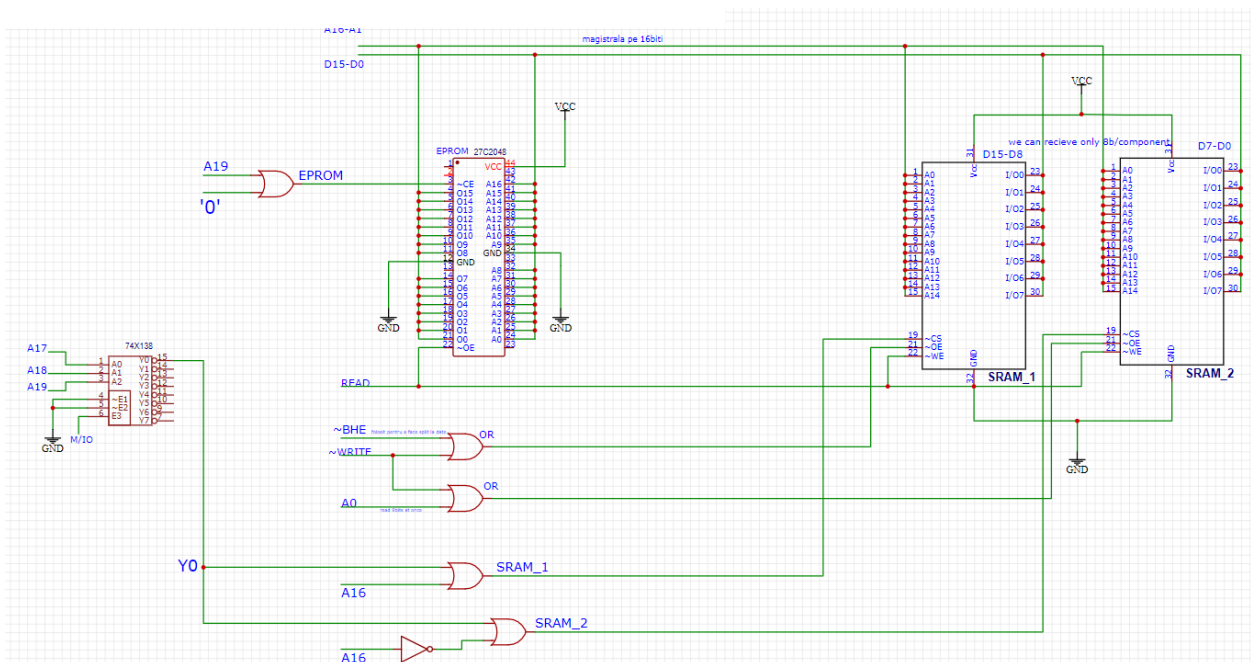
$SEL(EPROM) = \sim A19$

2.4 Implementarea semnalelor de selectie in schema

Pentru a putea selecta in mod corect intre cele 3 zone de memorie ale microsistemului avem nevoie de 4 biti(**A19, A18, A17, A16**).

- **selectia pentru zona SRAM_1 si SRAM_2** o vom realiza folosind un decodicator de tip 3-la-8(74X138) ce va primi ca input bitii A19, A18, A17, deoarece pentru a putea selecta cele 2 zone de SRAM trebuie sa ne asiguram ca valoarea bitilor de input este '0'. Ne asiguram de acest lucru prin folosirea iesirii **Y0**, iar diferentierea intre SRAM_1 si SRAM_2 o face bitul A16, ce este trecut prin poarta SAU impreuna cu Y0 pentru a verifica ca toti bitii sunt '0' pentru SRAM_1, respectiv A16 in cazul cand este '1' va fi negat si trecut tot cu Y0 prin poarta SAU pentru selectia lui SRAM_2.
- **selectia pentru zona EPROM** este realizata in mod simplu prin folosirea unei porti SAU, ce primeste bitul A19 impreuna cu un bit setat manual pe valoarea '0'. Astfel, semnalul de selectie pentru EPROM este respectat intrucat este influentat doar de bitul A19, ce trebuie sa aiba valoarea '0'.

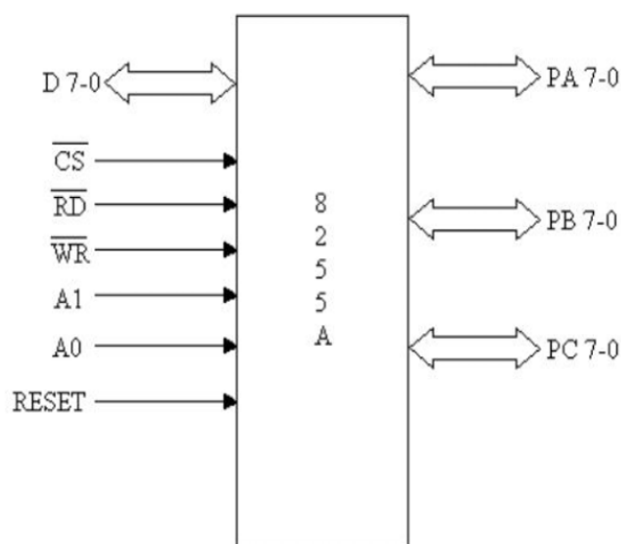
Conectarea Memoriilor in EasyEDA:



/CS	/RD	/WR	C/D	Operație
1	X	X	X	Magistrala de date în a 3-a stare
0	1	1	X	Magistrala de date în a 3-a stare
0	0	1	1	Citire a octetului de stare
0	0	1	0	Citire a datei
0	1	0	1	Scriere a cuvintelor de comandă
0	1	0	0	Scriere a datei

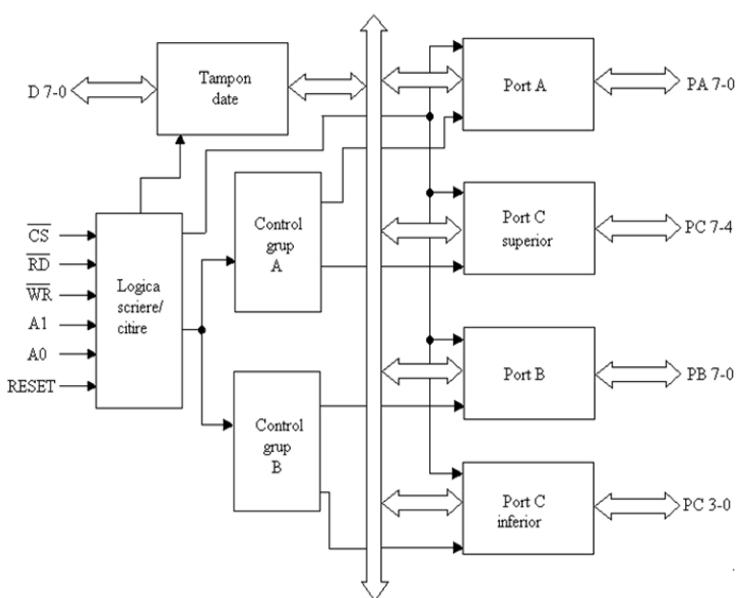
3.2 Interfata paralela (8255A)

- comunicarea cu circuitul 8255 se face cu ajutorul a **4 adrese de port**, corespunzătoare porturilor A, B, C și 1 port pentru cuvântul de comanda
- comunicarea cu magistrala de date a unității centrale de prelucrare se face prin intermediul **bufferului magistralei de date** pe liniile D0 - D7, unde prin acestea se transmit **date, cuvinte de control / stare**
- are **24 linii de intrare/ieșire** ce pot fi configurate după modul de lucru:
 - 2 grupe de 12 linii de intrare sau ieșire, fără semnale de dialog
 - 2 grupe de câte 8 linii de intrare sau ieșire, cu semnale de dialog
 - 1 grupă de 8 linii bidirecționale, cu semnale de dialog
- 16 linii au posibilități de **memorare**



configuratia terminalelor

Schema interna:



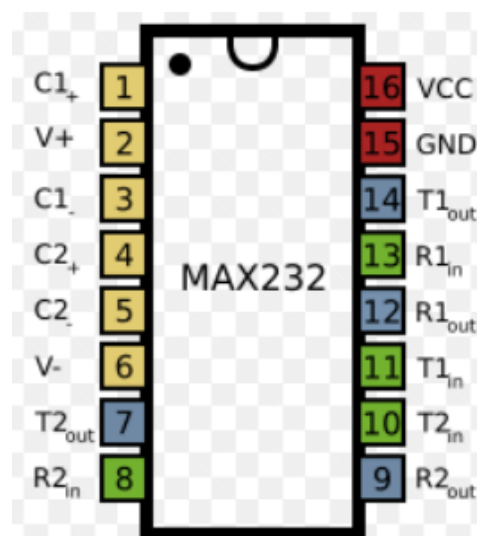
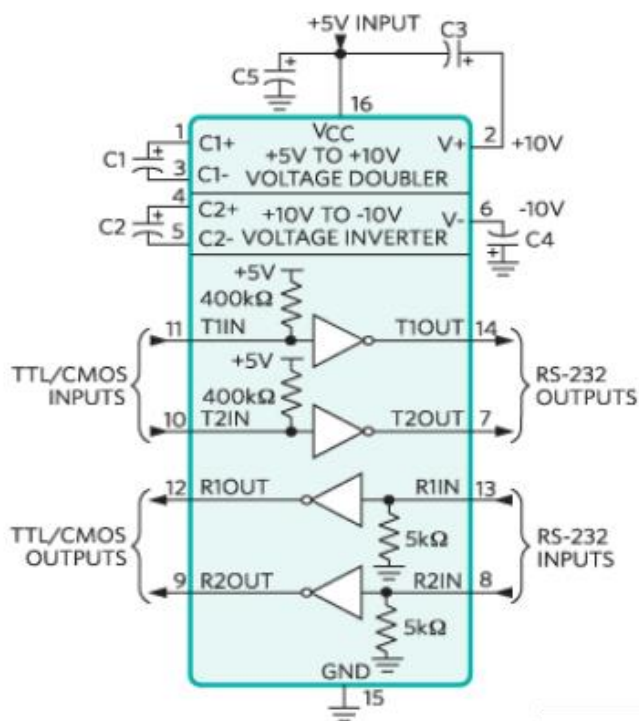
legatura intre operatiile realizate de circuit si starea terminalelor de

/CS	/RD	/WR	A1	A0	Operația
0	1	0	0	0	Scriere în portul A
0	1	0	0	1	Scriere în portul B
0	1	0	1	0	Scriere în portul C
0	1	0	1	1	Scriere în portul cuvântului de comandă
0	0	1	0	0	Citire din portul A
0	0	1	0	1	Citire din portul B
0	0	1	1	0	Citire din portul C
0	0	1	1	1	Fără operație – magistrala de date este în a 3-a stare
0	1	1	x	x	Fără operație – magistrala de date este în a 3-a stare
1	x	x	x	x	Magistrala de date este în a 3-a stare

3.3 Translatorul de nivel (MAX232)

- este un circuit integrat ce convertește semnalele dintr-un port serial **TIA 232 (RS-232)** in semnale adecvate pentru circuitele TTL din proiectul nostru, precum interfata seriala(8251A) si interfata paralela(8255A)
- foloseste protocolul de nivele de tensiune **EIA**, unde:
 - '0' -> (-12) V
 - '1' -> (+12) V
- este un **transmițător / receptor dual** ce este de obicei folosit pentru a converti semnalele RX si TX in semnalele RXD si TXD adecvate pentru interfetele noastre de tip TTL, ce asteapta tensiunile:
 - '0' -> 0 V
 - '1' -> 5 V
- principalul sau rol este de a **asigura reducerea tensiunii** la un nivel optim pentru circuitele TTL din proiect, altfel acestea ar putea fi afectate

Schema interna:



configuratia terminalelor

lista cu valoarea capacitatilor din componenta lui MAX232

CAPACITANCE (μF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	0.047	0.33	0.33	0.33	0.33
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

3.4 Decodificarea porturilor (pentru interfata seriala & paralela)

Pentru o decodificare corecta a porturilor interfetelor vom folosi un decodificator 74X138, ce pentru a functiona in mod corect are nevoie la **pinii de selectie ($\sim E1, \sim E2, E3$)** de valorile (0, 0, 1), obtinute prin folosirea semnalului **A6** pentru '1' logic, respectiv (**A0+A3+A12+A13+A14, A15**) pentru a obtine '0' logic.

Pentru cei **3 pini de input(A0, A1, A2)** vom folosi semnalele (**A8, A9, A10**), deoarece acestea identifica in mod corect zona interfetei seriale de zona interfetei paralele.

Interfata SERIALA este plasata in zona (0x0AF0 – 0x0AF2) sau (0x0BF0 – 0x0BF2) conform switch-ului S1

Zona **0x0AF0 – 0x0AF2**

- 1) port comenzi: 0x0AF0
- 2) port date: 0x0AF2

Zona **0x0BF0 – 0x0BF2**

- 1) port comenzi: 0x0BF0
- 2) port date: 0x0BF2

Interfata PARALELE plasata in zona (0x0D70 – 0x0D76) sau (0x0C70 – 0x0C76) conform switch-ului S2




Zona **0x0D70 – 0x0D76**

- 1) port A: 0x0D70
- 2) port B: 0x0D72
- 3) port C: 0x0D74
- 4) port D: 0x0D76

Zona **0x0C70 – 0x0C76**

- 1) port A: 0C70
- 2) port B: 0C72
- 3) port C: 0C74
- 4) port D: 0C76

zonele destinate	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	adresa port
interf. SERIALA	0	0	0	0	1	0	1	0	1	1	1	1	0	0	0	0	0x0AF0
	0	0	0	0	1	0	1	0	1	1	1	1	0	0	1	0	0x0AF2
	0	0	0	0	1	0	1	1	1	1	1	1	0	0	0	0	0x0BF0
	0	0	0	0	1	0	1	1	1	1	1	1	0	0	1	0	0x0BF2
interf. PARALELA	0	0	0	0	1	1	0	0	0	1	1	1	0	0	0	0	0x0C70
	0	0	0	0	1	1	0	0	0	1	1	1	0	0	1	0	0x0C72
	0	0	0	0	1	1	0	0	0	1	1	1	0	1	0	0	0x0C74
	0	0	0	0	1	1	0	0	0	1	1	1	0	1	1	0	0x0C76
	0	0	0	0	1	1	0	1	0	1	1	1	0	0	0	0	0x0D70
	0	0	0	0	1	1	0	1	0	1	1	1	0	0	1	0	0x0D72
	0	0	0	0	1	1	0	1	0	1	1	1	0	1	0	0	0x0D74
	0	0	0	0	1	1	0	1	0	1	1	1	0	1	1	0	0x0D76

-  bitii folositi pentru intrarile A0, A1, A2
-  bitul folosit pentru a aduce '1' la pinul E3
-  bitii ce sunt toti pe '0' sunt conectati la E2

3.5 *Determinarea ecuatiile semnalelor de selectie*

Calculam ecuatiile pentru fiecare zona a interfetei seriala / paralela conform valorilor din tabelul anterior si vom determina care iesire a decodorului va fi activa in fiecare caz.

SEL(SERIAL 1) = $\sim A_{10} + A_9 + \sim A_8 \Rightarrow \sim Y2$ **ACTIV**

SEL(PARALEL_1) = A10 + ~A9 + A8 => ~Y5 ACTIV

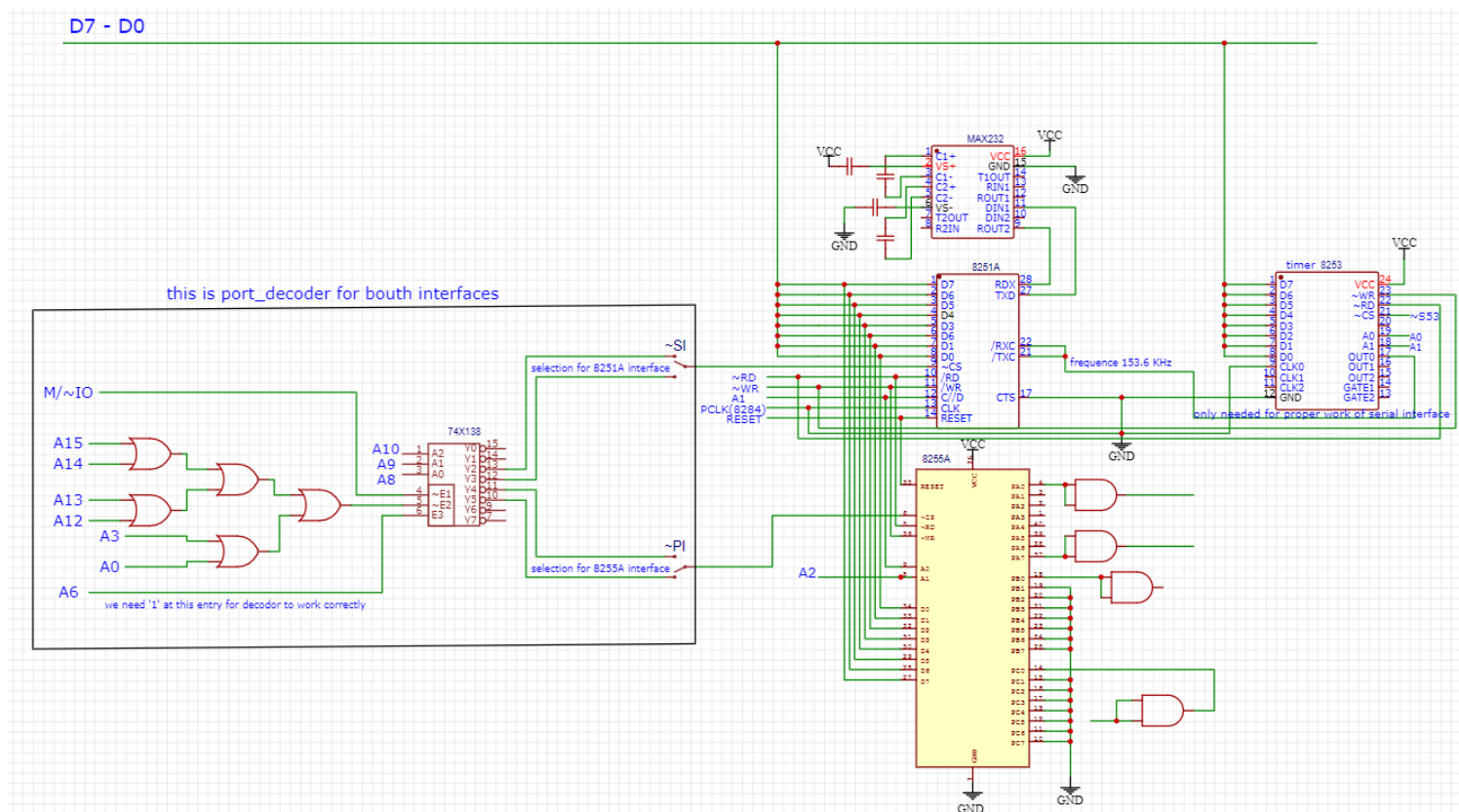
$$\text{SEL}(\text{SERIAL } 2) = \sim A_{10} + A_9 + A_8 \Rightarrow \sim Y3 \text{ ACTIV}$$

SEL(PARALEL 2) = A10 + ~A9 + ~A8 => ~Y4 **ACTIV**

!!IMPORTANT!! Am inclus partea de decodificare a porturilor pentru interfata seriala & paralela in aceeași schema cu implementarea lor, deoarece am considerat ca este mai lizibil de urmarit circuitul si toate datele relevante unei subparti a proiectului se afla grupate in aceeași schema.

In plus, am folosit in schema destinata interfetelor si un **timer 8253**, ce este absolut necesar pentru buna functionarea a interfetei seriale. Acesta generează rata de transfer pentru interfața serială, fiind conectat la terminalele TxC și RxC ale interfetei seriale 8251.

Interfetele in EasyEDA:

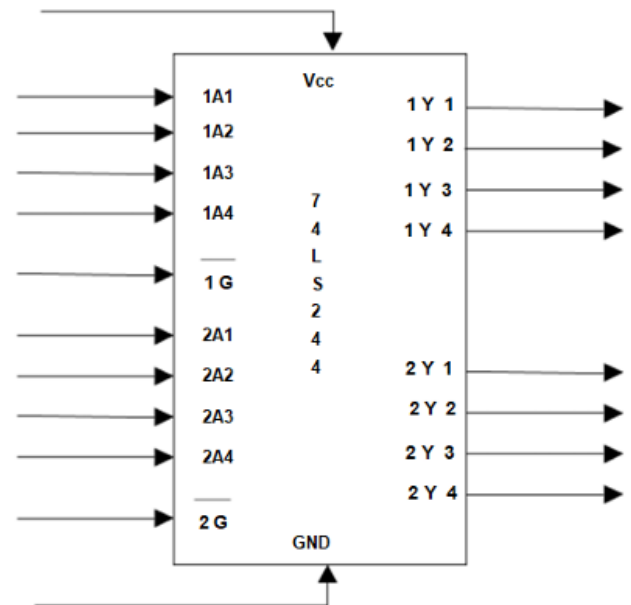
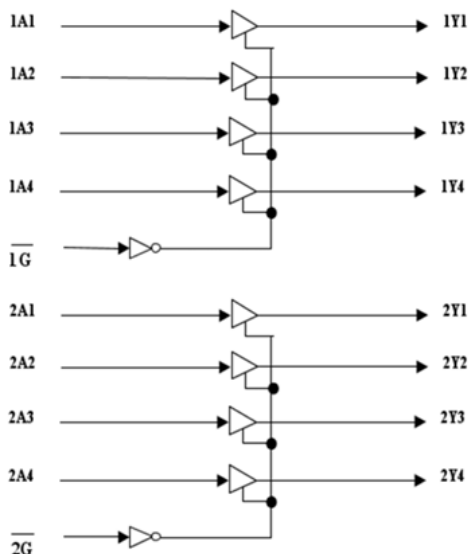


4. Perifericele *(tastatuta, led-uri, modulul afisare)*

4.1 Circuitul amplificator / separator (74X244)

- are o mare importanta pentru conectarea perifericelor deoarece asigură izolarea semnalelor, **prevenind conflictele pe magistrala de date** și **garantând transferul corect al informației**
- are in componenta un buffer logic care oferă protecție între circuite și poate fi utilizat pentru a preveni conflictele pe magistrala de date
- are 2 grupuri de câte 4 intrări și ieșiri fiecare, controlate independent prin semnale de activare (**/OE1, /OE2**), ce permit gestionarea eficientă a conexiunilor

Schema interna:



configuratia terminalelor

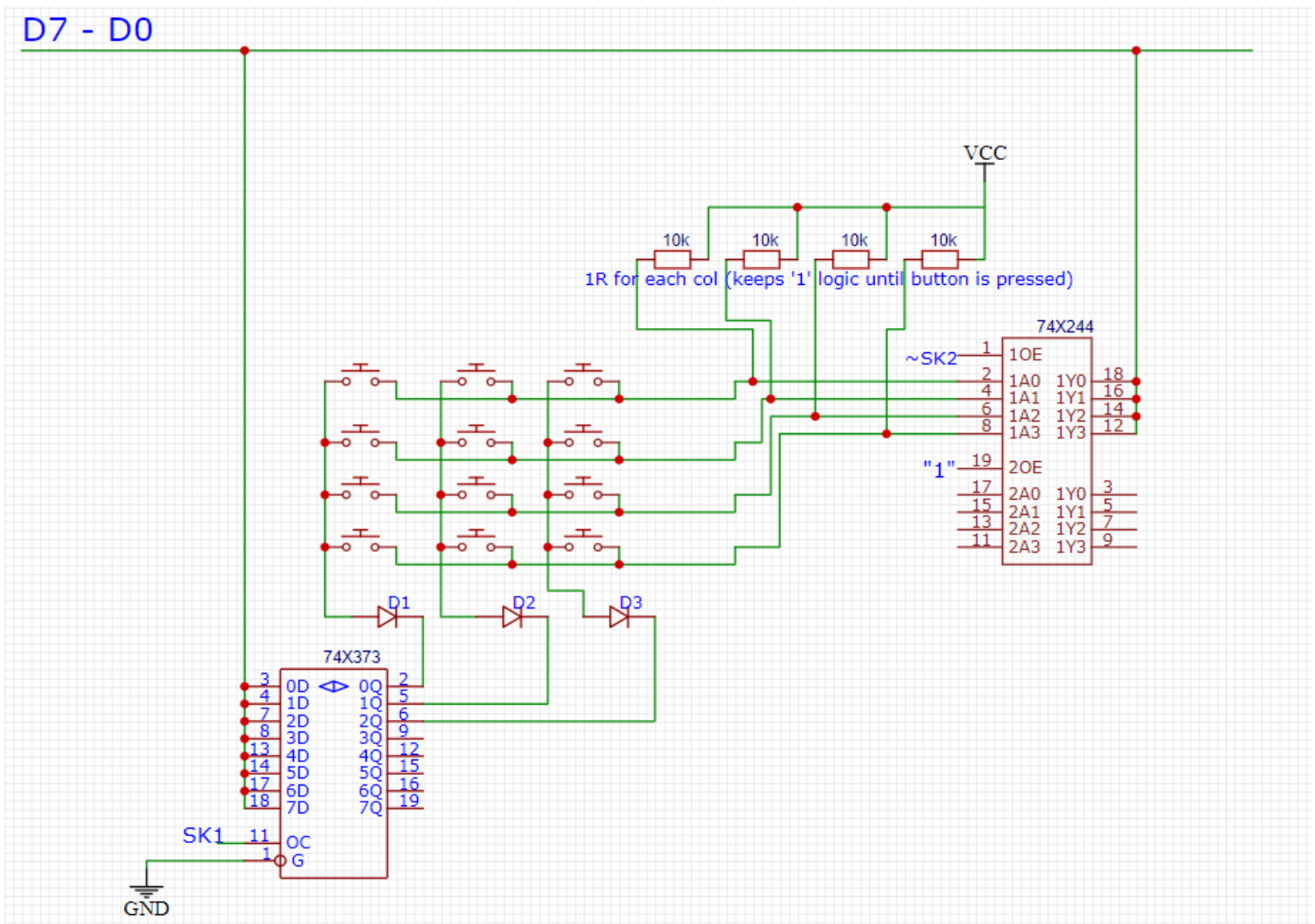
Tabelul de functionare:

1/G	2/G	1Y4-1Y1	2Y4-2Y1
0	0	1A4-1A1	2A4-2A1
0	1	1A4-1A1	a 3-a stare
1	0	a 3-a stare	2A4-2A1
1	1	a 3-a stare	a 3-a stare

4.2 Tastatura

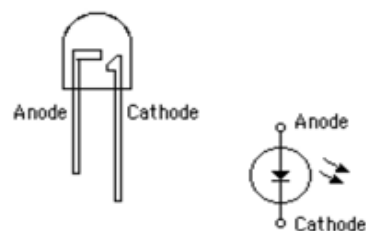
- este formata din 12 contacte, iar pentru a identifica tasta apasata trebuie citita tastatura cu o rutina de programare
- are 2 porturi pentru functionare: unul de iesire, prin care se activeaza coloana si unul de intrare, prin care se citeste linia activa. In portul de iesire se va scrie '0' logic numai pe o coloana, iar pe restul '1' logic si se citesc linii
- daca pe o linie se detecteaza '0' logic atunci tasta a fost actionata
- conectarea tastaturii la micro sisteme se face cu ajutorul unui circuit 74x373 (**portul de iesire**) si al unui circuit 74x244 (**portul de intrare**), avand semnalul de selectie **~SK2**

Tastatura in EasyEDA:



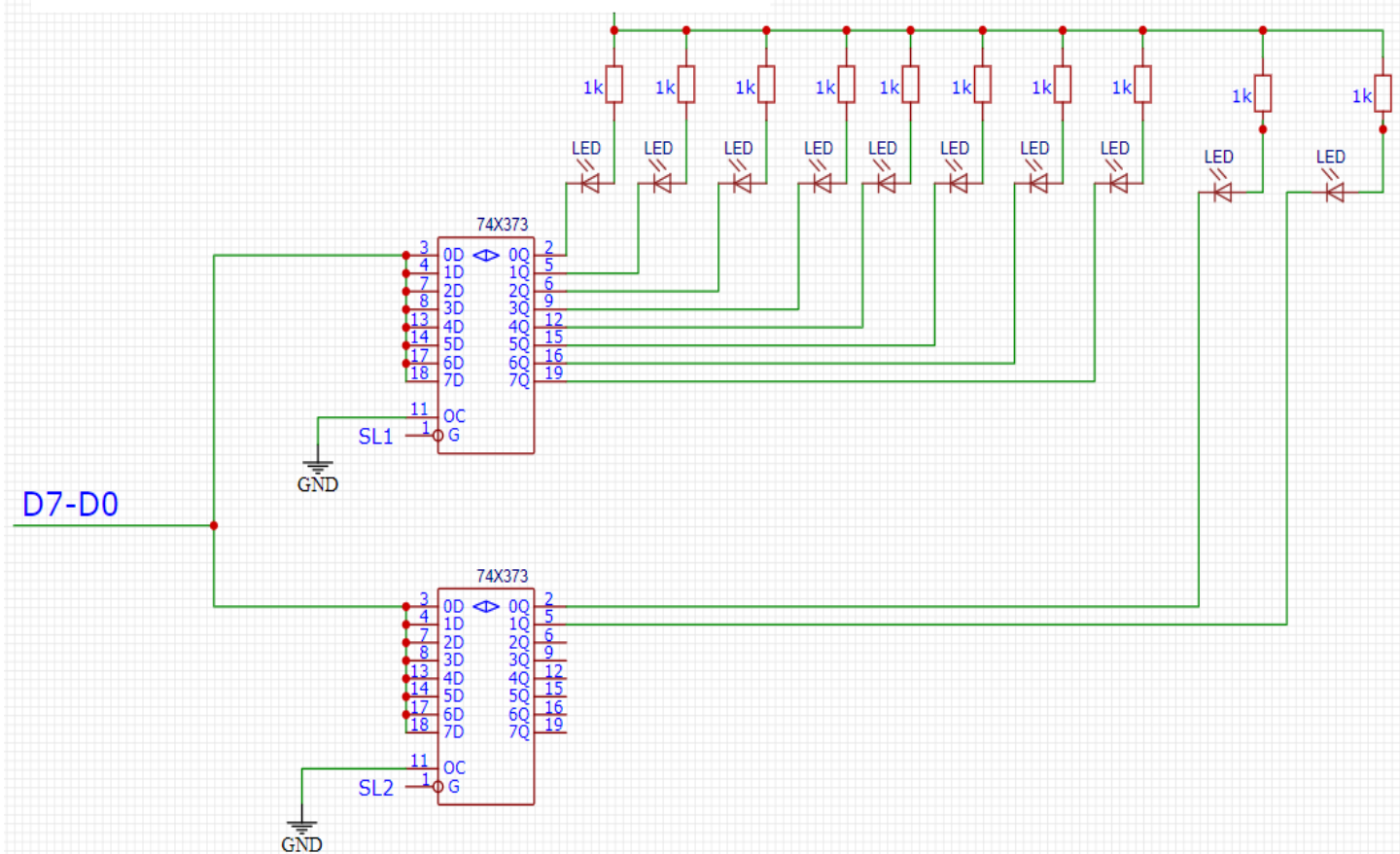
4.3 LED-urile

- sunt **diodे semiconductorе**, iar pentru ca un led sa fie aprins la iesirea portului trebuie sa fie '0' logic
- in proiect folosim 10 led-uri, prin urmare avem nevoie de **2 amplificatoare 74X373**, deoarece o componenta suporta doar 8 iesiri, iar numarul led-urilor este mai mare
- se vor folosi LED-uri pe care caderе de tensiune este de 1,6V. Curentul de lumina re al LED-ului este de 10mA, iar caderе pe bistabil este de 0.2V. LED-urile se conectează la VCC(5V)
- este nevoie de o diferentă de potențial pentru ca să obținem curentul necesar pentru aprinderea LED-ului. Rezultă că LED-ul se aprinde atunci când avem în bistabil '0' logic



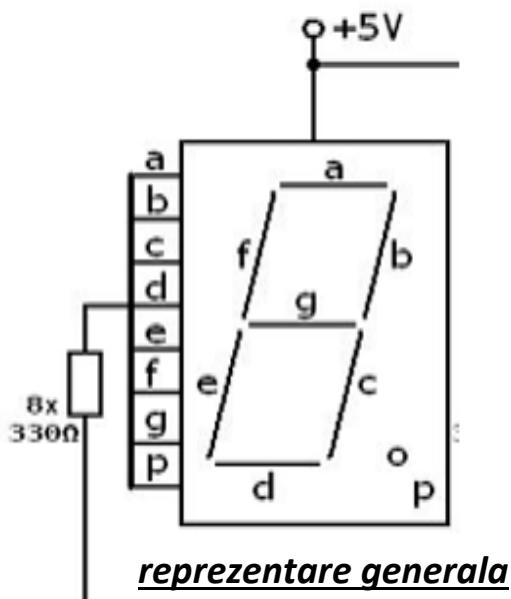
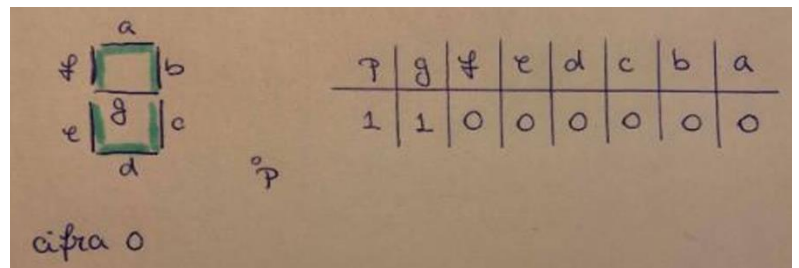
reprezentare generala

LED-urile in EasyEDA:

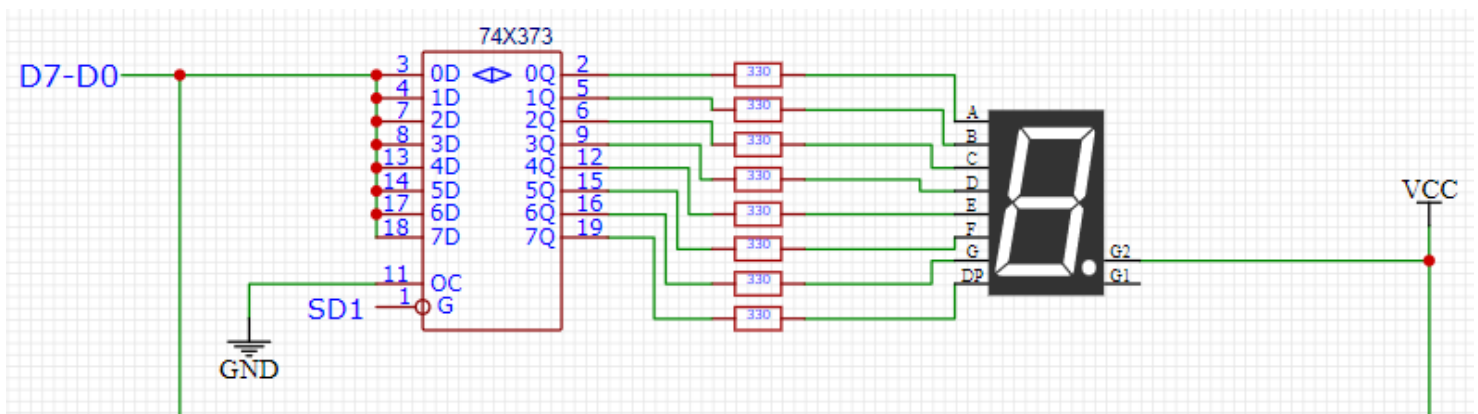


4.4 Modul de afisare cu segmente(6 ranguri)

- in acest proiect se vor folosi circuite afişaj cu **anod comun**, ceea ce înseamnă că toţi anozii vor fi conectaţi la un terminal la care se conectează VCC, iar catodii sunt accesibili la terminalele circuitului
- pentru ca un segment să lumineze este necesară comanda terminalului la care este accesibil catodul său, cu '0' logic, prin intermediul unei rezistenţe
- dimensionarea rezistenţei** este : $(5-1.6-0,2)V/10mA \cong 330 \Omega$
- la intrarea **G** a circuitelor 74LS373 se va conecta câte o ieşire a decodicatorului de porturi, in cazul de mai jos semnalul **SD1**, iar intarea **OC** va fi conectată la GND
- de exemplu**, pentru cifra '0' va trebui sa incarcam: (1100 0000) deci segmentele ce vor lumina sunt: (f e d c b a)



Afisaj singular in EasyEDA:



4.2 Decodificarea porturilor

semnal	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	adresa port
SD1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0x6000
SD2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0x6002
SD3	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0x6800
SD4	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0x6802
SD5	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0x6100
SD6	0	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0x6102
SK1	0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0x6900
SK2	0	1	1	0	1	0	0	1	0	0	0	0	0	0	1	0	0x6902
SL1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0x6600
SL2	0	1	1	0	0	1	1	0	0	0	0	0	0	0	1	0	0x6602
S53	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0x6E00

SDx = semnal de selectie pentru display-ul de rang 'x'

SKx = semnal de selectie pentru tastatura

SLx = semnal de selectie pentru led-uri

S53 = semnal de selectie pentru timer-ul 8253A al interfetei seriale

4.3 Determinarea semnalelor de selectie

$SEL(SD1) = \sim A11 + \sim A10 + \sim A8 + \sim A1 \Rightarrow \sim Y0$ **ACTIV**

$SEL(SD2) = \sim A11 + \sim A10 + \sim A8 + A1 \Rightarrow \sim Y1$ **ACTIV**

$SEL(SD3) = A11 + \sim A10 + \sim A8 + \sim A1 \Rightarrow \sim Y8$ **ACTIV**

$SEL(SD4) = A11 + \sim A10 + \sim A8 + A1 \Rightarrow \sim Y9$ **ACTIV**

$SEL(SD5) = \sim A11 + \sim A10 + A8 + \sim A1 \Rightarrow \sim Y2$ **ACTIV**

$SEL(SD6) = \sim A11 + \sim A10 + A8 + A1 \Rightarrow \sim Y3$ **ACTIV**

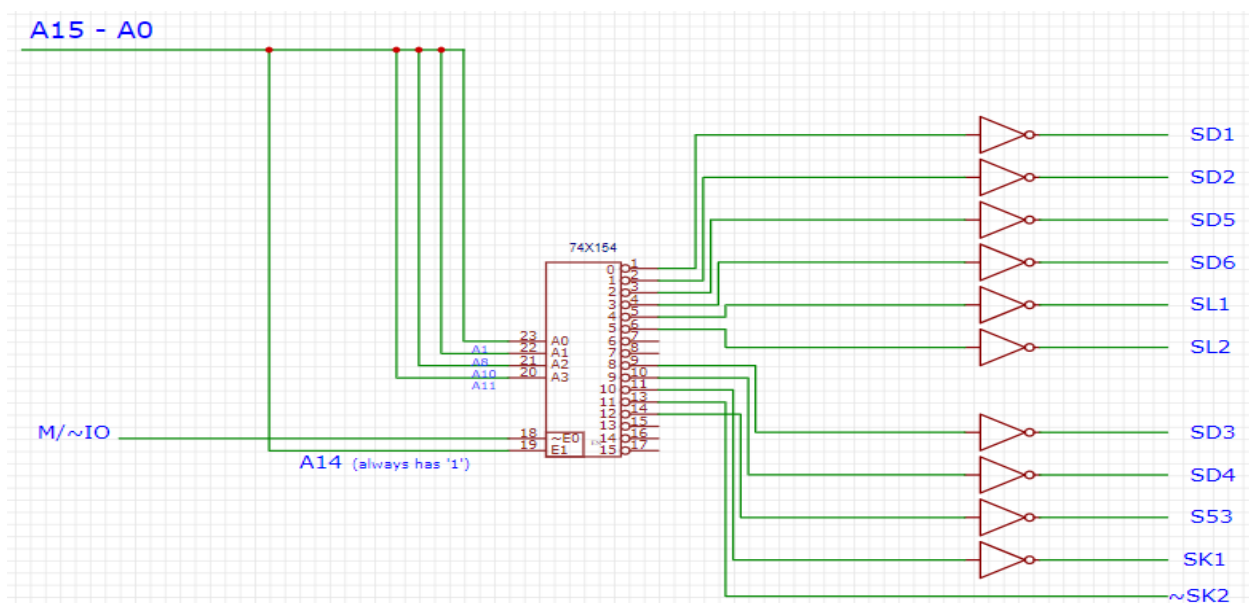
$SEL(SK1) = A11 + \sim A10 + A8 + \sim A1 \Rightarrow \sim Y10$ **ACTIV**

$SEL(SK2) = A11 + \sim A10 + A8 + A1 \Rightarrow \sim Y11$ **ACTIV**

$SEL(SL1) = \sim A11 + A10 + \sim A8 + \sim A1 \Rightarrow \sim Y4$ **ACTIV**

$SEL(SL2) = \sim A11 + A10 + \sim A8 + A1 \Rightarrow \sim Y5$ **ACTIV**

$SEL(S53) = A11 + A10 + \sim A8 + \sim A1 \Rightarrow \sim Y12$ **ACTIV**



5. Rutinele de programare

- a) rutinele de programare ale circuitelor 8251(i. seriala) si 8255(i. paralela)
- b) rutinele de emisie / receptie caracter pentru interfata seriala
- c) rutina de emisie caracter pentru interfata paralela
- d) rutina de scanare a tastaturii
- e) rutina de aprindere / stingere a unui LED
- f) rutina de afisare a unui caracter hexa pe un rang cu segmente

a) Rutina de programare a circuitului 8251

- **Cazul_1 (0x0AF0 – 0x0AF2)**, switch-ul **SI = 0**, unde adresele de port:

0x0AF0-comenzi/stari

0x0AF2-date

MOV AL, 0x0CE ; incarca in AL a cuvantul de mod (1100 1110): 2 biti de stop

OUT 0x0AF0, AL ; se transmite cuvantul de mod la adresa portului

MOV AL, 15H ; incarcarea in AL a cuvantului de stare

OUT 0x0AF0, AL ; se transmite la adresa portului

RET

- **Cazul_2 (0x0BF0 – 0x0BF2)**, switch-ul **SI = 1**, unde adresele de port:

0x0BF0-comenzi/stari

0x0BF2-date

MOV AL, 0CEH

OUT 0x0BF0, AL

MOV AL, 15H

OUT 0x0BF0, AL

RET

a) Rutina de programare a circuitului 8255

Adresele de port:

- **0x0C70** pentru portul A (sau **0x0D70**)
- **0x0C72** pentru portul B (sau **0x0D72**)
- **0x0C74** pentru portul C (sau **0x0D74**)
- **0x0C76** pentru portul RCC(sau **0x0D76**)

- **Cazul_1 (0x0C70 – 0x0C76), switch-ul PI = 0**

```
MOV DX, 0x0C76 ;incarcarea in registrul AL
MOV AL, 81H    ; se transmite la adresa portului RCC
OUT DX, AL
RET
```

- **Cazul_2 (0x0D70 – 0x0D76), switch-ul PI = 1**

```
MOV DX, 0x0D76
MOV AL, 81H
OUT DX, AL
RET
```

b) Rutina de emisie caracter pentru interfata seriala

Rutina începe prin citirea și testarea rangului TX_RDY din octetul de stare.

Dacă are valoarea '0' logic, înseamnă că circuitul 8251 încă nu poate primi un nou caracter de la microprocesor și acesta va fi nevoit să aștepte. În caz că are valoarea '1' logic, microprocesorul trimite date.

- **Cazul_1 (0x0AF0 – 0x0AF2), switch-ul SI = 0**

TR: IN AL, 0x0AF0 ;citesc cuvantul de stare de la adresa portului de comenzi

RCR AL, 1 ;rotire la dreapta cu o pozitie, carry bitul cel mai putin semnificativ

JNC TR ; pentru 0, reluam operatia

MOV AL, CL ;mut in AL cuvantul care trebuie transmis

OUT 0x0AF2, AL ;transmit caracterul la portul de date

RET

- **Cazul_2 (0x0BF0 – 0x0BF2), switch-ul SI = 1**

TR: IN AL, 0x0BF0

RCR AL, 1

JNC TR

MOV AL, CL

OUT 0x0BF2, AL

RET

b) Rutina de receptie caracter pentru interfata seriala

Rutina incepe prin citirea si testarea rangului RX_RDY din octetul de stare. Daca are valoarea '0' logic, inseamna ca circuitul inca nu are un caracter asamblat pentru microprocesor și acesta va fi nevoit să aștepte. În caz că are valoarea '1' logic, microprocesorul preia date și le depune în registrul CL.

- **Cazul_1 (0x0AF0 – 0x0AF2), switch-ul SI = 0**

```
REC: IN AL, 0x0AF0    ; preluare cuvant de stare de la portul de comenzi
      RCR AL, 1       ; se pune in flag-ul de carry bitul Rx_DRY
      JNC REC         ; daca Rx_DRY nu are valoarea '1', sar la RC
      IN AL, 0x0AF2   ; preiau caracterul de la portul de date
      MOV CL, AL      ; se depune data in registrul CL
      RET
```

- **Cazul_2 (0x0BF0 – 0x0BF2), switch-ul SI = 1**

```
REC: IN AL, 0x0BF0   ; preluare cuvant de stare de la portul de comenzi
      RCR AL, 1       ; se pune in flag-ul de carry bitul Rx_DRY
      JNC REC         ; daca Rx_DRY nu are valoarea '1', sar la RC
      IN AL, 0x0BF2   ; preiau caracterul de la portul de date
      MOV CL, AL      ; se depune data in registrul CL
      RET
```

c) Rutina de emisie caracter pentru interfata paralela

Rutina începe prin citirea și **testarea liniei BUSY** pentru a vedea dacă receptorul este liber. Microprocesorul așteaptă până când receptorul este liber și apoi trimite data, activând și dezactivând semnalul **~STB**.

- **Cazul_1 (0x0C70 – 0x0C76), switch-ul PI = 0**

```
PAR: IN AL, DX          ;citire si testare BUSY

      RCR AL, 1

      JNC PAR           ;asteptam ca procesorul sa fie liber

      MOV AL, CL        ;se preia caracterul din registrul CL

      MOV DX, 0x0C70

      OUT DX, AL        ;se trimite caracterul

      OR AL, 0x01

      MOV DX, 0x0C72

      OUT DX, AL        ; cazul cand ~STB = 1

      AND AL, 0x00

      OUT DX, AL        ; cazul cand ~STB = 0

      OR AL, 0x01

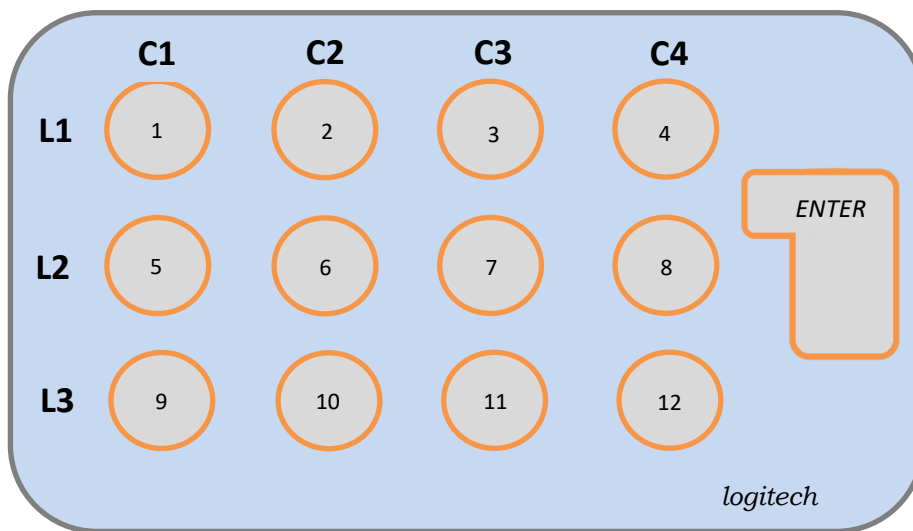
      OUT DX, AL        ; verificam din nou cazul cand ~STB = 1

      RET
```

- **Cazul_2 (0x0D70 – 0x0D76), switch-ul PI = 1**

```
PAR: IN AL, DX
      RCR AL, 1
      JNC PAR
      MOV AL, CL
      MOV DX, 0x0D70
      OUT DX, AL
      OR AL, 0x01
      MOV DX, 0x0D72
      OUT DX, AL
      AND AL, 0x00
      OUT DX, AL
      OR AL, 0x01
      OUT DX, AL
      RET
```

d) Rutina de scanare a tastaturii



!! IMPORTANT ! Tasta ce a fost apasata se stocheaza in registrul **CL**.

LOOP:

; se pune 0 pe prima coloană (C1) și se verifică tastele 1, 5, 9

MOV AL, 0x0FE ; activam C1

OUT 0x6900, AL

IN AL, 0x6902

AND AL, 0x01

JZ TASTA1 ; dacă este apăsată tasta 1, sar la TASTA1

IN AL, 0x6902

AND AL, 0x02

JZ TASTA5 ; dacă este apăsată tasta 5, sar la TASTA5

IN AL, 0x6902

AND AL, 0x04

JZ TASTA9 ; dacă este apăsată tasta 9, sar la TASTA9

; se pune 0 pe a doua coloană (C2) și se verifică tastele 2, 6, 10

MOV AL, 0x0FD ; activăm C2

OUT 0x6900, AL

IN AL, 0x6902

AND AL, 0x01

JZ TASTA2 ; dacă este apăsată tasta 2, sar la TASTA2

IN AL, 0x6902

AND AL, 0x02

JZ TASTA6 ; dacă este apăsată tasta 6, sar la TASTA6

IN AL, 0x6902

AND AL, 0x04

JZ TASTA10 ; dacă este apăsată tasta 10, sar la TASTA10

; Se pune 0 pe a treia coloană (C3) și se verifică tastele 3, 7, 11

MOV AL, 0x0FB ; activăm C3

OUT 0x6900, AL

IN AL, 0x6902

AND AL, 0x01

JZ TASTA3 ; dacă este apăsată tasta 3, sar la TASTA3

IN AL, 0x6902


```

AND AL, 0x02

JZ TASTA7      ; dacă este apăsată tasta 7, sar la TASTA7

IN AL, 0x6902

AND AL, 0x04

JZ TASTA11     ; Dacă este apăsată tasta 11, sar la TASTA11
; se pune 0 pe a patra coloană (C4) și se verifică tastele 4, 8, 12

MOV AL, 0x0F7  ; activăm C4

OUT 0x6900, AL

IN AL, 0x6902

AND AL, 0x01

JZ TASTA4      ; dacă este apăsată tasta 4, sar la TASTA4

IN AL, 0x6902

AND AL, 0x02

JZ TASTA8      ; dacă este apăsată tasta 8, sar la TASTA8

IN AL, 0x6902

AND AL, 0x04

JZ TASTA12     ; dacă este apăsată tasta 12, sar la TASTA12

JMP LOOP      ; reluăm scanarea

```

; rutina pentru fiecare tasta individuala

TASTA1: CALL DELAY ; stabilizarea contactelor

AST1: IN AL, 0x6902

AND AL, 0x01

JZ AST1 ; așteptăm dezactivarea tastei

CALL DELAY ; stabilizarea contactelor

MOV CL, 0x01 ; plasăm în registrul CL tasta citită

JMP LOOP ; reluăm scanarea

TASTA2: CALL DELAY ; stabilizarea contactelor

AST2:

IN AL, 0x6902

AND AL, 0x01

JZ AST2

CALL DELAY

MOV CL, 0x02

JMP LOOP

TASTA3: CALL DELAY ; stabilizarea contactelor

AST3:

IN AL, 0x6902

AND AL, 0x01

JZ AST3

CALL DELAY

MOV CL, 0x03

JMP LOOP

TASTA4: CALL DELAY ; stabilizarea contactelor

AST4:

IN AL, 0x6902

AND AL, 0x01

JZ AST4

CALL DELAY

MOV CL, 0x04

JMP LOOP

TASTA5: CALL DELAY ; stabilizarea contactelor

AST5:

IN AL, 0x6902

AND AL, 0x02

JZ AST5

CALL DELAY

MOV CL, 0x05

JMP LOOP

TASTA6: CALL DELAY ; stabilizarea contactelor

AST6:

IN AL, 0x6902

AND AL, 0x02

JZ AST6

CALL DELAY

MOV CL, 0x06

JMP LOOP

TASTA7: CALL DELAY ; stabilizarea contactelor

AST7:

IN AL, 0x6902

AND AL, 0x02

JZ AST7

CALL DELAY

MOV CL, 0x07

JMP LOOP

TASTA8: CALL DELAY ; stabilizarea contactelor

AST8:

IN AL, 0x6902

AND AL, 0x02

JZ AST8

CALL DELAY

MOV CL, 0x08

JMP LOOP

TASTA9: CALL DELAY ; stabilizarea contactelor

AST9:

IN AL, 0x6902

AND AL, 0x04

JZ AST9

CALL DELAY

MOV CL, 0x09

JMP LOOP

TASTA10: CALL DELAY ; stabilizarea contactelor

AST10:

IN AL, 0x6902

AND AL, 0x04

JZ AST10

CALL DELAY

MOV CL, 0x0A ; punem A ce inseamna 10 în hexazecimal

JMP LOOP

TASTA11: CALL DELAY ; stabilizarea contactelor

AST11:

IN AL, 0x6902

AND AL, 0x04

JZ AST11

CALL DELAY

MOV CL, 0x0B ; punem B ce inseamna 11 în hexazecimal

JMP LOOP

TASTA12: CALL DELAY ; stabilizarea contactelor

AST12:

IN AL, 0x6902

AND AL, 0x04

JZ AST12

CALL DELAY

MOV CL, 0x0C ; punem C ce inseamna 12 în hexazecimal

JMP LOOP

e) Rutina de aprindere/stingere a unui LED

!! IMPORTANT ! Starea dorita a ledurilor se trimite in **CX** (0-aprins,1-stins) si in **CL** va fi pus numarul led-ului folosite.

MOV DX, 0x6600 ; folosim semnalul SL1 pentru a selecta led-urile 1 – 8

; LED-UL 1

MOV AL, 0x7F ; in aceasta linie pornim led-ul dorit doar setand bitul coresp. pe '1' logic

OUT 0x00, AL ; trimitem val. din AL către portul de intrare-ieșire (M/~IO) de la adresa **0x00**

; LED-UL 2

MOV AL, 0xBF

OUT 0x00, AL

; LED-UL 3

MOV AL, 0xDF

OUT 0x00, AL

; LED-UL 4

MOV AL, 0xEF

OUT 0x00, AL

; LED-UL 5

MOV AL, 0xF7

OUT 0x00, AL

; LED-UL 6

MOV AL, 0xFB

OUT 0x00, AL

; LED-UL 7

MOV AL, 0xFD

OUT 0x00, AL

; LED-UL 8

MOV AL, 0xFE

OUT 0x00, AL

MOV DX, 0x6602 ; ca sa activa si led-urile 9 si 10 vom selecta semnalul SL2

; LED-UL 9

MOV AL, 0xFE

OUT 0x00, AL

; LED-UL 10

MOV AL, 0xFD

OUT 0x00, AL

e) Rutina pentru a stinge toate LED-urile

MOV DX, 0x6600 ; revenim la semnalul SL1

MOV AL, 0xFF ; **setam bitii tuturor led-urilor de pe semnalul SL1 ca fiind OFF**

OUT 0x00, AL ; trimitem valoarea actualizata din AL catre portul M/~IO

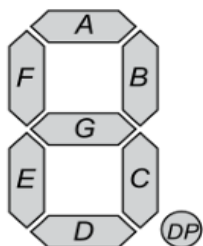
MOV DX, 0x6602 ; revenim la semnalul SL2

MOV AL, 0xFF ; stingem led-urile 9 si 10

OUT 0x00, AL

f) Rutina de afisare a unui caracter hexa pe un rang cu segmente

!! IMPORTANT ! In **CL** se afla caracterul in hexa si in **CH** se afla semnalul corespunzator rangului pe care vrem sa afisam.



Anod comun => ACTIV PE 0

Symbol	ABCD EFG (DP)	Hexadecimal
0	0000 0011b	03h
1	1001 1111b	9Fh
2	0010 0101b	25h
3	0000 1101b	0Dh
4	1001 1001b	99h
5	0100 1001b	49h
6	0100 0001b	41h
7	0001 1111b	1Fh
8	0000 0001b	01h
9	0000 1001b	09h
A	0001 0001b	11h
B	1100 0001b	C1h
C	0110 0011b	63h
D	1000 0101b	85h
E	0110 0001b	61h
F	0111 0001b	71h

<i>Rang</i>	<i>Semnal Activare</i>	<i>Zona de memorie</i>
1	000b	0x6000
2	001b	0x6002
3	010b	0x6800
4	011b	0x6802
5	100b	0x6100
6	101b	0x6102

DISPLAY:

CMP CL, 0x00

JZ **DISPLAY_0**

CMP CL, 0x01

JZ **DISPLAY_1**

CMP CL, 0x02

JZ **DISPLAY_2**

CMP CL, 0x03

JZ **DISPLAY_3**

CMP CL, 0x04

JZ **DISPLAY_4**

CMP CL, 0x05

JZ **DISPLAY_5**

CMP CL, 0x06

JZ **DISPLAY_6**

CMP CL, 0x07

JZ **DISPLAY_7**

CMP CL, 0x08

JZ **DISPLAY_8**

CMP CL, 0x09

JZ **DISPLAY_9**

CMP CL, 0x0A

JZ **DISPLAY_10**

CMP CL, 0x0B

JZ **DISPLAY_11**

CMP CL, 0x0C

JZ **DISPLAY_12**

CMP CL, 0x0D

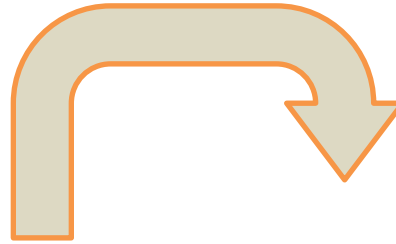
JZ **DISPLAY_13**

CMP CL, 0x0E

JZ **DISPLAY_14**

CMP CL, 0x0F

JZ **DISPLAY_15**



DISPLAY_0:

MOV AL, 0x03 ; punem in **AL** adresa cifrei curente

JMP DISPLAY ; revenim la bucla principala

DISPLAY_1:

MOV AL, 0x9F

JMP DISPLAY

DISPLAY_2:

MOV AL, 0x25

JMP DISPLAY

DISPLAY_3:

MOV AL, 0x0DH

JMP DISPLAY

DISPLAY_4:

MOV AL, 0x99

JMP DISPLAY

DISPLAY_5:

MOV AL, 0x49

JMP DISPLAY

DISPLAY_6:

MOV AL, 0x41

JMP DISPLAY

DISPLAY_7:

MOV AL, 0x1F

JMP DISPLAY

DISPLAY_8:

MOV AL, 0x01

JMP DISPLAY

DISPLAY_9:

MOV AL, 0x09

JMP DISPLAY

DISPLAY_10:

MOV AL, 0x11

JMP DISPLAY

DISPLAY_11:

MOV AL, 0xC1

JMP DISPLAY

DISPLAY_12:

MOV AL, 0x63

JMP DISPLAY

DISPLAY_13:

MOV AL, 0x85

JMP DISPLAY

DISPLAY_14:

MOV AL, 0x61

JMP DISPLAY

DISPLAY_15:

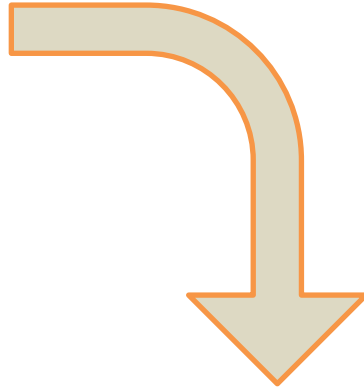
MOV AL, 0x71

JMP DISPLAY



DISPLAY_ALL_RANGS:

```
CMP CH, 000b
JZ DISPLAY_RANG_1
CMP CH, 001b
JZ DISPLAY_RANG_2
CMP CH, 010b
JZ DISPLAY_RANG_3
CMP CH, 011b
JZ DISPLAY_RANG_4
CMP CH, 100b
JZ DISPLAY_RANG_5
CMP CH, 101b
JZ DISPLAY_RANG_6
```



DISPLAY_RANG_1:

```
MOV DX, 0x6000 ; adresa de port pentru rang
OUT DX, AL ; trimitem valoarea din AL la DX
JMP END ; terminam executia programului
```

DISPLAY_RANG_2:

```
MOV DX, 0x6002
OUT DX, AL
JMP END
```

DISPLAY_RANG_3:

```
MOV DX, 0x6800
OUT DX, AL
JMP END
```

DISPLAY_RANG_4:

```
MOV DX, 0x6802
OUT DX, AL
JMP END
```

DISPLAY_RANG_5:

```
MOV DX, 0x6100 ;
OUT DX, AL
JMP END
```

DISPLAY_RANG_6:

```
MOV DX, 0x6102
OUT DX, AL
JMP END
```

END:

```
RET ; scurtcircuitam executia codului rutinei
```

❖ Bibliografie

- curs 'Proiectarea cu Microprocesoare' tinut de prof. Mircea Popa
- lab 'Proiectarea cu Microprocesoare' tinut de Sebastian Petruc
- https://www.ti.com/lit/ds/symlink/74ac11244.pdf?ts=1735655579340&ref_url=https%253A%252F%252Fwww.google.com%252F
- <https://www.igelectronics.com/products/j98fb34fc5/1726810000000230778>
- <https://www.alldatasheet.com/datasheet-pdf/pdf/66096/INTEL/8251A.html>
- schemele sunt realizate cu suport EasyEDA
- [EasyEDA - Online PCB design & circuit simulator](#)